Optimal Spatial Registration of SLAM for Augmented Reality

dem Fachbereich Informatik
der Technischen Universität Darmstadt
vorzulegende

DISSERTATION

zur Erlangung des akademischen Grades eines
Doktor-Ingenieurs (Dr.-Ing.)
von

Folker Wientapper

geboren in Caracas, Venezuela

Referenten der Arbeit:	Prof. Dr. Arjan Kuijper Technische Universität Darmstadt Prof. Dr. techn. Dieter W. Fellner Technische Universität Darmstadt Prof. Dr. Didier Stricker Technische Universität Kaiserslautern
Tag der Einreichung:	31.01.2019
Tag der mündlichen Prüfung:	15.03.2019

Darmstädter Dissertation
D 17

Wientapper, Folker : Optimal Spatial Registration of SLAM for Augmented Reality
Darmstadt, Technische Universität Darmstadt,
Jahr der Veröffentlichung der Dissertation auf TUprints: 2019
Tag der mündlichen Prüfung: 15.03.2019

Erklärung zur Dissertation

Hiermit versichere ich, die vorliegende Dissertation selbständig nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die aus Quellen entnommen wurden, sind als solche kenntlich gemacht. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Darmstadt, den 31.01.2019 Folker Wientapper

Acknowledgments

I am grateful for the time I spent at Fraunhofer IGD as a research assistant. The opportunity to work in this environment and to be able to contribute to exciting scientific topics and technical problems is not a matter of course, but rather a privilege that enabled me to pursue this dissertation in the first place. I would like to thank all the people who contributed to this opportunity and who accompanied me on my way.

I would like to thank my supervisor Prof. Arjan Kuijper for guiding my doctor study, for his time spent during reviews and discussions, for his numerous and valuable advices, and for keeping me focused.

Thanks to Prof. Dieter Fellner for being my secondary advisor, for reviewing my thesis, for fostering research and promoting scientific work at the IGD, for his critical mind, and for the fruitful discussions during 'Abteilungsdemos'.

Thanks to Prof. Didier Stricker as third supervisor. As the former leader of the department 'Virtual and Augmented Reality' (VRAR) at the IGD, he paved the way for me to become a member of this group. Despite the fact that we had only worked together for a short time, he sparked my enthusiasm for AR and computer vision, which still lives in me today.

Thanks to Uli Bockholt for successfully leading the VRAR department over the many years that followed, enabling me to work on a variety of exciting AR research projects in a collegial environment.

I would also like to thank all other colleagues, in particular Timo Engelke, my longtime friend and roommate, Harald Wuest, friend, counselor, and initial mentor when I was still HiWi, as well as Jens Keil, Michael Schmitt, Sabine Webel, Patrick Riess and Hugo Binder, who together we successfully managed a variety of exciting AR projects at the institute.

Thanks to Holger Graf, Jens Keil, Max Limper, and Margaret Klimmek for proofreading and providing valuable advice.

I would like to especially thank my parents, Marlis and Helmut Wientapper, who have supported me in every situation. Thanks to my wife Petra for her patience and support. Thanks to my two daughters, Anna and Cleo, for the many joyful moments in between.

Abstract

Augmented reality (AR) is a paradigm that aims at fusing the perceived real environment of a human with digital information located in 3D space. Typically, virtual 3D graphics are overlayed into the captured images of a moving camera or directly into the user's field-of-view by means of optical see-through displays (OST). For a correct perspective and view-dependent alignment of the visualization, it is required to solve various static and dynamic geometric registration problems in order to create the impression that the virtual and the real world are seamlessly interconnected.

The advances during the last decade in the field of simultaneous localization and mapping (SLAM) represent an important contribution to this general problem. It is now possible to reconstruct the real environment and to simultaneously capture the dynamic movements of a camera from the images without having to instrument the environment in advance. However, SLAM in general can only partly solve the entire registration problem, because the retrieved 3D scene geometry and the calculated motion path are spatially related only with regard to an arbitrarily selected coordinate system. Without a proper reconciliation of coordinate systems (*spatial registration*), the real world of the human observer still remains decoupled from the virtual world. Existing approaches for solving this problem either require the availability of a virtual 3D model that represents a real object with sufficient accuracy (model-based tracking), or they rely on use-case specific assumptions and additional sensor data (such as GPS signals or the Manhattan-world assumption). Therefore, these approaches are bound to these additional prerequisites, which limit the general applicability. The circumstance that automated registration is desirable but not always possible, creates the need for techniques that allow a user to specify connections between the real and the virtual world when setting up AR applications, so that it becomes possible to support and control the process of registration. These techniques must be complemented with numerical algorithms that optimally exploit the provided information to obtain precise registration results.

Within this context, the present thesis provides the following contributions.

- We propose a novel, closed-form (non-iterative) algorithm for calculating a Euclidean or a similarity transformation. The presented algorithm is a generalization of recent state-of-the-art solvers for computing the camera pose based on 2D measurement points in the image (*perspective-n-point problem*) - a fundamental problem in computer vision that has attracted research for many decades. The generalization consists in extending and unifying these algorithms, so that they can handle other types of input correspondences than originally designed for. With this algorithm, it becomes possible to perform a *rigid registration* of SLAM systems to a target coordinate system based on heterogeneous and partially indeterminate input data.
- We address the global refinement of structure and motion parameters by means of iterative sparse minimization (bundle adjustment or BA), which has become a standard technique inside SLAM systems. We propose a variant of BA in which information about the virtual domain is integrated as constraints by means of an optimization-on-manifold approach. This serves for compensating low-frequency deformations (*non-rigid registration*) of the estimated camera path and the reconstructed scene geometry caused by measurement error accumulation and the ill-conditionedness of the BA problem.
- We present two approaches in which a user can contribute with his knowledge for registering a SLAM system. In a first variant, the user can place markers in the real environment with predefined connections to the virtual coordinate system. Precise positioning of the markers is not required, rather they can be

placed arbitrarily on surfaces or along edges, which notably facilitates the preparative effort. During run-time, the dispersed information is collected and registration is accomplished automatically. In a second variant, the user is given the possibility to mark salient points in an image sequence during a preparative preprocessing step and to assign corresponding points in the virtual 3D space via a simple point-and-click metaphor. The result of this preparative phase is a precisely registered and ready-to-use reference model for camera tracking at run-time.

- Finally, we propose an approach for geometric calibration of optical see-trough displays. We present a parametric model, which allows to dynamically adapt the rendering of virtual 3D content to the current viewpoint of the human observer, including a pre-correction of image aberrations caused by the optics or irregularly curved combiners. In order to retrieve its parameters, we propose a camera-based approach, in which elements of the real and the virtual domain are simultaneously observed. The calibration procedure was developed for a head-up display in a vehicle. A prototypical extension to head-mounted displays is also presented.

Zusammenfassung

Erweiterte Realität (AR) bezeichnet ein Paradigma, welches darauf abzielt, die wahrgenommene, reale Umgebung eines Menschen mit im 3D-Raum verorteten, digitalen Informationen zu verschmelzen. Typischerweise werden dabei in Echtzeit virtuelle 3D Grafiken in die aufgenommenen Bilder einer sich bewegenden Kamera oder direkt in das Sichtfeld des Nutzers über optische Durchsichtanzeigen eingebettet. Die perspektivisch korrekte und lagerichtige Darstellung erfordert hierzu die Lösung verschiedener statischer und dynamischer Registrierungsprobleme, um den Eindruck zu erzeugen, dass die virtuelle und die reale Welt nahtlos miteinander verbunden sind.

Die im letzten Jahrzehnt erreichten Fortschritte im Bereich simultaner Lokalisierung und Kartierung (SLAM) liefern hierzu einen wichtigen Beitrag. Hiermit ist es möglich, die reale Umgebung zu rekonstruieren und dabei gleichzeitig die dynamische Eigenbewegung einer Kamera aus den Bildern zu erfassen, ohne dass die Umgebung hierzu präpariert werden muss. Dennoch löst SLAM damit nur einen Teil des gesamten Registrierungsproblems, da die erstellte 3D Szenengeometrie und der berechnete Bewegungspfad räumlich nur in Bezug zu einem frei gewählten Koordinatensystem gesetzt wird. Ohne einen entsprechenden Abgleich der Koordinatensysteme bleibt die reale Welt des menschlichen Beobachters stets von der virtuellen Welt entkoppelt. Bestehende Ansätze zur Lösung dieses Problems erfordern entweder die Verfügbarkeit eines virtuellen 3D-Modells, welches einem realen Objekt mit ausreichender Genauigkeit entsprechen muss (modellbasiertes Tracking), oder sie stützen sich auf anwendungsfallspezifische Annahmen und zusätzliche Sensordaten (wie GPS-Signale oder der "Manhattan Welt"-Annahme), welches die allgemeine Anwendbarkeit dieser Verfahren einschränkt. Der Umstand, dass eine automatisierte Registrierung wünschenswert jedoch nicht immer möglich ist, schafft den Bedarf an Techniken, mit denen ein Benutzer beim Einrichten von AR-Anwendungen Verbindungen zwischen der realen und der virtuellen Welt spezifizieren und somit die Registrierung begleiten und kontrollieren kann. Diese Techniken benötigen die Unterstützung durch numerische Algorithmen, welche die Informationen optimal ausnutzen, um somit genaue Registrierungsergebnisse zu erreichen.

In diesem Zusammenhang liefert die vorliegende Arbeit die folgenden Beiträge.

- Es wird ein neuartiger, nicht-iterativer Algorithmus zur Berechnung einer euklidischen Transformation oder einer Ähnlichkeitstransformation präsentiert. Der vorgestellte Algorithmus stellt eine Verallgemeinerung neuester Ansätze zur Berechnung der Kameraposition und -Orientierung anhand von 2D-Messpunkten im Bild dar (*räumlicher Rückwärtsschnitt*) - ein grundlegendes Problem im Bereich des maschinellen Sehens mit einer langen Forschungshistorie. Die Verallgemeinerung besteht darin, diese Algorithmen so zu erweitern und zu vereinheitlichen, dass sie mit anderen Arten von Eingangskorrespondenzen als ursprünglich vorgesehen umgehen können. Der Algorithmus ermöglicht es, eine *rigide Registrierung* von SLAM-Systemen zu einem Zielkoordinatensystem auf der Grundlage heterogener und partiell unbestimmter Eingangsdaten durchzuführen.
- Zudem wird die globale Minimierung von Struktur- und Bewegungsparametern durch den Bündelblockausgleich (BA) addressiert, welcher sich als Standardtechnik innerhalb von SLAM-Systemen etabliert hat. In dieser Arbeit wird hierzu eine Variante des BA vorgeschlagen, bei der Informationen über die virtuelle Domäne als Gleichheitsnebenbedingungen integriert werden, wobei hierfür als Technik eine Parameteroptimierung entlang ihrer zugehörigen Mannigfaltigkeiten gewählt wird. Die Integration der Nebenbedi-

gungen erfolgt mit dem Ziel, niederfrequente Deformationen (*nicht-rigide Registrierung*) des geschätzten Kamerapfades und der rekonstruierten Szenengeometrie zu kompensieren, welche durch die Messfehlerakkumulation und die schlechte Konditioniertheit des BA-Minmierungsproblems verursacht werden.

- Weiterhin werden zwei Ansätze vorgestellt, bei denen ein Benutzer mit seinem Wissen zur Registrierung eines SLAM-Systems beitragen kann. In einer ersten Variante kann der Benutzer in der realen Umgebung Marker mit vordefinierten Verbindungen zum virtuellen Koordinatensystem platzieren. Eine genaue Positionierung der Marker ist nicht erforderlich, vielmehr können sie beliebig auf Oberflächen oder entlang von Kanten platziert werden, was den präparativen Aufwand erheblich reduziert. Zur Laufzeit werden dann die verteilten Informationen erfasst und die Registrierung des SLAM erfolgt automatisch. In einer zweiten Variante hat der Benutzer im Rahmen eines Vorverarbeitungsschrittes die Möglichkeit, markante Punkte in einer bereits aufgenommenen Bildsequenz zu selektieren und entsprechenden Punkten im virtuellen 3D-Raum über eine einfache Nutzerschnittstelle zuzuordnen. Das Ergebnis dieser Vorbereitung ist ein präzise registriertes Referenzmodel, welches zur Laufzeit unmittelbar zur Kameraverortung eingesetzt werden kann.
- Schließlich wird ein Ansatz zur geometrischen Kalibrierung optischer Durchsichtanzeigen vorgeschlagen. Es wird ein parametrisches Modell vorgestellt, mit dem das Rendern von virtuellem 3D-Inhalt an den aktuellen Blickpunkt des menschlichen Beobachters dynamisch angepasst werden kann, einschließlich einer Vorkorrektur der durch die Optik oder unregelmäßig gekrümmter Kombinatoren verursachten Bildverzerrungen. Um die Modellparameter zu ermitteln, wird ein kamerabasierter Ansatz vorgeschlagen, bei dem Elemente der realen und der virtuellen Domäne gleichzeitig erfasst werden. Das Kalibrierungsverfahren wurde examplarisch für ein Head-up-Display in einem Fahrzeug entwickelt. Eine prototypische Erweiterung für Head-Mounted-Displays wird ebenfalls vorgestellt.

Mathematical Notation

Throughout this thesis, we will use the following notation for mathematical formulas.

General Character Formatting

a, A :	*scalars, indexes, mappings and functions*
$\mathbf{a}$:	*column vectors*
$\mathbf{A}$:	*matrices*
$\mathcal{A}$:	*coordinate frames, subspaces, manifolds, probability distributions*
$\mathbb{A}$:	*fields, sets, or subsets, esp.* $\mathbb{N}, \mathbb{Z}, \mathbb{R}, \mathbb{C}$ *denote the whole, integer, real and complex numbers*

Operators

$\mathbf{a}^\mathsf{T}, \mathbf{A}^\mathsf{T}$:	*vector transpose, matrix transpose*
$\mathbf{AB}$ *or* $\mathbf{A} \cdot \mathbf{B}$:	*general matrix product of two matrices* $\mathbf{A}$ *and* $\mathbf{B}$
$\mathbf{a}^\mathsf{T}\mathbf{b}$ *or* $\mathbf{a}^\mathsf{T} \cdot \mathbf{b}$:	*(special case) inner product / dot product of two vectors*
$\mathbf{a}\mathbf{b}^\mathsf{T}$ *or* $\mathbf{a} \cdot \mathbf{b}^\mathsf{T}$:	*(special case) outer product / dyadic product of two vectors*
$\mathbf{a} \times \mathbf{b}$:	*cross product for* $\mathbf{a}, \mathbf{b} \in \mathbb{R}^3$
$[\mathbf{a}]_\times$:	*cross-product matrix for a vector* $\mathbf{a} \in \mathbb{R}^3$ *with* $[\mathbf{a}]_\times = \begin{bmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{bmatrix}$
$\mathbf{a} \propto \mathbf{b}$:	*parallelism / identity up to scale for vectors* $\mathbf{a}, \mathbf{b}$
$\mathbf{A} \otimes \mathbf{B}$:	*Kronecker product*
$\mathbf{A}^{-1}, A^{-1}$:	*inverse of an invertible matrix or mapping*

Special, Frequently Used Characters

s :	*scaling factor*
$\mathbf{q}$:	*quaternion vector. For* $\mathbf{q} = [q_0, q_1, q_2, q_3]^\mathsf{T}$, q_0 denotes the real part
$\mathbf{t}$:	*translation vector in* $\mathbb{R}^3$
$\mathbf{x}, \mathbf{y}$:	*points / offset vectors, usually in* $\mathbb{R}^3$
$\mathbf{J}$:	*Jacobian matrix*
$\mathbf{K}$:	3×3 *camera calibration matrix*
$\mathbf{R}$:	3×3 *rotation matrix*
$\mathcal{C}$:	*(C)amera coordinate system*
$\mathcal{V}$:	*coordinate system of a (V)irtual model*
$\mathcal{W}$:	*coordinate system of reconstructed real (W)orld model*
E :	*Euclidean transformation*
S :	*similarity transformation*

Coordinate Systems, Transformations and Subscript Notation

When working with coordinate systems, it often comes in handy to use a subscript notation for the variables. We will distinguish between parameters that can be expressed as coordinates in some coordinate system, $\mathcal{A}$, and parameters that represent a mapping between two coordinate systems, e.g. $\mathcal{A}$ and $\mathcal{B}$. The former usually refer to points and vectors, and they will be marked by a single subscript:

$\mathbf{x}_\mathcal{A}$: *a point or vector expressed in coordinate system* $\mathcal{A}$.

Mappings that represent coordinate transformations will be given two coordinate system subscripts connected with an arrow representing the transformation from one to another:

$T_{\mathcal{A}\to\mathcal{B}}$: *a mapping representing a coordinate transformation (usually* $\mathbb{R}^3 \to \mathbb{R}^3$*) from coordinate system* $\mathcal{A}$ *to* $\mathcal{B}$.

In this thesis, we are particularly concerned with translations, rotations, Euclidean and similarity transformations of points or vectors in $\mathbb{R}^3$. When switching to a higher dimensional space using homogeneous coordinates $[\mathbf{x}^\mathsf{T}, 1]^\mathsf{T}$, these transformations can be represented by linear mappings, i.e. matrices $\mathbf{T} \in \mathbb{R}^{4\times4}$, where a transformation reduces to a simple matrix-vector product. Based on this one-to-one correspondence to linear algebra, we will define a coordinate transformation by a left-handed multiplication in analogy to a matrix product:

$\mathbf{x}_\mathcal{B} = T_{\mathcal{A}\to\mathcal{B}} \cdot \mathbf{x}_\mathcal{A}$: *a transformation of a point or vector* $\mathbf{x}$ *from* $\mathcal{A}$ *to* $\mathcal{B}$ *(product form).*

From $\mathbf{x}_\mathcal{B} = T_{\mathcal{A}\to\mathcal{B}} \cdot \mathbf{x}_\mathcal{A}$ and $\mathbf{x}_\mathcal{C} = T_{\mathcal{B}\to\mathcal{C}} \cdot \mathbf{x}_\mathcal{B}$ it follows that $\mathbf{x}_\mathcal{C} = T_{\mathcal{B}\to\mathcal{C}} \cdot T_{\mathcal{A}\to\mathcal{B}} \cdot \mathbf{x}_\mathcal{A}$. Hence, a transformation can itself be subject to a coordinate system change as follows:

$T_{\mathcal{A}\to\mathcal{C}} = T_{\mathcal{B}\to\mathcal{C}} \cdot T_{\mathcal{A}\to\mathcal{B}}$: *change of basis of a transformation (product form).*

We define the inverse of a transformation by swapping of subscripts:

$$T_{\mathcal{A}\to\mathcal{B}}^{-1} := T_{\mathcal{B}\to\mathcal{A}} : \qquad \textit{inverse of a transformation.}$$

Some simple rules follow from this definition on subscripts. In particular, for concatenations of transformations we have:

$$\begin{aligned} T_{\mathcal{A}\to\mathcal{C}}^{-1} &= T_{\mathcal{C}\to\mathcal{A}} \\ &= T_{\mathcal{B}\to\mathcal{A}} \cdot T_{\mathcal{C}\to\mathcal{B}} \quad : \quad \textit{inverse of a concatenation of transformations.} \\ &= T_{\mathcal{A}\to\mathcal{B}}^{-1} \cdot T_{\mathcal{B}\to\mathcal{C}}^{-1} \end{aligned}$$

For a product of two transformations, $T_{\mathcal{B}\to\mathcal{C}} \cdot T_{\mathcal{A}\to\mathcal{B}}$, the left subscript of the left operand and the right subscript of the right operand must always be identical (in this case $\mathcal{B}$). When working with many different coordinate systems, this property offers a simple method for validating the correctness of derived formulas. If this subscript rule is violated, there is most likely an error in the formulation.

Similarity transformations, $S_{\mathcal{A}\to\mathcal{B}} \in sim(3)$, consist of a rotation $\mathbf{R}_{\mathcal{A}\to\mathcal{B}} \in SO(3)$, a translation $\mathbf{t}_{\mathcal{A}\to\mathcal{B}} \in \mathbb{R}^3$, and a scaling factor $s_{\mathcal{A}\to\mathcal{B}} \in \mathbb{R}_+$. The used notation and some basic rules are exemplified in the table below. Note that this summary also applies to Euclidean transformations, $E_{\mathcal{A}\to\mathcal{B}} \in SE(3)$ (similarity transform with unit scale), as well as pure rotations or translations.

Notations for similarity transformations and their interpretation

Argument of transformation / Description	*Short-hand notation*	*Explicit formula / Meaning*
Points $\mathbf{x} \in \mathbb{R}^3$	$\mathbf{x}_{\mathcal{B}} = S_{\mathcal{A}\to\mathcal{B}} \cdot \mathbf{x}_{\mathcal{A}}$	$\mathbf{x}_{\mathcal{B}} = s_{\mathcal{A}\to\mathcal{B}} \cdot \mathbf{R}_{\mathcal{A}\to\mathcal{B}} \cdot \mathbf{x}_{\mathcal{A}} + \mathbf{t}_{\mathcal{A}\to\mathcal{B}}$
Vectors $\mathbf{v} \in \mathbb{R}^3$	$\mathbf{v}_{\mathcal{B}} = S_{\mathcal{A}\to\mathcal{B}} \cdot \mathbf{v}_{\mathcal{A}}$	$\mathbf{v}_{\mathcal{B}} = s_{\mathcal{A}\to\mathcal{B}} \cdot \mathbf{R}_{\mathcal{A}\to\mathcal{B}} \cdot \mathbf{v}_{\mathcal{A}}$
Gaussian distributions on points $\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, with: $\boldsymbol{\mu} \in \mathbb{R}^3, \boldsymbol{\Sigma} \in \mathbb{R}^{3\times3}$	$\mathbf{x}_{\mathcal{B}} = S_{\mathcal{A}\to\mathcal{B}} \cdot \mathbf{x}_{\mathcal{A}}$	$\mathbf{x}_{\mathcal{B}} \sim \mathcal{N}(\boldsymbol{\mu}_{\mathcal{B}}, \boldsymbol{\Sigma}_{\mathcal{B}})$, with: $\boldsymbol{\mu}_{\mathcal{B}} = s_{\mathcal{A}\to\mathcal{B}} \cdot \mathbf{R}_{\mathcal{A}\to\mathcal{B}} \cdot \boldsymbol{\mu}_{\mathcal{A}} + \mathbf{t}_{\mathcal{A}\to\mathcal{B}}$ $\boldsymbol{\Sigma}_{\mathcal{B}} = s_{\mathcal{A}\to\mathcal{B}}^2 \cdot \mathbf{R}_{\mathcal{A}\to\mathcal{B}} \cdot \boldsymbol{\Sigma}_{\mathcal{A}} \cdot \mathbf{R}_{\mathcal{A}\to\mathcal{B}}^{\mathsf{T}}$
Other similarity transformations $S \in Sim(3)$	$S_{\mathcal{A}\to\mathcal{C}} = S_{\mathcal{B}\to\mathcal{C}} \cdot S_{\mathcal{A}\to\mathcal{B}}$	$\mathbf{R}_{\mathcal{A}\to\mathcal{C}} = \mathbf{R}_{\mathcal{B}\to\mathcal{C}} \cdot \mathbf{R}_{\mathcal{A}\to\mathcal{B}}$ $\mathbf{t}_{\mathcal{A}\to\mathcal{C}} = \mathbf{t}_{\mathcal{B}\to\mathcal{C}} + s_{\mathcal{B}\to\mathcal{C}} \cdot \mathbf{R}_{\mathcal{B}\to\mathcal{C}} \cdot \mathbf{t}_{\mathcal{A}\to\mathcal{B}}$ $s_{\mathcal{A}\to\mathcal{C}} = s_{\mathcal{B}\to\mathcal{C}} \cdot s_{\mathcal{A}\to\mathcal{B}}$
Inverse $S^{-1} \in Sim(3)$	$S_{\mathcal{A}\to\mathcal{B}}^{-1} = S_{\mathcal{B}\to\mathcal{A}}$	$\mathbf{R}_{\mathcal{B}\to\mathcal{A}} = \mathbf{R}_{\mathcal{A}\to\mathcal{B}}^{\mathsf{T}}$ $\mathbf{t}_{\mathcal{B}\to\mathcal{A}} = -s_{\mathcal{A}\to\mathcal{B}}^{-1} \cdot \mathbf{R}_{\mathcal{A}\to\mathcal{B}}^{\mathsf{T}} \cdot \mathbf{t}_{\mathcal{A}\to\mathcal{B}}$ $s_{\mathcal{B}\to\mathcal{A}} = s_{\mathcal{A}\to\mathcal{B}}^{-1}$

List of Abbreviations and Acronyms

AD	*automatic or algorithmic differentiation*
AR	*augmented reality*
BA	*bundle adjustment*
BIM	*building information model or building information management*
CAD	*computer aided design*
CAS	*computer algebra software*
DoF	*degrees-of-freedom*
EKF	*extended Kalman filter*
GAPS	*General-purpose Absolute Pose Solver*
GIS	*geographic information system*
GN	*Gauss-Newton*
HMD	*head-mounted display*
HUD	*head-up display*
IMU	*inertial measurement unit*
LM	*Levenberg-Marquardt*
OOM	*optimization-on-manifold*
OST	*optical see-through*
PCG	*preconditioned conjugated gradients*
PnP / PnL	*perspective-n-point / perspective-n-line (problem)*
PLM	*product life management (software)*
SfM	*structure from motion*
SLAM	*simultaneous localization and mapping*
VST	*video see-through*
VP	*vanishing point*

Contents

1. Introduction

1.1. Geometric Registration for Visual Augmented Reality

Augmented reality (AR) is a paradigm that aims to complement the perceived real environment of a human with virtual, digital information placed in 3D space. In contrast to *Virtual Reality*, where the user is immersed in a completely synthetic environment, augmented reality aims at integrating the virtual elements as part of the real environment of the user. According to a widely accepted definition by Azuma [Azu97] from 1997, it has three main characteristics: it combines real and virtual elements, it is interactive in real-time and the presented information is registered in 3D. Although this definition is not limited to visual perception only[1], most AR variants are targeted at overlaying 3D graphics onto the captured real images of a moving camera or into the user's field-of-view. At that time, registration was already considered as one of the major challenges in AR. The term *registration* was used to label the general objective that the displayed virtual objects appear at the right position and perspectively correct from the user's view. It is a necessary requirement for AR in order to create the illusion that they coexist with the real elements in the scene.

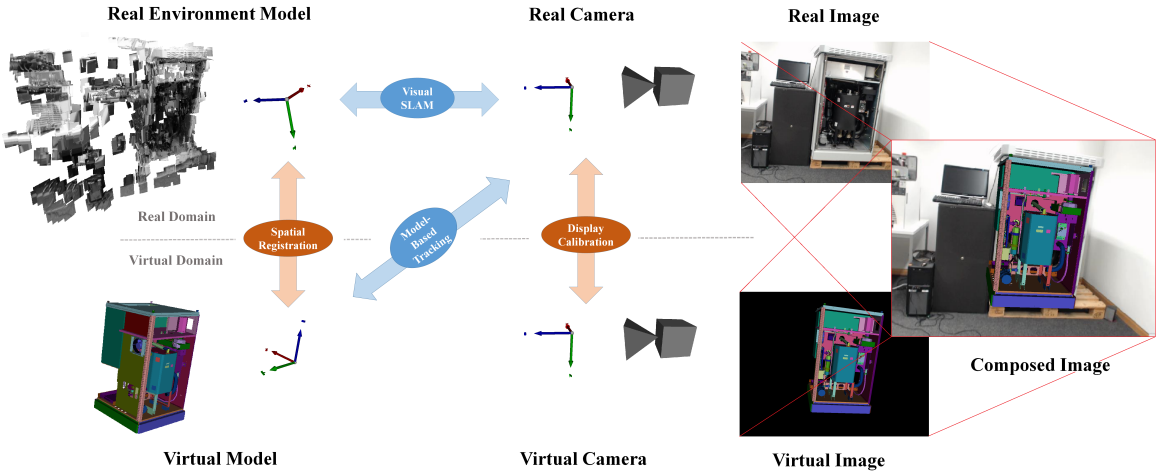


Figure 1.1.: Principle of AR and the roles of different subtasks in solving the overall registration problem (adapted figure from [MUS16]).

Typically, in visual AR the displayed virtual content is synthesized within a standard graphics pipeline. Rendering is controlled by setting a *virtual camera* from which the image of the virtual scene is produced. For real-time AR, its parameters need to be directly coupled with an estimate of the user's or the real camera's view-

[1] Augmented reality can in principle also address auditory, haptic or olfactory data.

point. After rendering, the real and the virtual image are composited together to create the augmented view (see Fig. 1.1, right).

High precision is important for many applications, whenever the virtual content is strongly linked to the geometry of the real scene. For example, in industrial quality inspection applications virtual models are overlaid onto the real objects for comparing their state as planned with the state after manufacturing. In this case, it is important that local deviations in the overlay can be identified as differences between the real and the desired state and do not occur as the result of errors in the registration process.

For an accurate overlay of the virtual content, various subproblems need to be solved. Regarding the terminology, these can be classified into 'tracking', 'spatial registration', and 'calibration' tasks [SH16]:

- *Tracking* refers to the continuous and dynamic update of parameters by means of measurements, in particular the pose of the AR display or user's viewpoint relative to the real world.
- *Spatial registration or alignment* comprises the reconciliation of coordinate systems, especially between the virtual world and the internal coordinate system of the motion tracking system.
- *Calibration* is the process of comparing measurements to a reference device or known values and is usually part of an offline, preprocessing step.

From a first order perspective, registration for visual AR can be considered as a dynamic motion estimation or *tracking* problem. In principle, there exist numerous sensing technologies for capturing the motion, such as mechanical, inertial, acoustical (ultrasonic), electromagnetic (including GPS), and optical sensors [WF02, vKP10]. Although the choice of the best-suited technology is application-dependent, vision-based ego-motion tracking, as one instance of optical sensing, quickly became popular for AR. This was also partly driven by the emergence of mobile devices (such as phones and tablets), which have both a camera and a display tightly integrated in one portable piece of hardware, which generally mitigates the overall calibration and registration demands compared to other setups.

Among the vision-based approaches, simultaneous localization and mapping (SLAM) has evolved as one of the key techniques for estimating the motion of a moving camera in unprepared environments. From the estimated camera trajectory, SLAM reconstructs a static and sparse 3D model of the surrounding environment (real world model), which in turn serves to estimate the pose of the real camera relative to it. SLAM has been an active research area for more than a decade, and recently the major software companies started releasing SLAM solutions for AR, such as the Hololens by Microsoft [Mic16], ARKit by Apple [App17], and ARCore by Google [Goo17].

However, SLAM can only partly solve the overall registration problem, since in absence of any prior information on the scene, it can only estimate the relative motion of the camera. This means that the coordinate system, in which the 3D model of the real environment and the camera path are reconstructed, is chosen arbitrarily. Usually the camera's position and orientation belonging to the first image is selected as the reference coordinate frame. As a result, the output of the tracking will vary randomly on every start of the AR application, if visual SLAM alone is used. Ultimately, rendering requires the viewpoint of the camera in model coordinates rather than in the coordinate system of the reconstructed model. Therefore it is necessary to retrieve the static relation between the two coordinate systems by means of a *spatial registration*.

Besides tracking and spatial registration, another important problem concerns the relationship between the tracking system and the viewpoint for rendering. In handheld or video see-through AR, where the virtual content is superimposed on the captured images, the situation is simple. In this case, the user - while looking at the displayed camera image - observes the real environment from the camera's perspective, not from his own. Perspective correct rendering only requires the knowledge of the intrinsic and extrinsic parameters of the physical camera that is used for tracking. By contrast, in the case of optical see-through head-mounted Displays (HMD)

or head-up displays (HUD), the virtual content is directly overlaid into the user's view by means of a semitransparent mirror. The composition of the virtual and the real image actually happens on the retina in his eyes. As the user simultaneously observes the real environment from his own perspective, the virtual camera for rendering must emulate how he perceives the world. The observation point of the user (strictly speaking, the optical center of the eyes) and the virtual display plane together form a physical setup which can also be approximated by a pinhole camera model (real camera). Thus, rendering for optical see-through displays can be accomplished with a standard graphics pipeline, but a proper display calibration is necessary to retrieve the appropriate parameters.

To summarize, the goal of geometric registration is - in simple words - to render the virtual content from the same perspective as from which the user perceives the real world surrounding him. SLAM as a nowadays mature technique provides a robust estimate of the relative motion trajectory. But without references to the virtual world, a SLAM-driven AR system remains mostly decoupled from the real world surrounding the user [Azu16]. The idea that something like a 'same perspective' exists implies that there must also exist links between the real and the virtual world, i.e. equivalence relations, which specify what belongs together on either side. In this context, we pose the following research questions:

1. What is the source (or underlying assumption) of these links or who provides it?
2. How can we exploit this information computationally in an optimal way for spatial registration of a SLAM system to a virtual coordinate system?
3. How do corresponding elements of both domains appear from a user's perspective when observed through an optical see-through display, and can these relations be captured and exploited as part of a calibration procedure for these devices?

In the following subsections, we will outline how the present thesis provides answers to these questions.

1.2. On the Need of User Involvement for Spatial Registration

Estimating the spatial registration parameters requires first of all establishing links between the real world and the virtual content, which for example in industrial scenarios may consist of pre-existing CAD models. In the spirit of computer science there is a natural desire to solve this data association problem in an automated fashion. A possible approach consists in using model-based recognition and tracking approaches [LF05] and to directly detect virtual objects and their pose in the image. However, in practical situations the automatic registration may be difficult or inappropriate as the following examples illustrate (see Fig 1.2):

- *Unavailability of appropriate models for registration*: In many AR applications, the virtual content may be almost entirely different to the real environment. In fact, this situation is not uncommon for AR, since the objective is to provide information different to what the user can observe without it. For example, in cultural heritage [KZB*11, VIK*02] or architecture, the virtual model represents some state in the past or in the future. Similarly, in industrial plant inspection [GSB*07], substantial site changes after construction are made that are not fed back into the CAD system as an update would be too expensive. In these cases, it may require the knowledge of experts (geodetics, archaeologists etc.) to establish correct correspondences between the real and the virtual world.
- *Strong accuracy requirements*: Even if the differences between the virtual model and the real scene are small, an automated registration may be inappropriate or undesirable due to accuracy constraints for the application. For example, in industrial quality inspection the reference object may consist of many assembled subparts, and the objective is to let the user identify whether the deviations of the mounted subparts are within their required tolerances [NK06, Nöl06, PBDM07]. Any registration method must necessarily be independent of possible errors in the construction. Thus, not the whole virtual model can be used but only

(a) (b) (c)

Figure 1.2.: Challenging examples where the automatic registration of virtual to real word is difficult or inappropriate. (a) The geometry of the virtual model does not match the corresponding real word object. (b) Registration shall only be based on a sparse set of highly accurate reference points instead of the whole model. (c) A highly textured real object may contradict the typical assumption of model-based trackers that edges in the 3D model also result in dominant intensity gradients in the image.

certain sub-elements. These can consist of some discrete reference points (e.g. drill holes), productions frames, or reference plates, which may be difficult to detect or are hardly visible due to self-occlusion, and thus may not provide dense enough correspondences for a robust, automated registration including outlier handling.

- *Violation of registration method assumptions*: Appearance-based methods using image features [GHT11] can be employed whenever a fully textured virtual model is available, but this is hardly the case in practical applications that go beyond simple poster or magazine tracking. Among those that purely rely on the geometry of objects, one can consider approaches based on the classical RAPiD approach [HS90] by Harris [DC02, WWS07], on Champfer matching [LTVC10, CC12, AAL*09], or on some mixed metric [LPT13]. Common to these methods is the assumptions that model edges (silhouettes or crease edges) are also observable as brightness jumps in the image. This assumption may be violated for soiled or highly textured objects, leading to a deception of the algorithm and a failure of the registration.

Due to these difficulties, there is a need for tools and algorithms that support a user or expert in contributing with prior knowledge to solving the registration problem while maintaining in control over the quality of the output. Having the user in the loop, also allows to capture implicit scene knowledge, which otherwise is often hard to formalize.

In this thesis, we propose two variants for the user input:

- *Post-editing reconstructed real-word models in a preparative phase*: When using natural features for tracking, the user may reconstruct the scene to obtain a real-world model, first. Then, with an appropriate interface, the user can browse through the recorded image sequence, select some of the reconstructed points,

and associate them to points, lines, or planes in the virtual model. This preparative stage may not only be used for spatial registration, but also for other optimizations, such as the improvement of the feature recognition rate and the handling of feature visibility with appropriate learning methods. The final output is a registered tracking model that can be embedded into a ready-to-use AR application. This workflow is particularly useful for static scenes that do not alter much over time and when the application is intended to be used many times afterwards.

- *Placement of markers with associated partial registration information*: In a second variant, the user places various markers in the scene, with some reference markers having predefined associations to the virtual world. During runtime the location and orientation of all visible markers is reconstructed as part of a real-world model within the SLAM. As soon as enough reference markers have been observed, registration is automatically preformed and the tracking continues in the coordinate system of the virtual world.

In order to keep the overall level of user involvement small and to enable a general applicability, we allow that the user-provided information is sparse, locally incomplete and may be distributed over the scene. For example, in the case of feature map editing, the user is only required to provide a small amount of correspondences (three at minimum), and it is not necessary for two or more of the supplied scene points to be simultaneously visible in one of the recorded images. As for markers, it is often argued against their usage in literature, because they presumably involve a high effort for instrumenting the scene. The objections are typically based on the idea that the reference markers need to be placed very accurately or their location and orientation has to be measured after placement with regard to all six DoF with external measurement devices [Nöl06]. In our case, markers are not used as fully self-contained tracking targets, but as a more abstract medium to communicate spatial registration data to the visual system. The user may place them freely on surfaces or along edges without having to completely specify their exact position and orientation in virtual coordinates. In this case, observing a single marker may not be sufficient for spatial registration, rather, the required information is then distributed in the layout of several markers together. The advantage of this possibility is that the placement, removal, and reattachment can be done with little effort without necessarily compromising the accuracy and reproducibility of the registration result.

1.3. Registration as a Minimization Problem

Apart from the correspondence problem, i.e. how the links between the real and the virtual world are established, the other question concerns the actual numerical computation of the registration parameters. Many problems in registration or computer vision in general, can be formulated as nonlinear least-squares minimization programs. Algorithms for computing solutions can be coarsely divided into recursive (Bayesian) estimation approaches, iterative (Newton-type) methods, and closed-form solvers. The first two usually are local minimizers which typically require a good initial guess in order to converge to the correct solution. Closed-form solvers may determine the global solution directly, but depending on the size and the complexity of the problem, appropriate algorithms are hard to find, too inefficient, or may not even exist. Therefore, for the solution of practical problems usually a mixture of closed-form and locally convergent minimizers are used. In the scope of this thesis both, closed-form and iterative algorithms, are covered and their mutual interplay for spatial registration is outlined.

One fundamental numerical estimation problem in tracking, SLAM, and computer vision in general is the perspective-n-point (PnP) problem. Here, the task is to estimate the orientation and translation of the camera in the environment based on 2D image points and associated 3D points of the real world model. Despite a long history of research, it was only recently that efficient, globally optimal, closed-form solvers have been proposed, which can handle arbitrary numbers of correspondences including minimal configurations as well as over-constrained cases with linear complexity. The difficulty of this problem is rooted in its nonlinearity which stems from two sources: the perspective division and the rotation. The first part, i.e. the nonlinear mapping

between 3D points and corresponding points in the image plane, can be overcome by minimizing the error in object space rather than in image space. As for the second part, rotations are often parametrized by quaternions. This makes it possible to express the solutions as the roots of a multivariate system of polynomial equations, for which methods from algebraic geometry such as the Gröbner basis technique can be used.

From a general perspective, the PnP problem deals with the estimation of a Euclidean transformation that connects two coordinate systems: the coordinate system of the reconstructed real world model and the coordinate system of the dynamically moving camera. Similarly, spatial registration also requires to estimate a Euclidean transformation or possibly a similarity transformation including a scaling; in this case between the real world model and the virtual model. It is therefore natural to ask, whether the same or similar closed-form algorithms can be reused for this purpose. In this work, we are particularly interested in the case, when the input for the spatial registration algorithm is heterogeneous and indeterminate to some extent, i.e. when it not only contains correspondences between full 3D points in either domain, but when the target location of a point is unknown in certain directions, e.g. when it must be as close as possible to only a line or a plane in 3D. An algorithm that can operate on such data would enable the envisioned registration method, where a user can place markers arbitrarily along edges or surfaces for spatial registration.

Simultaneous localization and mapping (SLAM) aims at enabling the tracking in unknown environments. PnP algorithms are an integral part in most SLAM solutions, but they assume that a 3D model of the environment is given. The realization of a SLAM system requires additional algorithms for reconstruction, e.g. by means of triangulation of scene points which are observed from various viewpoints. Since the accuracies of the estimated motion and the reconstructed 3D model mutually depend on each other, the recovery of both, structure and motion, is typically tightly coupled. There exist two basic strategies: recursive, filtering-based and global, iterative minimization-based approaches. The former is based on probabilistic (Bayesian) estimation. Measurements are sequentially processed to update a large state vector containing the unknowns. They are discarded after being processed, and the history of past observations is condensed into probability distributions of the variables. The latter repeatedly uses sparse optimization routines (bundle adjustment) to simultaneously minimize all measurement errors including those of the past for global optimality. It is generally acknowledged that the second strategy results in more efficient implementations regarding both, accuracy and speed, provided that some level of sparsity is ensured (reduction to key-frames) [SMD10a]. Therefore, bundle adjustment (BA) has become a key-component in most state-of-the-art SLAM implementations [KM07, MAMT15, ESC14].

Even though bundle adjustment provides an optimal estimate of the reconstructed environment given the available measurement data, systematic or random measurements may still result in low-frequency deformations of the estimated real world model and the camera path. In this case, spatial registration based solely on a rigid reconciliation of coordinate systems (Euclidean or similarity transformations) may not be sufficient for AR applications with high accuracy constraints. We therefore ask to what extent the provided user knowledge can be integrated into the BA, in order to further compensate some of the remaining errors. Again, we are particularly interested in the case, when the provided information is sparse and incomplete, e.g. when translation or point parameters are constrained to a plane or a line, or rotations are bound to one particular axis.

1.4. Closing the Loop for Optical See-Through Display Calibration

Video see-through, which refers to overlaying the virtual content on a camera live feed using an extra screen, can be considered as the standard displaying technique for AR. However, it is ultimately undesirable for various reasons. For example, when using smartphones or tablets for AR the user must hold the device in his hands while aiming with the camera at the target, which essentially prevents him from pursuing his actual task. Similarly, in automotive AR-based driver assistance, using the console display for AR visualization [RTG*14] requires the

user to redirect his attention off the road to the display in order to experience the overlayed AR information [GFK14]. Visualizing the virtual content directly into the user's field-of-view can be achieved with optical see-through displays, such as OST-HMDs or head-up displays (HUD), but they require additional calibration efforts.

In the traditional approach for calibrating OST displays [TN00] the user performs a calibration each time the display is used by aligning some displayed virtual points with real points in the scene. It is based on a monolithic calibration model whose parameters are only valid for one particular user and only for the time his viewing perspective remains unchanged relative to the display.

A better approach - as originally proposed by Owen et al. [OZTX04] for HMDs and extended in our work for HUDs - consists of essentially replacing the human observer by a moving camera. During calibration a suitable known pattern is displayed, which can be regarded as a substitute for any possible virtual AR content. The movement of the camera emulates possible variations of the user's observation points. The principle consists of observing how the displayed virtual pattern actually appears in reality from various viewpoints. With such a setup it is possible to retrieve the calibration parameters automatically, and moreover, to obtain a calibration model, where the hardware-specific parameters are decoupled from user-related parameters. This simplifies the adaptation of the calibration model to user switches or dynamic changes of the viewing position.

Using such a camera-based calibration procedure requires that the motion of the camera is continuously and accurately tracked, so that each observation of the displayed calibration pattern can also be associated precisely to possible observation points of the user. Furthermore, it is necessary to retrieve the calibration parameters in a specific reference coordinate system. In our case this is given by the virtual model of the car, in which the HUD is installed [2]. Therefore camera tracking and spatial registration are important elements. Thus, such a camera based calibration procedure can be seen as an application example of the other techniques presented in this work.

1.5. Relevant AR Applications

There exist numerous applications for augmented reality ranging from education, entertainment, surgery, tourism, marketing and sales, architecture and many others [SH16, vKP10, Nav04]. Since each of them has different requirements for registration and tracking, there certainly does not exist an optimal, general purpose strategy to address all of them. For example in Magic Mirror retail apps, where artificial makeup is superimposed on real faces [JRMF16], SLAM is rather inappropriate, whereas specialized face recognition techniques together with an accurate 2D segmentation of lip, eyelid, and eyebrow outlines are of much higher importance. Other applications, such as the gaming app Pokémon GO! [SFS*17], do not have strong accuracy constraints - the augmentations just need to look plausible. In this case, GPS, magnetic compass, and inertial sensors are sufficient for tracking, so that the resource-intensive visual processing of the camera signal can be dispensed with.

Besides HUD calibration - which is presented as a separate contribution and application example - the techniques developed in this work particularly proved valuable for the following use-cases:

Cultural heritage – The House of Olbrich: [KZB*11]: One of the applications is a mobile history and architecture guide for the Art-Nouveau quarter in Darmstadt (see Fig. 1.3). One of its buildings - the House of Olbrich - was destroyed during World War II and only rudimentary restored afterwards. The app enables tourists to take a photo of the current building, send it to the server, and receive an augmented view with a blueprint based 3D model showing its original state from the same perspective. Context-sensitive navigation icons let the user explore additional background information.

[2] For the calibration of HMDs, a suitable reference frame is the coordinate system of a built-in camera, which will be used for motion tracking.

One of the challenges during its realization was that the pose had to be estimated from a single image alone with a low failure rate, despite the high level of appearance changes which are typically encountered in outdoor environments. To this end, multiple feature maps were reconstructed at varying lighting and weather conditions. With the techniques presented in this work, all of these real world models could be accurately registered to the virtual blueprint model, even though the current building only had little in common with the original building. Having all these appearance models in a common coherent coordinate system, it was possible to combine them all for recognition, which provided enough redundancy for robust pose estimation.

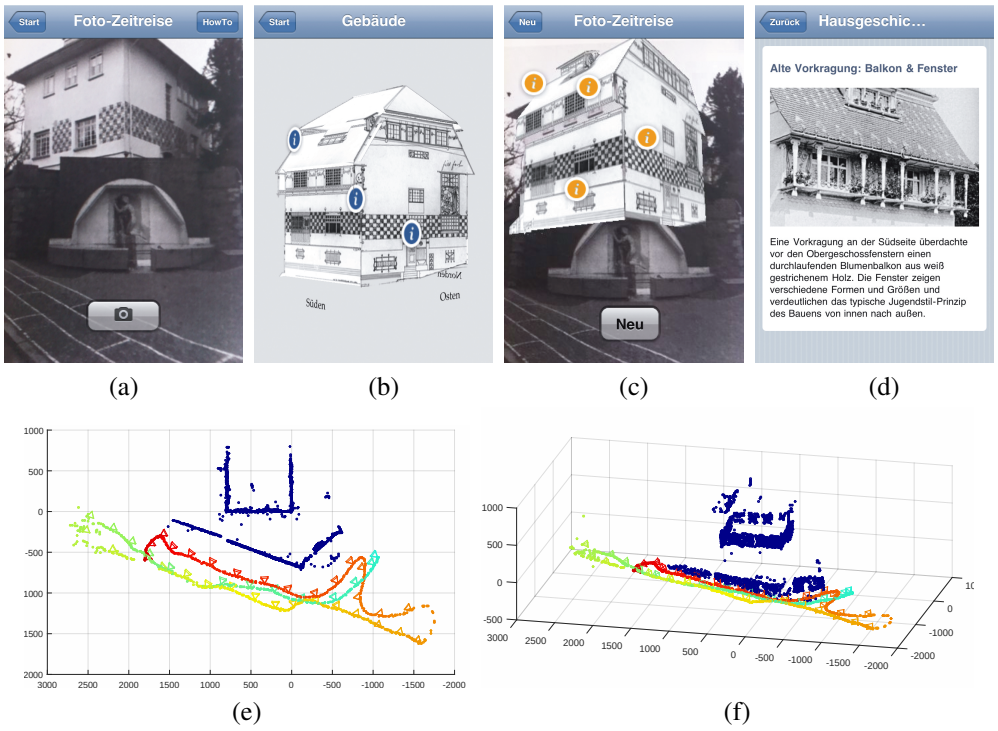
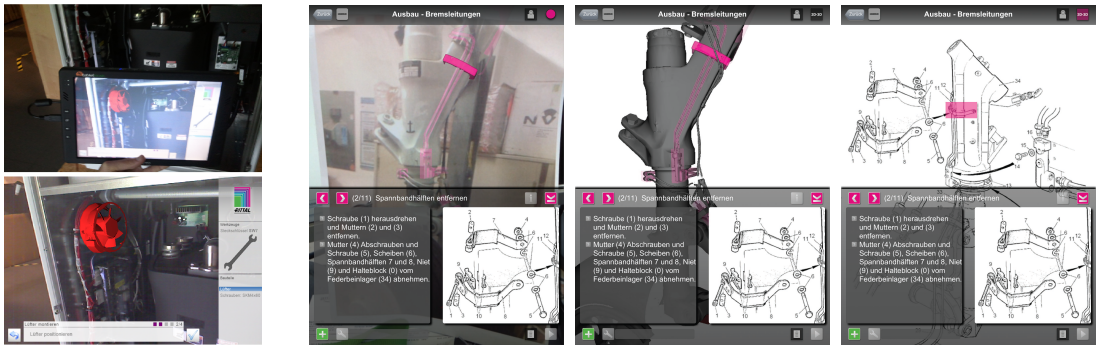


Figure 1.3.: The "House of Olbrich" app lets tourists experience a famous Art-Nouveau building in Darmstadt in its original state of before it was destroyed during World War II. On-site snapshots (a) are augmented with a blueprint-based model (b) on a server and sent back to the user (c). View-dependent information bubbles are linked to additional historical footage and textual information (d). On the server, the tourist's viewpoint is retrieved by matching the images against pre-reconstructed and pre-registered feature maps, as shown from top (e) and bird's-eye view (f).

Industrial visual inspection and quality control: In another application, augmented reality is used for quality control of assembled industrial objects. During inspection, the goal is to identify differences between the real object and its corresponding CAD model as planned on a mobile device in real-time. To this end, only the outlines of a CAD model are superimposed, e.g. only the vertices of the triangle mesh with appropriate occlusion culling.

For the realization it was required to ensure a high accuracy for tracking and registration: the misalignment in the overlay had to be less than one millimeter on the real object. In addition, the registration had to be carried out on the basis of a reference geometry, which encompassed only a small part of the real object. In order to meet

these requirements, tracking and registration was implemented with the help of a marker-based SLAM pipeline. For this purpose, the user was able to assign reference information in advance to a subset of the markers via a corresponding user interface. The exact placement and adjustment of the reference markers on drill holes, edges, or surfaces of the reference geometry was supported by physical adapters on which the markers were mounted. Reconstruction and registration of the marker ensemble took place either at runtime or within a further intermediate step, in which an SLR camera was used to capture the images for higher precision. The techniques presented in this thesis allow an optimal exploitation of the registration information provided by the user, in particular if they are only partially defined and distributed across the scene.



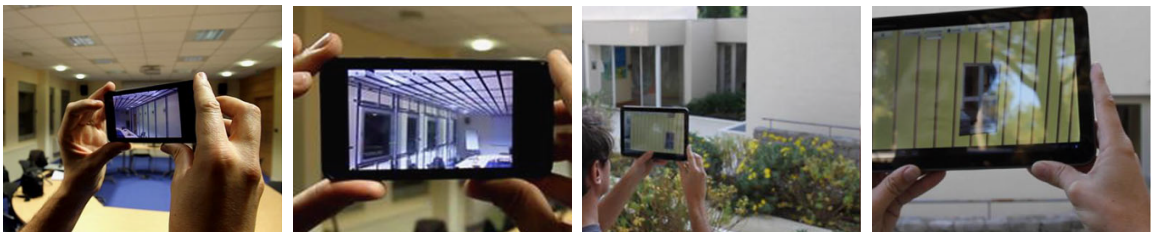
(a) (b)

Figure 1.4.: Augmented reality applications for maintenance and repair of industrial objects such as fuel cells (a) and landing gears (b).

AR maintenance [EKR*13, EKR*15]**:** A typical scenario is to use AR for maintenance and repair tasks (see Fig. 1.4). In fault analysis the user is guided through a sequence of checks which he has to perform on a non-functioning real object. The system is connected with a database of error symptoms and possible causes. Depending on the outcome of one particular test, the system may further narrow down the reason of the failure, and present a more specific test case on the next level, until the final source of the error is found. Subsequently the system will guide the user step-by-step through the process of repairing the machine including disassembly, aggregate exchange, and reassembly. During both, fault analysis and repair, AR offers the possibility to visualize the required procedural subtasks directly on the real object and in a context sensitive manner. In this way AR may offer substantial efficiency gains in maintenance, as the user does not have to browse through printed-out paper manuals in order to gather the required information.

The realization of maintenance applications typically involves the creation of a fairly large amount of 3D content, which is supposed to appear in the AR view. The techniques presented in this work, allow to decouple this content authoring process from the tracking. For setting up the latter, a model (feature map) of an exemplary real object is once reconstructed and accurately registered to the virtual model. This registered feature map can be reused many times on identical objects. Content authoring including complex 3D visuals and dynamic animations can be performed offline relative to the virtual CAD model. The fact, that the tracking model is also registered to it, will automatically ensure that this content appears correctly during runtime of the application.

AR-supported building information management [OGK*13, KOE*12]**:** Another use-case combines the AR paradigm with building information models (BIM) (see Fig. 1.5). Such a system may serve to support the



(a) (b) (c) (d)

Figure 1.5.: Mobile AR-supported building information management (BIM) system, providing an x-ray view on the interiors, such as the electric wiring, water supply, insulation or steel frames. It may support architects, technicians or facility managers to plan and document changes at the building. It can be used indoors (a/b) and outdoors (c/d).

planning and documentation of changes at the building, as for example the installation of a new ventilation system. By means of an x-ray view the user can inspect the internals inside the walls and ceilings. Critical installations, which must remain undamaged, such as electric wires or water pipes, are instantly revealed by the application. After completed work, the changes made can again be fed into the system for documentation.

Again, the prototype of this system has been realized by reconstructing feature maps during a preparative phase, which were registered to the BIM through manual editing. For indoor use, however, the re-usability of these tracking models remains limited, since natural features in indoor environments are usually sparse and often belong to movable objects such as furniture. Therefore, another viable alternative is to use markers for quick in-situ registration, and use natural features only temporarily for extendable SLAM. With the algorithms in this work, it would be sufficient to place three markers (with known scale) arbitrarily at three non-parallel surfaces of the building (e.g. two walls and the ceiling), thus ensuring a very simple setup procedure.

1.6. Contributions and Thesis Outline

The scope of the current thesis is the geometric registration for augmented reality. The general objective is to ensure that the displayed virtual content appears at the correct position and from the right perspective. Geometric registration notably includes tracking, spatial registration, and display calibration. Within this context, various algorithms are presented that allow the numerical estimation of the registration parameters. User knowledge is one important source of information, and we show how the presented algorithms can optimally exploit this data for accurate registration.

In Chapter 2, we present a closed-form solver to estimate the parameters of a single Euclidean or similarity transformation. The presented approach represents a further generalization of recent state-of-the-art perspective-n-point (PnP and GPnP) algorithms. It comprises several advantageous properties. First, it is universally applicable to arbitrary geometric configurations including the planar and the non-planar PnP case. Second, it can handle both, minimal and overconstrained cases, and only requires a linear effort with regard to the number of input correspondences. And third, it computes all relevant minima at once including the global solution. Our derivation is based on the idea that the PnP problem can be interpreted as the least-squares registration between a set of points and corresponding lines in 3D space. The generalization consists of extending the applicability to also cover correspondences between points and planes, points and other points, or any mixture of these three correspondence types. The algorithm is based on a decoupling of the linear parameters (translation and scale)

and the rotation using an orthogonal complement formulation. For the rotation parameters a system of multivariate polynomial equations is set up, and its solutions are obtained by means of the Gröbner basis method. Within comprehensive evaluations on synthetic data, we can show that our formulation is not only more general but also faster than previous approaches. The results of this chapter are based on our awarded ACCV conference publication [WK17] and its CVIU journal follow-up [WSFTK18]:

- WIENTAPPER F., SCHMITT M., FRAISSINET-TACHET M., KUIJPER A.: A universal, closed-form approach for absolute pose problems. *Computer Vision and Image Understanding (CVIU)* (2018) — [WSFTK18]
- WIENTAPPER F., KUIJPER A.: Unifying algebraic solvers for scaled Euclidean registration from point, line and plane constraints. In *Proc. Asian Conf. on Computer Vision (ACCV)* (2017), pp. 52–66, **Best Paper Honorable Mention Award** for regular papers at the ACCV conference. — [WK17]

Algorithms for the PnP problem are an important element for dynamic motion estimation inside the tracking or SLAM. They are used to continuously estimate how the camera used for tracking is oriented with regard to the surrounding environment or the real world model. Beyond that our presented algorithm is also particularly useful for spatial registration. Based on correspondences between the real world and the virtual model, an Euclidean or similarity transformation can be computed, which allows to transform the reconstructed real world model and its associated camera path to the coordinate system of the virtual model.

Chapter 3 focuses on the simultaneous, iterative refinement of a large number of motion and structure parameters based on measurements of an image sequence as part of a bundle adjustment. Our main contribution consists of embedding scene knowledge provided by the user as constraints into the BA. Instead of integrating these constraints by means of a standard Lagrangian formulation [TMHF00], we propose to interpret them as manifolds in parameter space, leading to a strictly feasible optimization-on-manifold approach for BA. Compared to a Lagrangian formulation it preserves the least-squares character of the minimization and also reduces the number of optimization parameters. Since the aim is to compensate low-frequency deformations of the reconstructed model and the camera path, we refer to this technique as a non-rigid registration. The chapter is dedicated to the differences of our proposed approach compared to ordinary BA, and it comprises three steps. First, the constraints and the parameters for BA must be transformed into a common coordinate system. Thus, it involves a rigid similarity transformation and represents a use-case for the algorithm of Chap. 2. Second, the constrained parameters must be projected onto their respective constraint manifolds to ensure the feasibility of the constraints right from the start. Third, during iterative minimization the feasibility is further maintained by forcing the parameters to evolve only along manifold geodesics. We exemplify these operations for various types of constraints including fully known parameters, line and plane-constrained points and translations, or axis-constrained rotations. We also discuss to what extent existing sparse minimization libraries can be used for the same purpose.

Next, we present two concrete application examples of these techniques in the subsections that follow. In Chapter 4, we consider a marker-based SLAM pipeline. In this case, the user provides his scene knowledge in advance by placing some reference markers at planar surfaces or along edges of the target environment and by associating partial registration information to these reference markers. During runtime, all observed markers are first reconstructed in a local coordinate system. Once enough reference markers are reconstructed, the rigid alignment, parameter projection, and manifold-constrained BA for non-rigid registration are executed automatically and the tracking continues in the desired coordinate frame of the virtual model.

Chapter 5 refers to the case, when a user manually establishes correspondences between a pre-reconstructed feature map and a virtual model with an appropriate interface. This user-provided information is exploited for rigid and subsequent non-rigid registration within our constrained BA framework. Moreover, we also demonstrate how this preparative stage additionally serves to improve feature recognition and tracking performance in order to setup a ready-to-use natural feature tracking-based application for AR.

For both use-cases we show that by internalizing the user information into the BA, a substantial improvement of registration accuracy is gained, as low-frequency deformations of the reconstructed real model and the camera path that occur due to systematic or random measurement errors are compensated to a large extent. We have participated with our system at various public tracking benchmarks for AR and we show the corresponding results at the end. Chapters 3-5 are mainly based on our IEEE-3DIMPVT conference publication [WWK11b], our Computers & Graphics journal publication [WWK11a] and in parts also on our CVIU journal publication [WSFTK18] (listed above):

- WIENTAPPER F., WUEST H., KUIJPER A.: Composing the feature map retrieval process for robust and ready-to-use monocular tracking. *Computers & Graphics 35*, 4 (2011), 778 – 788 — [WWK11a]
- WIENTAPPER F., WUEST H., KUIJPER A.: Reconstruction and accurate alignment of feature maps for augmented reality. In *IEEE Proc. of Int'l Conf. on 3D Imaging, Modeling, Processing, Visualization and Transmission (3DIMPVT)* (2011), pp. 140–147 — [WWK11b]

In Chapter 6, we consider the visualization side of geometric registration. We present a generic and cost-effective camera-based calibration for an automotive head-up display as one example of an optical see-through display for AR. Our contribution comprises two aspects. First, we present a model, which maps the setup consisting of the user (observer) and the display to pinhole camera parameters as needed for the rendering. These are naturally divided into user (head position) and hardware related parameters. The latter consists of the view-independent spatial geometry, i.e. the exact location, orientation and scaling of the virtual plane, and a view-dependent image warping transformation for correcting the distortions caused by the optics and the irregularly curved windshield. View-dependency is achieved by extending the classical polynomial distortion model for cameras and projectors to a generic five-variate mapping with the head position of the viewer as additional input. Our model enables the HUD to be used together with a head tracker to form a head-coupled display which ensures a perspectively correct rendering of any 3D object in vehicle coordinates from a large range of possible viewpoints. Second, we propose a procedure for the retrieval of the calibration parameters. The calibration involves the capturing of an image sequence from varying viewpoints, while displaying a known target pattern on the HUD. For the accurate registration of the camera path we use the techniques presented in Chapters 3 and 5, which is why HUD calibration can be regarded as another application example. In the resulting approach all necessary data is acquired directly from the images, so no external tracking equipment needs to be installed. The accuracy of our HUD calibration is evaluated quantitatively and qualitatively. Finally, the calibration method is also extended to OST-HMD calibration. The separation of user and hardware parameters allows for a quick user adaptation by means of a simple user interface. The results of this chapter are based on our ISMAR full paper [WWRF13] and the ISMAR short HMD demo paper [WEK*14]. Moreover, we also published a patent based on this work [GWW15].

- WIENTAPPER F., WUEST H., ROJTBERG P., FELLNER D.: A camera-based calibration for automotive augmented reality head-up-displays. In *IEEE Proc. Int'l Symp. on Mixed and Augmented Reality (ISMAR)* (Oct 2013), pp. 189–197, awarded by the Fraunhofer IGD and TU Darmstadt's Interactive Systems Group (GRIS) with the **Best Paper Award - Honorable Mentions** in the category *Impact on Business*. — [WWRF13]
- WIENTAPPER F., ENGELKE T., KEIL J., WUEST H., MENSIK J.: [demo] user friedly calibration and tracking for optical stereo see-through augmented reality. In *IEEE Proc. Int'l Symp. on Mixed and Augmented Reality (ISMAR)* (Sept 2014), pp. 385–386 — [WEK*14]
- GIEGERICH P., WIENTAPPER F., WUEST H.: Method and apparatus for controlling an image generating device of a head-up display. Patent. WO/2015/044280, 02 04, 2015 — [GWW15]

Finally, Chapter 7 concludes this thesis with a summary and a discussion of the achieved results.

2. Closed-Form Registration

2.1. Introduction

In this chapter, we are concerned with the non-iterative, numerical computation of a Euclidean (six DoF) or similarity transformation (seven DoF) that relates two coordinate systems. Within this work, our main motivation for considering this problem is the spatial registration of a SLAM output to the coordinate system of a given virtual model. In a vision-based-only monocular SLAM, the reconstruction of the environment and the camera path are defined up to a transformation which has seven deegrees-of-freedom (*gauge freedom*, [TMHF00]), including a rotation (tree DoF), a translation (tree DoF) and a scaling parameter (one DoF). A SLAM algorithm must internally select a coordinate system for computational reasons. But the concrete choice within these degrees of freedom is completely arbitrary, and therefore it has no relation to the virtual coordinate system. In AR the knowledge about the transformation that relates both domains is essential for the correct rendering of virtual content and for seamless integration into the user's real view.

For the considered problem of computing the spatial transformation, we assume that the input is given as a set of 3D correspondences between the SLAM model and the virtual domain. In this regard, we do not confine ourselves to 3D point-to-point correspondences only, as in the *absolute orientation problem* already solved around the 1990s [AHB87, Hor87, HHN88, Ume91], but look at the problem from a more general perspective. In particular we allow the correspondences to be indeterminate to some extend. That means, a 3D point in one coordinate system can be associated to infinitely many points comprising lines or planes in the other coordinate system, without knowing exactly in advance to which of these infinitely many points it actually belongs to. The considered registration problem from point-to-line and point-to-plane correspondences can thus be considered as an extension of the 3D point-to-point case. In later chapters, we will present applications for this generic problem, where we consider spatial registration based on reconstructed markers placed along edges or on surfaces of an object (Chap. 4) or based on manual assignments between reconstructed natural features to parts of a virtual model (Chap. 5).

Algorithms that solve this problem have a broad range of applications that go beyond the spatial alignment of SLAM models to a target coordinate system. In particular, it is important to note, that there is a close connection to *absolute pose problems*. These notably refer to camera pose estimation from image measurements (usually six DoF) and include the classical perspective-n-point problem (PnP), the perspective-n-line problem (PnL) and their variants for generalized cameras (GPnP / GPnL). Camera pose estimation also aims to calculate the spatial transformation parameters between two coordinate systems, in this case between the coordinate system of the tracking model and the local coordinate system of the dynamically moving camera. Absolute pose algorithms represent a central internal component of any visual tracking or SLAM solution and therefore have attracted attention for many decades. But despite this long history of research and a vast amount of proposed solutions, it was only recently that absolute pose algorithms emerged that comprise all of the following desirable properties [HR11, ZKS*13, KLS14, SFHT14, MR11]:

- *Universal applicability*: The algorithms should be applicable to minimal configurations as well as over-constrained cases. Different geometric configurations should be handled equally and coherently, so the

algorithms do not switch between different strategies for solving special cases, such as the planar configuration in the PnP case.

- *Global optimality and completeness of the solutions*: They should be capable of finding the global optimum with respect to a well-defined optimality criterion without the need of initialization. Moreover, in case of minimal or other ambiguous configurations [SP06, CB14], they should return all solutions including local minima, which is neccessary for achieving disambiguation on a higher level within the application.
- *Efficiency and linear complexity*: In view of the fact that some applications require solutions based on hundreds or even thousands of correspondences in real-time, it is necessary that the algorithms can efficiently determine the solutions with a runtime of linear complexity.

The DLS algorithm presented by Hesch and Roumeliotis [HR11] in 2011 for the central PnP problem is the first that encompasses all these characteristics simultaneously. This is achieved by (1) minimizing the *object space error* [LHM00, ENvG07] instead of the image space error, which circumvents the nonlinearities induced by the perspective division, (2) compressing the original set of equations in linear time to a compact, fixed size rotation-only problem (elimination of depth and translation parameters), and (3) computing the rotation as solutions to a system of multivariate polynomial equations with an algebraic geometry technique (Macaulay resultant). The same strategy was also adopted later for the generalized PnP (GPnP) problem by Kneip et al. [KLS14] with fixed scale and by Sweeney et al. [SFHT14] with a global scale as additional variable. Kneip et al. also presented an alternative solver for the rotation, which was derived with the Gröbner basis technique.

As we will discuss in this work, when formulating the PnP problem in terms of the *object space error* metric [LHM00, ENvG07], 2D points in the image are interpreted as lines in 3D space, which pass through the (homogenized) image point and the camera center. The minimized error corresponds to the orthogonal distance between the transformed 3D point and its corresponding line. In this context the PnP and GPnP are point-to-line registration problems that differ only on their configuration, i.e. whether the lines have a common intersection point or not. There is a similar analogy for the perspective-n-line (PnL) problem. Just as 2D image points can be represented by 3D lines, 2D lines in the image can be interpreted as planes in 3D space passing through the camera center [MR11]. Thus, the PnL-problem can be solved by registration of end-points of the 3D reference lines (or any other two or more points) to these planes.

The results of this chapter represent a further generalization of these algorithms. Based on the intuitions above, we analyze the general problem of finding optimal Euclidean or similarity transformations between a set of points and another set of corresponding points, lines or planes in 3D space for arbitrary configurations. To this end, we propose to represent the geometric primitives by means of their *orthogonal complements*, which results in a linear parameter elimination scheme, which is not only more general but also faster than previous formulations. Since the resulting rotation-only problem after linear parameter elimination is equivalent to to previous formulations in the PnP and GPnP case [HR11, KLS14, SFHT14], we consider applying the internal rotation solvers of these algorithms on the other geometric settings, as well. It turns out that they can be reused interchangeably for point-to-line and point-to-plane registration, but not for the exclusive point-to-point case. For this, we propose own rotation solvers, which can handle almost all settings considered in this chapter (see Fig. 2.1).

To summarize, this chapter discusses a general approach to calculate scaled or unscaled Euclidean coordinate transformations. It can process a wide range of input data, it is independent of special geometric configurations (e.g. the planar PnP problem from co-planar 3D points), and is applicable on both, minimal and overconstrained problems. It can be used for spatial registration between two 3D models as well as for camera pose computation from image measurements.

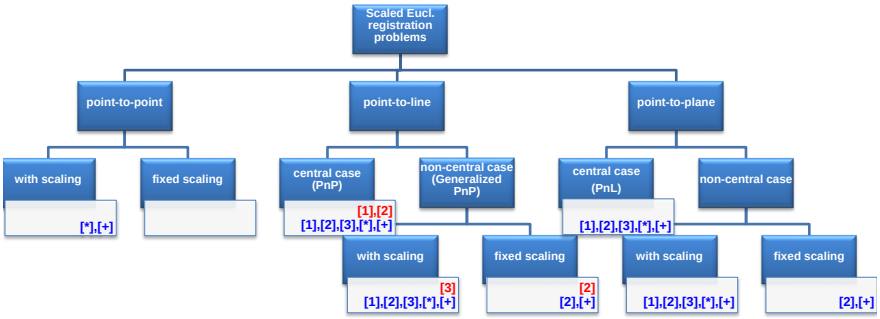


Figure 2.1.: Applicability of existing rotation solvers [1] [HR11],[2] [KLS14],[3] [SFHT14] to various registration problems as originally proposed (marked in red) and when using our orthogonal complement formulation (marked in blue). We also propose an additional rotation solver for the central cases with the lines and planes having a common intersection point and for the registration with variable scale (marked with an asterisk [*] [WK17]) as well as a rotation solver which is applicable to almost all cases [+].)

2.2. Related Work

Due to the high relevance for computer vision, most of the relevant work addresses camera pose estimation from 2D image information. In rare cases, other modalities are also considered, such as the measurements of a tactile robotic arm or line scans of a laser projector system [Che90, OKO09]. We will therefore focus on the area of *absolute pose estimation*, which commonly refers to the determination of Euclidean or similarity transformation parameters (pose) using image measurements with known associations to 3D reference data in a given coordinate system. By contrast, *relative pose estimation* is purely based on 2D measurements, notably between the views of two or more cameras [Nis04, LHhK08, KL14, Har97b, EE11]. As for the latter, the choice of the coordinate system is subject to some arbitrariness (gauge freedom) [TMHF00], so one deliberately selects the coordinate system of one of the cameras as reference, therefore the label *'relative'*. In this chapter, we are concerned with the former, and absolute pose algorithms can be categorized according to the following criteria:

- *Type of correspondence:* Camera pose estimation from 2D image point measurements and corresponding 3D points is termed *perspective-n-point problem* (PnP), whereas the pose estimation from 2D line segments in the image and corresponding 3D model lines is referred to as the *perspective-n-line problem* (PnL). Within the generalized perspective of this work, we also consider pose estimation between two sets of 3D points known as the *absolute orientation problem* [AHB87, Hor87, HHN88, Ume91, Lou16].
- *Central versus non-central:* Pose estimation from a single image can often be modeled with the pinhole camera model with a single center of projection (central PnP or PnL). Application examples of non-central or generalized camera models [SRT*11, Pon09] include pose estimation for calibrated multi-camera systems [Ple03, PFC*17], registration of SLAM or SfM trajectories with regard to pre-reconstructed maps [CSP16, SFHT14], wide-angle catadioptric cameras [ATR10], or cameras observing the scene through a planar refractive medium as in underwater imagery, resulting in infinitely many projection centers along a line (axial cameras) [ARTC12]. Pose estimation from an arbitrary configuration of measurement rays is referred to as the generalized PnP / PnL (GPnP / GPnL) problem.
- *Minimal versus overconstrained:* For solving the six degrees-of-freedom of the pose at least three 2D image measurements are necessary - each of them imposing two constraints (P3P and P3L). Efficient

minimal solvers play an important role inside a RANSAC-loop [FB81] for handling outlier measurements. However, it is desirable to have algorithms that can also work on a larger set of correspondences in order to suppress noise by optimally exploiting the redundancy.

- *Iterative versus closed-form:* Iterative approaches aim at successively reducing the error in each step, whereas closed-form solvers compute the solutions in a single pass - albeit possibly using higher level operations (e.g. SVD, Eigenvalues, polynomial root-finding) that internally may require iterative techniques [MP13].

Minimal solvers have been proposed for the central P3P [FB81, LHD88, HLON91, GHTC03], the generalized (non-central) P3P [CC04, NS07, HLLPF16, MA15], the central (perspective) P3L [DRLR89] and the non-central P3L problem [Che90, NF93]. In all of these cases, the problem is reduced to an eighth-order univariate polynomial from which up to eight solutions are obtained. In the P3P / GP3P case the derivation is typically based on the requirement that the distances between any two points must be preserved after rigid transformation, resulting in a polynomial parameterized with the unknown depth of one of the image points. The fastest and most stable P3P algorithm to date is proposed by Kneip et al. [KSS11], where solutions for the camera pose are calculated directly without the need of the intermediate point depths. In another work by Kneip et al. [KFS13], a minimal solver for the GP3P problem can be found, which uses the Cayley parameterization for rotations and the Gröbner basis technique. In the P3L / GP3L case, usually the coplanarity constraint between the transformed model line and a plane containing the image line segment is imposed. Using an intermediate coordinate frame the polynomial is formulated as a function of one of the Euler angles of the rotation. Ramalingam et al. [RBS11] also proposed a minimal solver for a mix of three point or line correspondences (hybrid P3P / P3L).

For generalized cameras the scale becomes important. For example, for pose computation with stereo rigs, the displacement of the camera centers must be defined in the same unit system as the 3D reference points. Otherwise, incorrect results are obtained or there even may not exist a feasible solution. Conversely, it is also possible to estimate a scaling parameter (7 DoF) if another fourth image measurement is added. Ventura et al. [VARS14b] proposed a minimal solver for the generalized pose and scale problem (GP3P with scale). They set up a linear system of equations and determine the optimal weights for the null space singular vectors by reimposing orthogonality on the rotation matrix by means of ten quadratic equations in 21 monomials. Again, up to eight solutions are obtained with a Gröbner basis solver. Recently, Kukelova et al. [KHF16] presented a general solution for the intersection of three quadrics in three variables. Its application to the generalized pose and scale problem resulted in a very efficient minimal algorithm that also covered the failure cases of Ventura's approach (planar scene configuration). Camposeco et al. [CSP16] proposed a minimal solver from two image rays and one 3D point (obtained by triangulation).

For the *overconstrained case* there exist many closed-form algorithms that determine a solution by using almost exclusively tools from linear algebra, in particular the singular value decomposition (SVD). The earliest variant - the DLT-algorithm [Sut74, Gan84, HZ04] - initially computes a linear solution (null space) to a set of equations, assuming that the nine entries of the rotation matrix are independent. This is followed by a subsequent re-orthogonalization of the rotation. Several variants of DLT for the central PnL problem have been recently published by Přilbyl et al. [PZv17]. Although these algorithms can determine a solution in linear time if properly implemented, they have limitations concerning the accuracy. More importantly, for less than six points, five lines or planar scene configurations the null space is spanned by more than one singular vector, rendering these algorithms inapplicable to such settings. The 'lifting technique' formalized by Ansar and Daniilidis [AD03] consists of reimposing nonlinear constraints to a general set of null-space vectors. The extended vector of monomial terms is again solved via SVD, which results in an algorithm having octic complexity with regard to the number of correspondences. Other approaches require a cubic [QL99] or quadratic [Fio01] effort. The idea of the EPnP approach [LMNF09] consists of expressing the equations in terms of four control points, for which reimposing nonlinear constraints is a constant operation and the overall effort becomes linear. The EPnP algorithm requires

special treatment for planar scene geometries, is known to be inaccurate for small numbers of correspondences and quasi-planar situations, and minimizes an artificial, non-intuitive algebraic error. The approach by Kneip et al. [KFS13] for generalized cameras (GPnP) is a straight-forward extension of EPnP and thus also inherits most of its drawbacks. Common to all these approaches is, that they use the SVD to estimate the solution space of nonlinear parameters (in particular the rotation). They generally fail whenever the solution is not unique as for minimal or other ambiguous configurations, because the nonlinearity is not modeled appropriately. The GPnP algorithm by Ess et al. [ENvG07] also falls into that category.

A number of iterative approaches have also been proposed that search for the global optimum. Some of them rely on bootstrapping the solution with a scaled orthographic projection approximation [DD95, LHM00]. This may be too coarse whenever the scene depth is very heterogeneous or when using wide-angle cameras, so there is a high risk that a false local minimum will be returned. Others seek for the global minimum by means of semi definite programming and convex relaxations [SP08, KH07, HK10, OKO06, OKO09], but despite linear complexity the overall computational burden remains fairly high. In general, all iterative solvers have in common that they only search for a *single solution*. If multiple global optima are present, such as for minimal configurations (three correspondences), or if a local minimum exists with a similar error level, the returned global minimum may nonetheless be the wrong one due to presence of measurement noise. Algorithms that are capable of determining all global and local minima at once offer the possibility of resolving this deception on a higher level within the application.

A closed-form solver (called RPnP), which is applicable in a general central PnP setting (including minimal, ambiguous and overconstrained cases), was proposed by Li et al. [LXX12]. From all K available correspondences they select a pair and form triplets with all remaining $K-2$ correspondences. From each triplet an eighth-order polynomial is derived (as in the minimal P3P algorithms), and they minimize the squared sum of all these polynomials (all expressed in one variable). First order optimality results in a 15-th order polynomial whose roots yield up to eight minima. Since the formulation is centered around the initially chosen correspondence pair, their solution is also particularly dependent on the noise of these two measurements. Xu et al. [XZCK17] is an variant of RPnP for lines (PnL). In a similar way, they consider triplets of line correspondences. And again, in order to obtain a mono-variate polynomial, each triplet contains the same two base lines, which again introduces a noise bias with regard to these measurements.

Recently, several linear complexity closed-from algorithms have been proposed, which can determine all relevant minima with regard to an unbiased, least-squares optimality criterion (object space error) for both, minimal or ambiguous and overconstrained cases. With this, they overcome all of the limitations of the earlier approaches. Among them are in particular to mention: the DLS algorithm for central PnP [HR11], UPnP for generalized PnP (GPnP) with fixed scale [KLS14], gDLS/gDLS+++ [SFHT14, SFHT16] for generalized PnP (GPnP) with variable scale and the algorithm of Mirzaei and Roumeliotis [MR11] for PnL. However, although they represent a generalization in their domain, they are nonetheless limited to their specific problem instance and can only be applied to their appropriate correspondence type.

In this chapter, we further unify these recent approaches. To this end, we take a generalized view and reinterpret them as 3D point-to-line, point-to-plane and point-to-point registration problems. We follow their general strategy by eliminating the linear parameters first, which results in a least squares error function that only depends on the nonlinear rotation parameters and a small symmetric coefficient matrix of fixed size. Then, in a second step, the rotation is solved with algorithms that are derived using methods from algebraic geometry such as the Gröbner basis method [CLO07]. Our unified formulation is based on a representation with orthogonal complements, which allows to combine different types of constraints elegantly in one single framework. We show that with our unified formulation the existing polynomial solvers of the former approaches can be interchangeably applied to problem instances other than those they were originally proposed for. It becomes possible to compare them on various registrations problems with respect to accuracy, numerical stability, and

computational speed. Our variable elimination procedure not only preserves linear complexity, it is even faster than previous formulations. For the second step we also derive an own algebraic equation solver, which can additionally handle the registration from 3D point-to-point correspondences, where other rotation solvers fail.

2.3. Unified Mathematical Framework for Rigid Registration Problems

In this section, we will derive our approach for calculating a Euclidean or similarity transformation from a mixture of 3D point-to-point, point-to-line, and point-to-plane correspondences. This is a unification of existing absolute pose algorithms from image measurements because of the analogy between PnP / GPnP and 3D point-to-line registration and between PnL / GPnL and 3D point-to-plane registration. We will discuss the algorithmic differences to existing approaches, which make it possible to use different types of correspondence within a single algorithm.

2.3.1. Objective Function and Vector-Matrix Representation

Suppose we have a set of K points, $\mathbf{x}_k \in \mathbb{R}^3$. Our goal is to find a transformation consisting of a rotation, $\mathbf{R} \in SO(3)$, a translation, $\mathbf{t} \in \mathbb{R}^3$, and an (inverse) scaling $s^{-1} \in \mathbb{R}_+$, so that the transformed points,

$$\mathbf{x}_k^{'} = s^{-1}(\mathbf{R}\mathbf{x}_k + \mathbf{t}), \tag{2.1}$$

are as close as possible to their corresponding geometric entities, which may either be a plane, π_k, a line, l_k, or another point, p_k. For point correspondences we measure the Euclidean distance between the transformed point $\mathbf{x}_k^{'}$ and the reference point p_k. For lines and planes we use the orthogonal distance, i.e. the length of the shortest vector that connects the transformed point with some other point on the line or the plane. We completely describe a geometric entity with an *offset point*, $\mathbf{y}_k$, and an *orthogonal complement matrix*, $\mathbf{N}_k$, whose columns are orthonormal vectors which are perpendicular to the affine subspace of the geometric entity. Then, the squared error can be written for all correspondence types in a uniform way

$$d_k^2 = \|\mathbf{N}_k^\mathsf{T}(\mathbf{x}_k^{'} - \mathbf{y}_k)\|_2^2. \tag{2.2}$$

The registration problem can then be formulated as the least-squares minimization of the total error for all correspondences,

$$\underset{\mathbf{R}\in SO(3),\mathbf{t}\in\mathbb{R}^3,s\in\mathbb{R}_+}{\arg\min} \sum_{k=1}^{K} \|\mathbf{N}_k^\mathsf{T}(\mathbf{R}\mathbf{x}_k + \mathbf{t} - s\mathbf{y}_k)\|_2^2. \tag{2.3}$$

By multiplying the error function with the squared scale, i.e. $s^2\sum d_k^2$, we have decoupled it from the other unknowns, and it now scales the offset points, $\mathbf{y}_k$. This only affects the absolute value of the error, but not the location of the minima. By stacking the rows of the rotation matrix, the scaling and the translation vector into a parameter vector, $\mathbf{s}^\mathsf{T} = [r_{11}, r_{12}, \ldots, r_{33}, s, \mathbf{t}^\mathsf{T}]$, Equation 2.3 can be re-factored and expressed in matrix-vector form,

$$\underset{\mathbf{R},\mathbf{t},s}{\arg\min} \|\mathbf{A}\mathbf{s}\|_2^2. \tag{2.4}$$

The corresponding rows, $\mathbf{A}_k$, belonging to the correspondence k can be written compactly using the kronecker product ('$\otimes$'):

$$\mathbf{A}_k \in \mathbb{R}^{n_k\times 13} = \left[\mathbf{N}_k^\mathsf{T} \otimes \mathbf{x}_k^\mathsf{T},\ -\mathbf{N}_k^\mathsf{T}\mathbf{y}_k,\ \mathbf{N}_k^\mathsf{T}\right], \qquad n_k \in \{1,2,3\}. \tag{2.5}$$

Each row of $\mathbf{A}$ corresponds to an equation in a (possibly overconstrained) homogeneous system of equations representing the least-squares problem of Eq. 2.4. Intuitively, the different types of correspondences should also impose different numbers of effective constraints on the registration problem. In our formulation, this is achieved naturally by the size of the orthogonal complement matrix. For *point-to-plane correspondences*, $\mathbf{N}_k^{(\pi)} \in \mathbb{R}^{3\times 1}$ is given by the normal vector of the plane which results in one equation per correspondence. *Points* can be interpreted as entities that span a zero-dimensional subspace, so any vector in $\mathbb{R}^3$ belongs to the orthogonal complement of a point. Thus, we are free to choose an arbitrary orthogonal matrix for $\mathbf{N}_k^{(p)} \in \mathbb{R}^{3\times 3}$ yielding three equations per correspondence. In particular, the identity matrix, $\mathbf{I}^{3\times 3}$, is a convenient choice. In case of *Point-to-line correspondences* often there is no orthogonal complement matrix at hand, but a bearing vector, $\mathbf{v}_k$, instead. For example, in the PnP problem and its generalized variant, the bearing vectors connect the homogenized image points on the normalized image plane with the camera centers, $\mathbf{y}_k$.[1] One can easily obtain a matrix $\mathbf{N}_k^{(l)} \in \mathbb{R}^{3\times 2}$ by means of an orthogonalization method like the Gram-Schmidt algorithm or a QR decomposition of $\mathbf{v}_k$. Alternatively, it is also possible to construct 3×3 matrices $\tilde{\mathbf{N}}_k^{(l)}$ with rank two directly from the bearing vectors after normalization ($\mathbf{v}_k^\mathsf{T}\mathbf{v}_k = 1$). Two possible options are:

$$\tilde{\mathbf{N}}_k^{(l)} \in \mathbb{R}^{3\times 3} = \begin{cases} [\mathbf{v}_k]_\times & \text{(cross product form)} \\ \mathbf{I} - \mathbf{v}_k\mathbf{v}_k^\mathsf{T} & \text{(annihilator form)} \end{cases}, \tag{2.6}$$

where $[\mathbf{v}_k]_\times$ is the skew-symmetric cross product matrix of $\mathbf{v}_k$. We note, that $\tilde{\mathbf{N}}_\mathbf{k}\tilde{\mathbf{N}}_\mathbf{k}^\mathsf{T} = \mathbf{N}_k\mathbf{N}_k^\mathsf{T}$, so the squared error, $d_k^2 = (\mathbf{x}_k^{'} - \mathbf{y}_k)^\mathsf{T}\mathbf{N}_k\mathbf{N}_k^\mathsf{T}(\mathbf{x}_k^{'} - \mathbf{y}_k)$, of the objective function (Eq. 2.2) will remain unchanged regardless of which matrix is used to represent the orthogonal complement.

Using the *cross product form* results in a similar system of equations than in the well-known DLT Algorithm [HZ04, Sut74, Gan84]. Since both variants of $\tilde{\mathbf{N}}_k^{(l)}$ only have rank two, there are also only two rows of $\mathbf{A}_k$ which are linearly independent. A common mistake is, however, to simply leave one of them out - as often proposed (e.g. [HZ04]) in order to reduce the number of equations. This would result in an anisotropic weighting of the measurements and to biased solutions[2], and therefore one should refrain from doing so.

2.3.2. Thin-SVD-Based Linear Parameter Elimination

Based on the matrix-vector notation of the objective function, the linear parameters, s and $\mathbf{t}$, can now be eliminated by representing them in terms of the nonlinear parameters, $\mathbf{r}^\mathsf{T} = [r_{11}, r_{12}, \ldots, r_{33}]$, using the pseudo-inverse and by back-substituting the resulting expression. Specifically, taking the derivative of Eq. 2.4 with respect to s and $\mathbf{t}$ and setting it zero yields the first order optimality conditions for $[s, \mathbf{t}^\mathsf{T}]$

$$\mathbf{A}_{st}^\mathsf{T}\mathbf{A}_r\mathbf{r} + \mathbf{A}_{st}^\mathsf{T}\mathbf{A}_{st}\begin{bmatrix} s \\ \mathbf{t} \end{bmatrix} = \mathbf{0}. \tag{2.7}$$

Here, we have partitioned $\mathbf{A} = [\mathbf{A}_r\ \mathbf{A}_{st}]$ column-wise into the submatrices $\mathbf{A}_r$ and $\mathbf{A}_{st}$ which belong to the nonlinear rotation parameters, $\mathbf{r}$, and the linear parameters, s and $\mathbf{t}$, respectively. Hence, we can express $[s, \mathbf{t}^\mathsf{T}]$ as

[1] In the central PnP case (pinhole camera) the image rays $\mathbf{v}_k$ are obtained by pre-multiplying the image points in pixel coordinates with the inverse calibration matrix, $\mathbf{K}^{-1}$, containing the focal lengths and the principal point. Since all image rays are supposed to intersect in the camera origin (camera center), typically $\mathbf{y}_k = \mathbf{0}$ is assumed for all measurements. For non-central imaging systems, e.g. stereo rigs or SfM-bundles, the $\mathbf{y}_k$ are the result of a calibration or reconstruction and represent the displacement vectors of the optical centers w.r.t. the predefined common coordinate system.

[2] $\tilde{\mathbf{N}}_\mathbf{k}\tilde{\mathbf{N}}_\mathbf{k}^\mathsf{T}$ or $\mathbf{N}_k\mathbf{N}_k^\mathsf{T}$ can be interpreted as a weighting (inverse covariance) matrix for the measurement errors, $(\mathbf{x}_k^{'} - \mathbf{y}_k)$, having one zero singular value in direction $\mathbf{v}_k$ and two other singular values equal to one. If one simply leaves out a row of $\tilde{\mathbf{N}}_k$, the non-zero singular values will in general be unequal to one and mutually different.

a function of $\mathbf{r}$ using the *Moore-Penrose pseudo-inverse*, $\mathbf{A}_{st}^{\dagger} = (\mathbf{A}_{st}^{\mathsf{T}}\mathbf{A}_{st})^{-1}\mathbf{A}_{st}^{\mathsf{T}}$ which in turn can be computed efficiently and numerically stably with the singular value decomposition, $\mathbf{A}_{st} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^{\mathsf{T}}$ [GVL96],

$$\begin{bmatrix} s \\ \mathbf{t} \end{bmatrix} = -\underbrace{\mathbf{V}\mathbf{\Sigma}^{\dagger}\mathbf{U}^{\mathsf{T}}}_{\mathbf{A}_{st}^{\dagger}}\mathbf{A}_r\mathbf{r}. \tag{2.8}$$

By plugging this expression back into Eq. 2.4 and by defining the fixed size matrix

$$\mathbf{M}_h \in \mathbb{R}^{9\times 9} = \mathbf{A}_r^{\mathsf{T}}\mathbf{A}_r - \mathbf{A}_r^{\mathsf{T}}\mathbf{U}(\mathbf{\Sigma}^{\dagger}\mathbf{\Sigma})\mathbf{U}^{\mathsf{T}}\mathbf{A}_r, \tag{2.9}$$

we can describe the registration problem by the following constrained minimization problem which now only depends on the rotation parameters $\mathbf{r}$,

$$\underset{\mathbf{R}\in SO(3)}{\arg\min}\ \{\mathbf{r}^{\mathsf{T}}\mathbf{M}_h\mathbf{r}\}, \tag{2.10}$$

The expression for $\mathbf{M}_h$ has been simplified thanks to the orthogonality of the singular vector matrices, $\mathbf{V}^{\mathsf{T}}\mathbf{V} = \mathbf{I}$ and $\mathbf{U}^{\mathsf{T}}\mathbf{U} = \mathbf{I}$. The diagonal matrix $(\mathbf{\Sigma}^{\dagger}\mathbf{\Sigma})$ usually is a 4×4 identity matrix. Only in certain degenerate configurations it may also have zeros on its diagonal whenever some singular values of $\mathbf{A}_{st}$ are zero (or smaller than a machine precision dependent threshold). It then acts as a column selector for $\mathbf{U}$, so that only those singular vectors are chosen that effectively span the column space of $\mathbf{A}_{st}$. Rewriting $\mathbf{M}_h$ in Eq. 2.9 as $\mathbf{M}_h = \mathbf{A}_r^{\mathsf{T}}(\mathbf{I} - \mathbf{U}(\mathbf{\Sigma}^{\dagger}\mathbf{\Sigma})\mathbf{U}^{\mathsf{T}})\mathbf{A}_r$ one can see that $\mathbf{A}_r$ is again projected onto the orthogonal complement $(\mathbf{I} - \mathbf{U}(\mathbf{\Sigma}^{\dagger}\mathbf{\Sigma})\mathbf{U}^{\mathsf{T}})$ of the column space of $\mathbf{A}_{st}$. This mechanism allows to elegantly cope with some degenerate configurations which we will discuss later in this chapter.

Considering the computational effort, one might still ask, whether using the singular value decomposition for computing the pseudoinverse is a good choice, because for general $n\times m$ matrices the complexity for its computation is $\mathcal{O}(n^2m + m^3)$ [GVL96]. In our case, $\mathbf{A}_{st} \in \mathbb{R}^{n\times 4}$, so even though $m = 4$ is constant, we are still left with quadratic complexity with respect to the number of equations or constraints, n. It is important to note, however, that only the first four columns of $\mathbf{U}$ are needed. In this case - which is often referred to as *thin SVD* or *reduced SVD* - the number of computations can be reduced to $\mathcal{O}(nm^2 + m^3)$. This is an important part of our formulation because it still allows to compute $\mathbf{M}_h$ in linear time and thus also the effort for the whole registration problem remains linear. Most matrix libraries offer appropriate routine options.[3]

2.3.3. Relation to Existing PnP Approaches

We would like to point out commonalities and differences on how the registration problem is formulated in previous approaches for the PnP and generalized PnP case [HR11, KLS14, SFHT14, SFHT16]. The typical procedure is to describe the problem by a system of (noise-free) equations, which in their most general form are as follows:

$$\underbrace{\begin{bmatrix} \mathbf{v}_1 & & & \mathbf{y}_1 & -\mathbf{I}^{3\times 3} \\ & \ddots & & \vdots & \vdots \\ & & \mathbf{v}_K & \mathbf{y}_K & -\mathbf{I}^{3\times 3} \end{bmatrix}}_{=:\tilde{\mathbf{A}}} \underbrace{\begin{bmatrix} \lambda_1 \\ \vdots \\ \lambda_K \\ s \\ \mathbf{t} \end{bmatrix}}_{=:\tilde{\mathbf{u}}} = \underbrace{\begin{bmatrix} \mathbf{R} & & \\ & \ddots & \\ & & \mathbf{R} \end{bmatrix}}_{=:\tilde{\mathbf{W}}} \underbrace{\begin{bmatrix} \mathbf{x}_1 \\ \vdots \\ \mathbf{x}_K \end{bmatrix}}_{=:\tilde{\mathbf{x}}}, \tag{2.11}$$

[3] Matlab provides the option `'economy'` to the SVD-routine. In LAPACK the routine `DGESVD` comes with the option `JOBU='S'` or `JOBU='O'`. In Eigen one can set `ComputeThinU` in the constructor of the template `JacobiSVD<.>`.

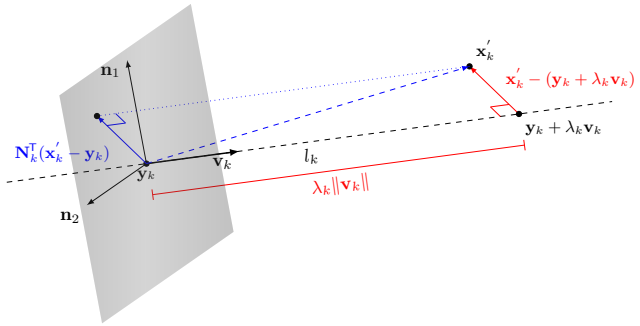


Figure 2.2.: Visualization of the equivalence of the geometric errors minimized in previous approaches (red) and in our approach (blue).

where additional virtual depth parameters $\lambda_k \in \mathbb{R}$ for the points are introduced. These ensure that in the noise-less case the scaled image points, $\mathbf{y}_k + \lambda_k \mathbf{v}_k$, coincide with their corresponding transformed points $\mathbf{x}'_k$. As in our case, the next step consists in eliminating the linear parameters $\tilde{\mathbf{u}}$ by means of the pseudo-inverse, $\tilde{\mathbf{A}}^\dagger = (\tilde{\mathbf{A}}^\mathsf{T}\tilde{\mathbf{A}})^{-1}\tilde{\mathbf{A}}^\mathsf{T}$, and expressing them in terms of the rotation parameters

$$\tilde{\mathbf{u}} = \tilde{\mathbf{A}}^\dagger \tilde{\mathbf{W}} \tilde{\mathbf{x}}. \tag{2.12}$$

After inserting this expression back into Eq. 2.11, the resulting system of equations depends only on the rotation $\mathbf{R} \in SO(3)$ (or the vectorized rotation matrix $\mathbf{r}$). The final minimization problem to determine the rotation has again the following form:

$$\underset{\mathbf{R} \in SO(3)}{\arg\min} \{\mathbf{r}^\mathsf{T} \tilde{\mathbf{M}} \mathbf{r}\}, \tag{2.13}$$

with

$$\tilde{\mathbf{M}} = \begin{bmatrix} \mathbf{I} \otimes \mathbf{x}_1, & \cdots, & \mathbf{I} \otimes \mathbf{x}_K \end{bmatrix} (\mathbf{I} - \tilde{\mathbf{A}}\tilde{\mathbf{A}}^\dagger) \begin{bmatrix} \mathbf{I} \otimes \mathbf{x}_1^\mathsf{T} \\ \vdots \\ \mathbf{I} \otimes \mathbf{x}_K^\mathsf{T} \end{bmatrix}. \tag{2.14}$$

It is important to note that this formulation for the PnP problem and its generalization minimizes the same error as in our case with point-to-line correspondences. The previous approaches [HR11, KLS14, SFHT14, SFHT16] minimize the Euclidean distance between the transformed point $\mathbf{x}'_k$ and the point $\mathbf{y}_k + \lambda \mathbf{v}_k$, which represents the line l_k parameterized by the depth value λ_k. Since the depth parameter is included as optimization variable inside the whole minimization problem, it will attain its optimal value when the vector $\mathbf{x}'_k - (\mathbf{y}_k + \lambda \mathbf{v}_k)$ is exactly orthogonal to the line l_k or the vector $\mathbf{v}_k$ (see Fig. 2.2). Otherwise, the error could still be reduced by changing λ_k while leaving the other parameters fixed. In our formulation, the vector $\mathbf{x}'_k - \mathbf{y}_k$ is directly projected onto the orthogonal complement of the line, which is spanned by the columns of the matrix $\mathbf{N}_k = [\mathbf{n}_1, \mathbf{n}_2]$. As a consequence, the lengths of both vectors, $\|\mathbf{x}'_k - (\mathbf{y}_k + \lambda \mathbf{v}_k)\|$ and $\|\mathbf{N}_k^\mathsf{T}(\mathbf{x}'_k - \mathbf{y}_k)\|$, are equal at the minimum of their respective objective functions. We also note that for non-degenerate configurations the resulting matrices, $\tilde{\mathbf{M}}$ and $\mathbf{M}$, are identical up to small numerical differences when computed with the previous approaches and with ours.

The major difference of previous formulations is that the geometric entities are described by their *affine subspaces* (represented by the bearing vector $\mathbf{v}_k$) and not by their *orthogonal complement* as in our case. This makes it necessary to introduce the virtual depth parameters λ_k. The downside is that the involved matrices $\tilde{\mathbf{A}}$

and $\tilde{\mathbf{W}}$ become very sparse and much larger than in our case. In particular, computing the pseudoinverse of $\tilde{\mathbf{A}} \in \mathbb{R}^{3K \times K+4}$ is prohibitively costly if one resorts to standard techniques for dense matrices. For this reason, an important aspect in the aforementioned papers is the presentation of a custom-made computation of the pseudoinverse of $\tilde{\mathbf{A}}$ and the final composition of $\tilde{\mathbf{M}}$. Special care was taken to exploit the sparsity of the matrices, and thus, to preserve the linear complexity of the whole algorithm. Yet still, the computation of $\tilde{\mathbf{M}}$ remains up to 2.5 times slower than in our proposed method, as we show in our simulations.

Furthermore, the derivations of $\tilde{\mathbf{M}}$ make specific assumptions on the targeted problem instance, so they are only applicable to PnP-type problems (or to 3D point-to-line registrations). It would be possible to extend these subspace-based parameterizations to the registration of point-to-plane correspondences. One would then introduce two "depth" parameters per plane and a 3×2 matrix $\mathbf{V}_k$ whose orthonormal column vectors span the subspace of the plane instead of a single bearing vector $\mathbf{v}_k$. However, computations would only get more complicated as one would have to track down the type of correspondence along the whole process of generating $\tilde{\mathbf{M}}$. In our case, once the matrix $\mathbf{A}$ is set up (Eq. 2.5), all information on the correspondence type is essentially hidden. In order to compute the pseudoinverse of the dense matrix $\mathbf{A}_{st}$ and finally $\mathbf{M}$, one can always use the same technique, no matter if $\mathbf{A}_{st}$ was composed from point-to-point, point-to-line, point-to-plane correspondences or any mixture of them.

2.3.4. Minimal Number of Constraints and the Inhomogeneous Case

So far, we have restricted our discussion on the full seven DoF problem, i.e. the Euclidean registration *with scale*. Intuitively, one will also need seven constraining equations for the problem in Eq. 2.4 to be solvable in general. It does not matter from which kind of correspondence types these constraints are obtained, the important part is that the minimal number of seven effective constraints are reached in total and that each 3D point-to-plane, point-to-line, or point-to-point yields one, two or three constraints, respectively. For example one can compute the registration parameters from seven point-to-plane correspondences only, where each correspondence gives rise to one equation. In previous approaches to pose-and-scale estimation [VARS14b, SFHT14] at least 'three-and-a-half' 2D-3D correspondences are needed, which is in accordance with our formulation, where constraints arising from 2D image point measurements are translated into 3D point-to-line correspondences. For general configurations, i.e. when the image measurements are distributed in more than one camera (also referred to as the *non-central case* [KLS14]), the sub-matrix $\mathbf{A}_{st}$ has full rank four. So in the process of eliminating the linear parameters by means of the pseudoinverse, $\mathbf{A}_{st}^{\dagger}$, the final matrix $\mathbf{M}_h \in \mathbb{R}^{9\times 9}$ will have at least rank three, which is a necessary requirement for solving for the remaining three DoF of the rotation. Since the seven DoF problem forms a homogeneous system of equations we also refer to this as the *homogeneous case*.

By constrast, if the scale parameter is already known or only the rotation and translation is to be estimated, then one would not eliminate the scale parameter. The correct procedure is then to compute the SVD only on the matrix $\mathbf{A}_t$, i.e. the colums of $\mathbf{A} = [\mathbf{A}_{rs}\ \mathbf{A}_t]$ that belong to the translation part, and finally solve the slightly modified problem

$$\underset{\mathbf{R}\in SO(3)}{\arg\min} \left\{ [\mathbf{r}^{\mathsf{T}}, 1]\, \mathbf{M}_i \begin{bmatrix} \mathbf{r} \\ 1 \end{bmatrix} \right\}, \quad \text{with } \mathbf{M}_i \in \mathbb{R}^{10\times 10}, \tag{2.15}$$

which we call the *inhomogeneous case*.

There is an important connection between the homogeneous and the inhomogeneous version, which happens when all lines and planes have one common intersection point. This corresponds to the situation, when the camera pose is estimated from measurements in a single camera only as in the *central PnP or PnL* case. Clearly, for single-view pose estimation, the scale parameter is meaningless and cannot be computed. In the

homogeneous case, the problem is therefore ill-conditioned which manifests itself in the matrix $\mathbf{A}_{st} \in \mathbb{R}^{n\times 4}$ having only rank three.[4] As a consequence, the pseudoinverse cannot be computed using the explicit formula $\mathbf{A}_{st}^{\dagger} = (\mathbf{A}_{st}^{\mathsf{T}}\mathbf{A}_{st})^{-1}\mathbf{A}_{st}^{\mathsf{T}}$ because $(\mathbf{A}_{st}^{\mathsf{T}}\mathbf{A}_{st})$ is singular. However, by using the SVD instead, as proposed in Sec. 2.3.2, this degeneracy is automatically handled correctly by means of the matrix $\mathbf{\Sigma}^{\dagger}\mathbf{\Sigma}$. This leads to the important property that the solutions to the remaining parameters, $\mathbf{R}$ and $\mathbf{t}$, can still be computed even if only six effective constraints are provided (e.g. three 2D-3D correspondences for the PnP problem), because the resulting matrix $\mathbf{M}_h$ still has rank three. Regarding the inhomogeneous matrix $\mathbf{M}_i$, we note that the column $\mathbf{a}_s$ of $\mathbf{A} = [\mathbf{A}_r, \mathbf{a}_s, \mathbf{A}_t]$ belonging to the scale parameter is a linear combination of the columns of $\mathbf{A}_t$ and therefore its projection onto the orthogonal complement of $\mathbf{A}_t$, i.e. $(\mathbf{I} - \mathbf{U}(\mathbf{\Sigma}^{\dagger}\mathbf{\Sigma})\mathbf{U}^{\mathsf{T}})$, is zero. Consequently, the last row and the column of the resulting matrix $\mathbf{M}_i$ will then also be zero and the upper-left 9×9 sub-matrix in $\mathbf{M}_i$ is identical to $\mathbf{M}_h$.

To summarize, in the central case both matrices $\mathbf{M}_i$ and $\mathbf{M}_h$ carry the same information for the solution of the rotation. And as both, the central case (like [HR11, ZKS*13]) and the homogeneous non-central case ([SFHT14]), can be represented by 9×9 matrices $\mathbf{M}_h$, we expect that the corresponding algebraic solvers for the nonlinear rotation can be used interchangeably for both types of problems. Further, we expect that an algebraic solver working on a 10×10 matrix $\mathbf{M}_i$ capable of solving both the central case and the inhomogeneous non-central case, such as the one inside the approach of Kneip et al. [KLS14], can be applied to all problems considered here.

2.3.5. Point-to-Plane Metric and its Relation to the PnL Problem

In this section, we discuss the connection between the registration from point-to-plane correspondences and the perspective-n-line problem (PnL). PnL refers to the camera pose computation w.r.t. a reference coordinate system from a set of detected straight 2D line segments in the image and their associated 3D reference lines. Common to the vast majority of existing PnL algoritms is that 2D line segments are interpreted as planes that pass through the camera center point and all of the (homogenized) image points of the 2D image line. They differ only in the way, how the 3D reference lines are integrated into to optimization problem.

In the classical approaches (e.g. [MR11, XZCK17, DRLR89, Che90]), solutions for the rotation are obtained by using only the normals of the plane, $\mathbf{n}_k \in \mathbb{R}^{3\times 1}$, and the *directions* of the 3D reference lines, $\mathbf{d}_k$, i.e. the normalized vector between any points on the reference line ($\mathbf{d}_k = (\mathbf{x}_{k_1} - \mathbf{x}_{k_2})/\|\mathbf{x}_{k_1} - \mathbf{x}_{k_2}\|$). Ideally the 3D lines should lie inside their corresponding planes after transformation, i.e. the rotated direction vector should be orthogonal to the plane normal. Thus, the following error is minimized [MR11]:

$$\underset{\mathbf{R}\in SO(3)}{\arg\min} \sum_{k=1}^{K} \mathbf{n}_k^{\mathsf{T}}\mathbf{R}\mathbf{d}_k. \tag{2.16}$$

Again, this can be expressed as a rotation-only least-squares problem of the form:

$$\underset{\mathbf{R}\in SO(3)}{\arg\min}\ \mathbf{r}^{\mathsf{T}}\mathbf{M}_{(l)}\mathbf{r}, \qquad \text{with} \quad \mathbf{M}_{(l)} = \mathbf{A}_{(l)}^{\mathsf{T}}\mathbf{A}_{(l)}. \tag{2.17}$$

Each row of $\mathbf{A}_{(l)}$ is given by

$$\mathbf{a}_{(l)k} \in \mathbb{R}^{1\times 9} = \mathbf{n}_k^{\mathsf{T}} \otimes \mathbf{d}_k^{\mathsf{T}}. \tag{2.18}$$

[4] The rank deficiency of $\mathbf{A}_{st}$ in the central case is obvious, if $\mathbf{y}_k = \mathbf{0}$ for all measurements (see Eq. 2.5) as in standard PnP or PnL. Beyond that, it is directly linked to the geometric configuration, i.e. it will occur whenever the lines or planes share a common intersection point. It neither depends on the actual coordinates of the intersection nor on the individual choices of offset points $\mathbf{y}_k$ for representing the lines or the planes.

This approach treats all lines equally regardless of their actual length in the image. Intuitively, both 2D and 3D lines are interpreted as if they were stretching to infinity. In practice however, the detected 2D image lines only have a limited and varying length. Shorter 2D line segment detections usually tend to be noisier than longer ones, so it makes sense to also let them have different influences on the outcome of the registration.

In a recent publication by Vakhitov et al. [VFMN16], these considerations are naturally addressed by registering the end-points of the 3D line to the plane spanned by image line and the camera center. Having an initial estimate for the pose, they additionally back-project the end-points of the 2D line segments onto the transformed 3D lines.[5] Then, they recompute the pose with the new point pairs of the 3D lines. We note that this second pass is only necessary for overconstrained cases in order to better address the noise from the 2D line measurements. For the minimal case the actual choice of reference points on the 3D line is irrelevant and will always yield the same solutions. The authors demonstrated the superiority of their PnL approach compared to classical ones.

As in our case, the approach Vakhitov et al. also allows to combine different correspondence types (including 2D point measurements). However, it is based on the minimization of an *algebraic error* (inspired by OPnP [ZKS*13] and EPnP [LMNF09]), which makes a further extension to generalized cameras difficult. Our formulation for point-to-plane correspondences is based on the *object space error* and we consider the registration problem for general cases, including situations where the planes do not share a common intersection point.

2.3.6. Efficiently Pre-Rotating Reference Points

Often it is advantageous to work with a modified $\widehat{\mathbf{M}}_h$ that is derived by simply pre-rotating the reference points $\mathbf{x}_k$ with some rotation matrix $\mathbf{R}_0$, i.e. $\widehat{\mathbf{x}}_k = \mathbf{R}_0\mathbf{x}_k$. Any solution, $\widehat{\mathbf{R}}$, obtained on the basis of $\widehat{\mathbf{M}}_h$ is then also a rotated version of the original solution $\mathbf{R}$, i.e. $\widehat{\mathbf{R}} = \mathbf{R}\mathbf{R}_0^\mathsf{T}$. The algebraic solvers that will be discussed in the next section may fail to determine the correct solutions in all cases. In particular, solvers based on the Cayley parameterization for rotation matrices will not succeed whenever the correct solution for the rotation has an angle of π. In this case one can re-evaluate the problem for different $\widehat{\mathbf{M}}_h$ and collect all solutions. Although not published by Hesch et al. [HR11], the same authors implemented this strategy as an improved version of the DLS algorithm[6], which was later also adopted by others [Nak15]. Another use case is the post-refinement of solutions for $\mathbf{R}$ with a second order Newton minimization applied on a matrix $\widehat{\mathbf{M}}_h$ with the rotation $\widehat{\mathbf{R}}$ being optimized is close to the identity. This was done *e.g.* by Zheng et al. and Kneip et al. [ZKS*13, KLS14].

Instead of re-evaluating $\widehat{\mathbf{M}}_h$ each time for the rotated points $\widehat{\mathbf{x}}_k$ from scratch, we observe that it is also possible to manipulate the matrix $\mathbf{M}_h$, directly. This has the same effect but it can be computed in constant time, whereas a full re-evaluation requires a linear effort with respect to the number of correspondences. Recomputing $\widehat{\mathbf{M}}_h$ for some $\mathbf{R}_0$ thus becomes a negligible operation compared to the actual solving step of the algebraic solver.

As can be seen from Eq. 2.5, rotating the reference points $\mathbf{x}_k$ leaves $\mathbf{A}_{st}$ unchanged and $\mathbf{A}_r$ changes as follows,

$$\widehat{\mathbf{A}}_r = \mathbf{A}_r\mathcal{R}_0, \quad \text{with } \mathcal{R}_0 = \begin{bmatrix} \mathbf{R}_0 & & \\ & \mathbf{R}_0 & \\ & & \mathbf{R}_0 \end{bmatrix}, \tag{2.19}$$

which together with Eq. 2.9 yields

[5] In practice the end-points of the detected line correspondences often do not correspond to the end-point of a 3D model line, due to occlusions or other failures in the detection process.

[6] The implementation of DLS is available at `http://www-users.cs.umn.edu/~joel/`

$$\widehat{\mathbf{M}}_h = \mathcal{R}_0^\mathsf{T} \mathbf{M}_h \mathcal{R}_0. \tag{2.20}$$

One can partition $\mathbf{M}_h$ into nine 3×3 submatrices $\mathbf{M}_h^{(i,j)}$ and transform each of them individually, i.e. $\widehat{\mathbf{M}}_h^{(i,j)} = \mathbf{R}_0^\mathsf{T} \mathbf{M}_h^{(i,j)} \mathbf{R}_0$. Thus, it is possible to exploit the sparsity of $\mathcal{R}_0$ and to avoid explicitly contructing it as a matrix.

Considering again the Cayley parameterization, the traditional procedure consists of re-evaluating the problem for *two extra randomly gerenated* $\mathbf{R}_0$. We note, that the set of Cayley singularities actually forms a two-dimensional manifold. So, in order to guarantee that $\widehat{\mathbf{R}}$ never is near the set of these singularities for all evaluations, one actually has to perform *four evaluations* in total. This is because for three arbitrary pre-rotating matrices, $\mathbf{R}_i$, $i \in 1,2,3$, one can always find a forth rotation, $\mathbf{R}_4$, so that $\mathbf{R}_4^\mathsf{T} \mathbf{R}_i$ has an rotation angle of π for all $\mathbf{R}_i$. Instead of generating the pre-rotation matrix $\mathbf{R}_0$ randomly, we propose to select it from the canonical set of rotations,

$$\mathbf{R}_0 \in \left\{ \begin{bmatrix} 1 & & \\ & 1 & \\ & & 1 \end{bmatrix}, \begin{bmatrix} 1 & & \\ & -1 & \\ & & -1 \end{bmatrix}, \begin{bmatrix} -1 & & \\ & 1 & \\ & & -1 \end{bmatrix}, \begin{bmatrix} -1 & & \\ & -1 & \\ & & 1 \end{bmatrix} \right\}, \tag{2.21}$$

where the relative rotation between any two of these elements has the angle π. In this case, recomputing $\widehat{\mathbf{M}}_h$ simply amounts to changing signs of some of the entries in $\mathbf{M}_h$. This can be achieved almost instantly and it alleviates some of the common objections against the use of Cayley parameterization.

2.4. Algebraic Solvers for the Rotation

We will now turn our attention towards solving for the nonlinear rotation, i.e. finding all solutions of Eq. 2.10, Eq. 2.15 or Eq 2.17. To this end, the rotation matrix is parameterized either by quaternions or via Cayley parameters. For a quaternion, $\mathbf{q} = [q_0, q_1, q_2, q_3]^\mathsf{T}$, with real part q_0, the rotation matrix is given by

$$\mathbf{R}(\mathbf{q}) = \frac{1}{\mathbf{q}^\mathsf{T}\mathbf{q}} \begin{bmatrix} q_0^2 + q_1^2 - q_2^2 - q_3^2, & 2(q_1 q_2 - q_0 q_3), & 2(q_1 q_3 + q_0 q_2) \\ 2(q_1 q_2 + q_0 q_3), & q_0^2 - q_1^2 + q_2^2 - q_3^2, & 2(q_2 q_3 - q_0 q_1) \\ 2(q_1 q_3 - q_0 q_2), & 2(q_2 q_3 + q_0 q_1), & q_0^2 - q_1^2 - q_2^2 + q_3^2 \end{bmatrix}. \tag{2.22}$$

The Cayley parameterization is given by simply fixing $q_0 = 1$.

The first order optimality conditions are obtained by taking the derivative of the error function with respect to the four quaternion parameters, which leads to a system of four equations with monomials in q_i of degree three (three equations for Cayley parametrization).

$$\frac{\partial}{\partial q_i} \mathbf{r}(\mathbf{q})^\mathsf{T} \cdot \mathbf{M} \cdot \mathbf{r}(\mathbf{q}) = 0 \tag{2.23}$$

In order to obtain the solutions for the rotation, it is required to find the roots of this system of multivariate polynomial equations.

2.4.1. DLS/gDLS Solver

The DLS solver [HR11] uses the Macaulay resultant method to solve the algebraic equations. It uses the Cayley parametrization. For the final solution, the eigenvectors of a 27×27 matrix are evaluated, so the solver may return up to 27 real solutions. This is way more than the maximum number of eight that can theoretically occur

for minimal configurations (e.g. three lines or six planes in the central case). The solutions may also contain other stationary points such as maxima and saddle points. These must be filtered out afterwards with additional checks.

The solver of gDLS [SFHT14] is a transcription of the DLS solver from Matlab code to C++ using Eigen as math library. Apart from that they are absolutely identical and later we use the gDLS-version for efficiency reasons. It can be found inside the Theia library[7] [SHT15]. For the evaluations, we separate everything related to setting up the matrix $\mathbf{M}_h$ from the actual rotation solver, so we are able to evaluate them separately.

2.4.2. UPnP Solver

For the UPnP algorithm, Kneip et al. [KLS14] derived a rotation solver using the Gröbner basis technique by means of an own automatic solver generator similar to Kukelova's et al. [KBP08]. The derived rotation solver internally builds up an elimination template of size 141×149. This is reduced to an 8×8 action matrix, from which the solutions are obtained by eigenvalue decomposition. Thus, it returns at most eight solutions. This is exactly the maximum number that can theoretically occur for minimal configurations (such as three correspondences in the central PnP case), which is one of the appealing properties of this rotation solver.

This was achieved by three particularities:

1. They used quaternions and dropped the normalization term $1/\mathbf{q}^\mathsf{T}\mathbf{q}$ in Eq. 2.22, but in order to enforce *unit norm length* on the quaternions, they added four additional equations,

$$\frac{\partial}{\partial q_i}(\mathbf{q}^\mathsf{T}\mathbf{q}-1)^2 = 0, \qquad i \in \{0,\cdots,3\} \tag{2.24}$$

   .
2. They took into account the *two-fold symmetry* of quaternions[8] [AKr12] exploiting the fact that the system of equations only involves polynomials of odd degree. This excludes the trivial solution and halves the number of non-trivial ones.
3. They derived the solver for the minimal case, which ensures that it will always return eight solutions. To this end, the solver generator needs to be *instantiated in $\mathbb{Z}_p$ in a coherent way*, so that the Eqs. 2.23 and 2.24 are satisfied for integer values. When applying the resulting solver to real-valued, non-minimal configurations after derivation, some of the solutions will be complex-valued, which can be discarded right away.

A C++ Version of the final algorithm can be found inside the OpenGV framework[9] [KF14]. Again, we separated the linear parameter elimination step from the actual rotation solver.

2.4.3. GAPS: Our Own Solver

We also developed an own solver following the main ideas presented by Kneip et al. [KLS14], but with the difference that we used an own, modified version of the automatic generator by Kukelova et al. [KBP08], as we did not have access to the generator by Kneip et al. . The main idea of the generator is that it 'simulates' the given problem with integer values in the field $\mathbb{Z}_p$.[10] From this integer-valued simulation, the Gröbner basis is derived together with several other structural properties, such as the size of the elimination template, linear dependent

[7] `http://www.theia-sfm.org/`
[8] For a quaternion $\mathbf{q}$, $\mathbf{R}(\mathbf{q}) = \mathbf{R}(-\mathbf{q})$.
[9] `http://laurentkneip.github.io/opengv/`
[10] $\mathbb{Z}_p$ denotes the field of integer values modulo a large prime number p.

rows and columns, etc. until finally a piece of code is produced: the solver for the desired problem. The generator uses the computer algebra software *Macaulay2* as back-end for Gröbner basis computation. When experimenting with this generator and Macaulay2, it quickly became apparent that it is necessary to also consider all three particularities of the UPnP solver derivation in order to obtain a rotation solver with similar characteristics and efficiency. Notably, the following insights can be gained (summarized for the homogeneous case, see Eq. 2.10):

- Using unconstrained quaternion parameters (dropping Eqs. 2.24) for a general matrix $\mathbf{M}_h$ with randomly instantiated coefficients yields a working solver. However, it always outputs 81 solutions (including the trivial one $\mathbf{q} = \mathbf{0}$) and internally constructs an elimination template matrix of size 575×656. Therefore it is very inefficient.
- Respecting the two-fold symmetry reduces the number of solutions to 40, very much like it was done for the PnP solver of Zheng et al. [ZKS*13].
- Adding the four unit norm constraints of Eqs. 2.24 for a randomly (or even symmetrically) instantiated $\mathbf{M}_h$ results in a non-working solver that only returns the trivial solution.
- Adding the unit norm constraints *and* instantiating $\mathbf{M}_h$ for the minimal case, but without considering the two-fold symmetry, produces a solver with 17 solutions. However, this solver will *only* work for the minimal case. For overconstrained problems the structure of the elimination template changes and the solver fails.

Unfortunately, Kukelova's generator does not have the option of specifying any n-fold symmetries for certain problems. In order to implement this, modifications were necessary at various points deeply inside the code.

The $\mathbb{Z}_p$-instantiation can be modified very easily. It possible to write own code and add this as a function pointer to the generator (replacing the default one, `gbs_RandomInstanceZp()`). In order to generate coherent integer values for $\mathbf{M}_h$ and $\mathbf{M}_i$, so that they represent 'noiseless' minimal problems, we experimented with two strategies:

- *Pythagorean Quintuples*: This first variant is based on generating random integer rotation / orthogonal matrices in $\mathbb{Z}_p$ by means of Pythagorean quintuples. These are five integer values $c_1, \cdots, c_5$ that satisfy $c_5^2 = \sum c_i^2, i \in 1, \cdots, 4$. We can interpret $c_1, \cdots, c_4$ as the elements of a quaternion and c_5 as its norm and use Eq. 2.22 to create rotation / orthogonal matrices that satisfy $\mathbf{R}^\mathsf{T}\mathbf{R} \bmod p = \mathbf{I}$, where 'mod' denotes the modulus operator. With that tool at hand we can fully simulate a problem instance by generating orthogonal $\mathbf{R}$ and $\mathbf{N}$ and some $\mathbf{t}, \mathbf{x}, \mathbf{y}$ in $\mathbb{Z}_p$ and finally constructing $\mathbf{M}$.
- *Rank Three Constraint*: As pointed out in Sec. 2.3.4 both matrices, $\mathbf{M}_h$ and $\mathbf{M}_i$, have rank three for minimal configurations. We can exploit this property by creating random integer matrices $\mathbf{C}_h \in \mathbb{Z}_p^{3\times 9}$ or $\mathbf{C}_i \in \mathbb{Z}_p^{3\times 10}$ and setting $\mathbf{M} = \mathbf{C}^\mathsf{T}\mathbf{C} \bmod p$.

We observe that both strategies yield identical algorithms. In our ACCV paper [WK17], we derived an algorithm exclusively for the homogeneous case. Here, we additionally considered the more general inhomogeneous case, resulting in a rotation solver that is applicable to almost all problems considered in this work, including the fixed scale variants. Both C++ implementations internally construct an elimination template matrix of size 176×184, which is reduced to a final action matrix of size 8×8. Thus, the algorithms also return at most eight real solutions. We call the general, inhomogeneous variant GAPS[11] and the older version [WK17] hGAPS.

[11] The acronym stands for *General-Purpose Absolute Pose Solver*.

2.5. Evaluation

2.5.1. Accuracy

We conducted several synthetic evaluations for measuring the accuracy of the solvers. Our focus is on the applicability of our unifying framework presented in Sec. 2.3. Therefore, we replace the computation of $\mathbf{M}_h$ by our version and only use the polynomial solvers inside the algorithms (see Sec. 2.4). The solvers are denoted by **DLS/gDLS (OC)** [HR11, SFHT14], **UPnP (OC)** [KLS14], **hGAPS (OC)**, and **GAPS (OC)** (homogeneous and inhomogeneous solvers of Sec. 2.4.3), where *'OC'* refers to the substitution with our *orthogonal complement* formulation. We only use the raw polynomial solvers, so no root polishing is applied on the obtained solutions afterwards (as opposed to the original UPnP algorithm). For the DLS/gDLS solver which uses the cayley parametrization, we employ the strategy outlined in Sec. 2.3.6, i.e. we solve the problem four times, collect all solutions, and among the duplicates we only keep the ones that have the smaller error according to Eq. 2.10. We also apply the same strategy for the UPnP solver and our own solvers (hGAPS / GAPS), although they do not suffer from the singularities by the Cayley parameterization. We explain the reason for that in a separate evaluation below in Seq. 2.5.2.

2.5.1.1. General Configurations for Point, Line and Plane Registration

In a first experiment, we analyzed the accuracy of the full Euclidean registration with scale for general geometric configurations. We evaluated point-to-point, point-to-line, and point-to-plane registration separately (no mixture of correspondence types).

We created the evaluation data by first generating Gaussian distributed transformed points $\mathbf{x}_k^{'}$ with identity covariance. Given random ground-truth rotation $\mathbf{R}_{GT}$, translation $\mathbf{t}_{GT}$ and scale $\mathbf{s}_{GT} \in [0.1, 10]$ the inverse transformation is applied to obtain the reference points $\mathbf{x}_k$. The subspace spanned by a geometric entity and its orthogonal complement are obtained by partitioning the columns of a random orthogonal matrix into $\mathbf{N}_k$ and $\mathbf{V}_k$. In this subspace, a point is chosen randomly as offset point $\mathbf{y}_k$. Finally, Gaussian 3D noise is added to $\mathbf{x}_k$, whose covariance was kept fixed to 0.001 times the identity matrix throughout the evaluation run.

We varied the number of input correspondences that made up the matrix $\mathbf{M}_h$, which in turn was used as input for the algorithms. We evaluated the mean error of the rotation, translation, and scale with respect to ground-truth. Fig. 2.3 summarizes the results. It can be seen that it is possible to successfully estimate the transformation parameters with all four solvers for point-to-line and for point-to-plane correspondences and with almost identical accuracy.

This is an important result in several aspects. While Sweeney *et al.* [SFHT14] demonstrated that DLS Algorithm can be extended from the classical PnP Problem to the GPnP Problem with scale, we show here that it can also be used for registration from point-to-plane constraints. The same is true for the UPnP solver, which also has never been used for point-to-plane registration. In addition, we note that the UPnP Algorithm was originally proposed to solve the classical and generalized pose problem with *fixed scale*. Our evaluations demonstrate that it can also be used to solve the 7-DoF problem including the scale as free parameter.

However, we also observe, that UPnP and gDLS cannot be extended to point-to-point registration as they fail completely in estimating correct results. By contrast, our own solver succeeds in this scenario. We compare it to the algorithm of Umeyama [Ume91], which is the standard algorithm for this case. Both algorithms are almost identical regarding the accuracy.

In a complementary evaluation, we varied the (relative) noise level on the reference points, $\mathbf{x}_k$, instead of varying the number of correspondences. The number of correspondences was kept fixed to 50 throughout all

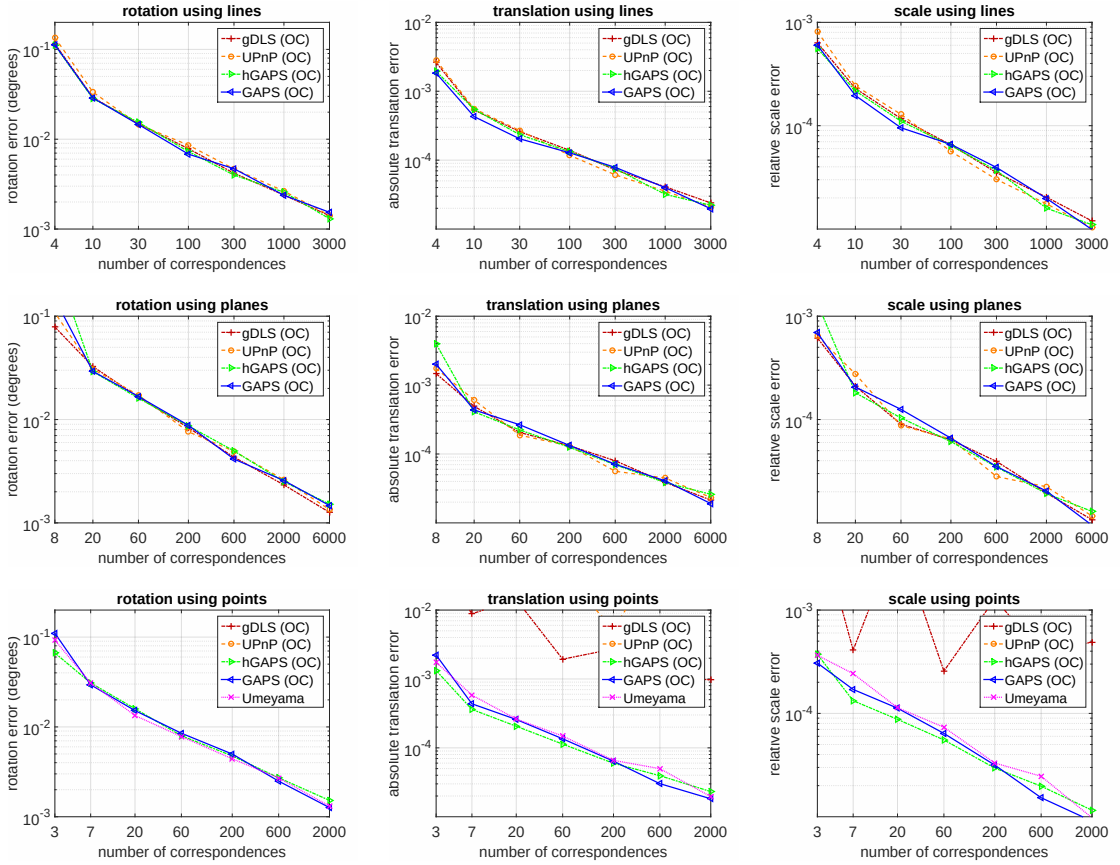


Figure 2.3.: Mean errors of the estimated rotation, translation, and scale for general geometric configurations with varying numbers of input correspondences.

evaluations. In each run, we computed the mean of the singular values of the covariance matrix of the points $\mathbf{x}_k$ and the covariance of the noise. The *'relative noise level'* was defined as the square root of their ratios. Again, we evaluated general geometric configurations of the lines, planes, and points.

Fig. 2.4 shows the results. Again, it can be seen that DLS/gDLS and UPnP do not succeed in correctly determining the registration parameters from point-to-point correspondences whereas our solvers do. For the other cases, we see that gDLS has slightly higher mean errors for very low noise levels.

2.5.1.2. Fixed Scale and the Inhomogeneous Case - General Configuration

We also analyzed the benefits of the GAPS and UPnP rotation solvers, because these are the only ones that accept the inhomogeneous matrix $\mathbf{M}_i \in \mathbb{R}^{10\times 10}$ as in Eq. 2.15. The other two algorithms can only handle 9×9 input matrices. Again, we performed the evaluation on general geometric configurations by varying the number of input correspondences. The relative noise ratio was set to 0.0001. For our own hGAPS solver [WK17] and the

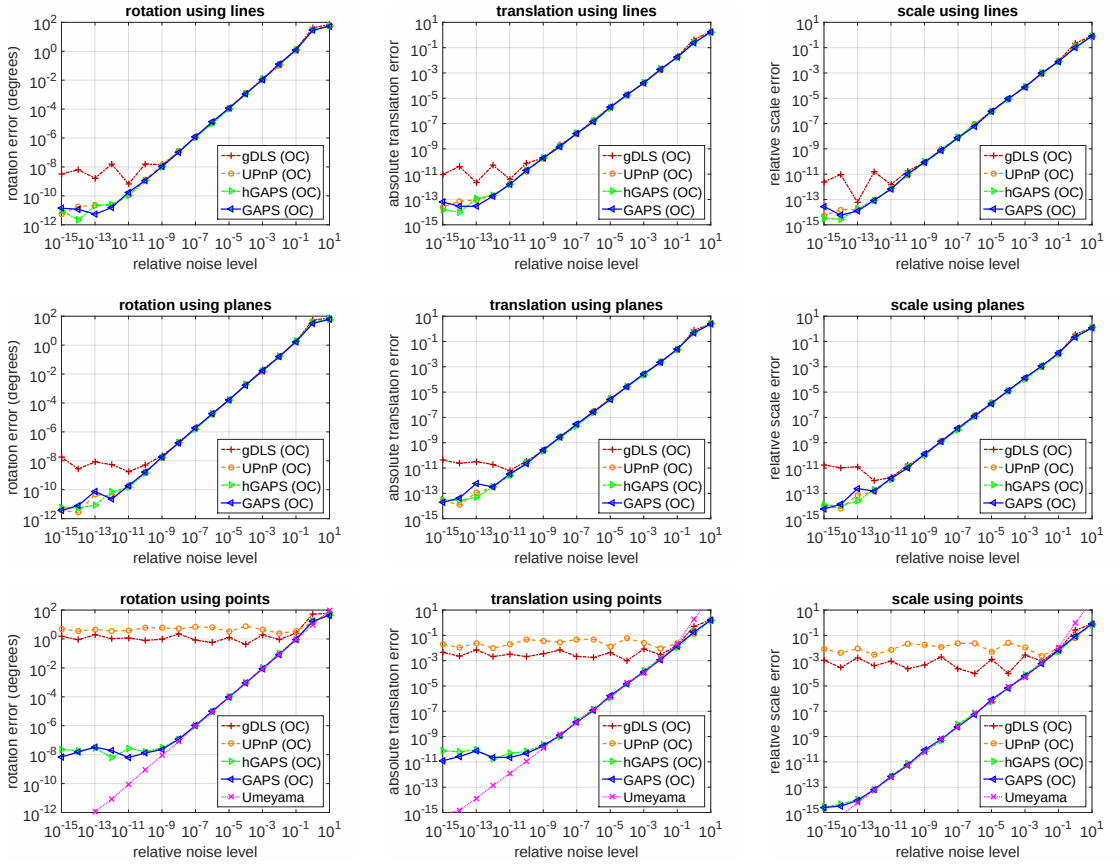


Figure 2.4.: Mean errors of the estimated rotation, translation, and scale for scaled Euclidean registration with 50 correspondences in general geometric configurations with varying noise. The relative noise level refers to the ratio between the standard deviations of the noise and the centered points $\mathbf{x}_k$.

DLS/gDLS solver, we computed the homogeneous matrix $\mathbf{M}_h$ as in Sec. 2.3.2. Note that this corresponds to leaving the scale as free parameter.

Fig. 2.5 shows the results. Using GAPS and UPnP on $\mathbf{M}_i$ leads to more accurate results as expected, because the problem is modeled more appropriately. A rather surprising result is that this only affects the translation, whereas the rotation estimates are equally accurate for all solvers. Note also that only GAPS and UPnP can solve the minimal cases of six point-to-plane or three point-to-line correspondences.

Fig. 2.5 does not include our evaluations when point-to-point correspondences are used exclusively. The matrix $\mathbf{M}_i$ will then have a very special block-diagonal structure (except the last row and columns) for which also our inhomogeneous solver GAPS has difficulties. At this point we note, however, that as soon as any mixture of correspondences is used, all of the considered solvers will succeed. Using only point-to-point correspondences remains a special case for which existing algorithms [AHB87, Hor87, HHN88, Ume91] are better suited, also regarding the runtime behavior.

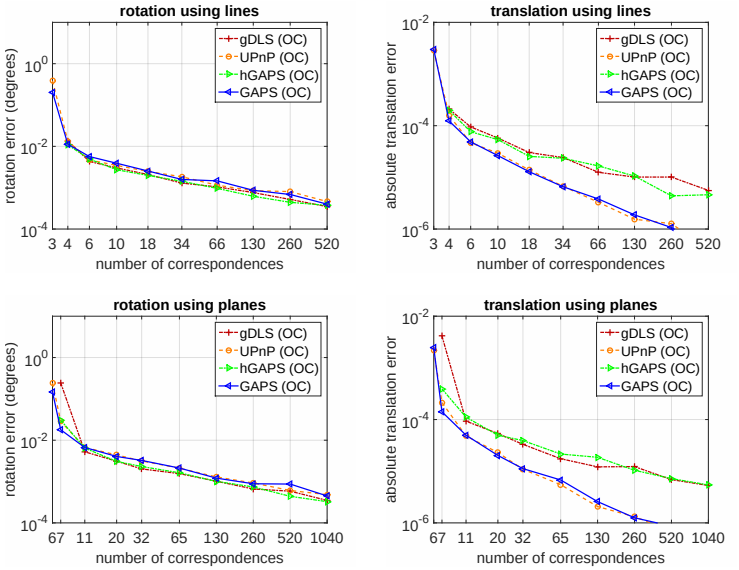


Figure 2.5.: Mean errors of the estimated rotation and translation for Euclidean registration with known and fixed scale (inhomogeneous case) versus number of used correspondences. The relative noise level was set to 0.0001. Only UPnP and GAPS can handle the minimal cases, i.e. 3 line correspondences or 6 plane correspondences.

2.5.1.3. The Degenerate Central Case for Lines and the PnP Problem

As we mentioned in Sec. 2.3.4, the central case, i.e. when all 3D lines have a common point of intersection, can be considered as a degenerate situation because the scale cannot be computed. This special case has a direct connection to the PnP problem, where all lines intersect at the camera center. For this reason - and also because the PnP problem has a very high relevance in computer vision - we use the PnP evaluation framework of Zheng et al. [ZKS*13][12] and included **UPnP**, **gDLS/DLS** and our own solvers **GAPS/hGAPS** into it. Again, we replaced the linear parameter elimination of the algorithms by our *orthogonal complement* based version (marked by '**(OC)**'). **Newton (OC)** refers to the direct second order (Newton) based minimization of Eq. 2.10.

For comparison we also include the well-known algorithms **LHM** [LHM00], **EPnP** [LMNF09], and **OPnP** [ZKS*13]. **LM** refers to the solution obtained by minimizing the reprojection error in the image plane with Levenberg-Marquardt iterations, which represents the baseline for PnP problems.

The evaluation setup is as follows. Random points $\mathbf{x}_k'$ in the camera coordinate frame are created inside the cube $[(-2,-2,4)\times(2,2,8)]$ and corresponding 2D image measurements $\mathbf{q}_k \in \mathbb{R}^2$ are generated by perspective division and by adding pixel noise with varying variance. Next, with a random ground-truth rotation $\mathbf{R}_{GT}$ and translation $\mathbf{t}_{GT}$ the corresponding reference points $\mathbf{x}_k$ are obtained. The image points, $\mathbf{q}_k$, and the reference points, $\mathbf{x}_k$, are used as input for all PnP algorithms.

Fig. 2.6 shows the results. It can be seen that the modified algorithms (OC) compare favorably with the state-of-the-art even for the minimal cases of three correspondences. Perhaps more important is the fact that

[12]The evaluation framework of Zheng and a collection of the most prominent PnP solvers are available at `https://sites.google.com/site/yinqiangzheng/`.

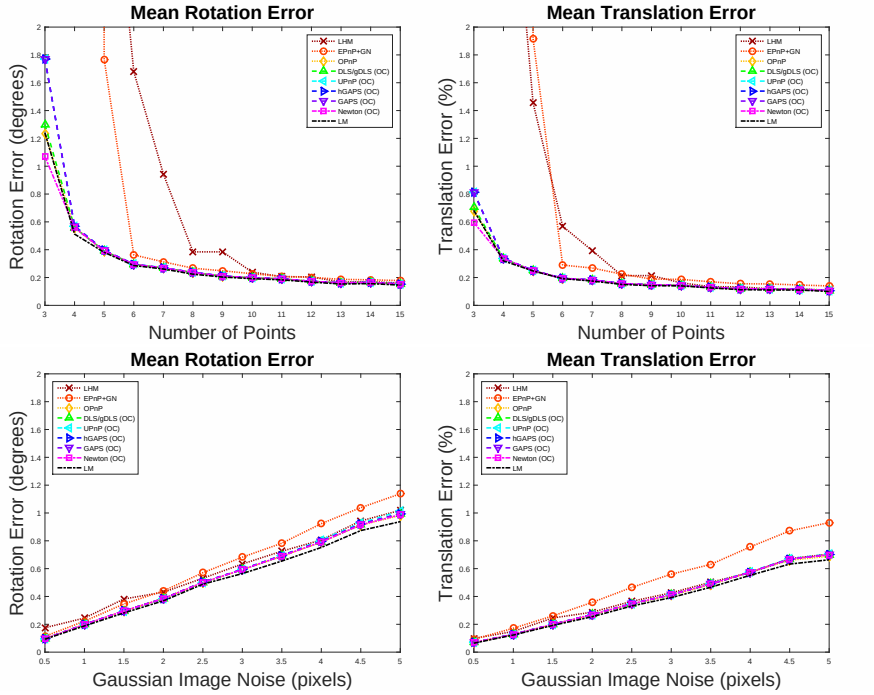


Figure 2.6.: Accuracy for the central PnP problem. Top: error of rotation and translation as a function of the number of input correspondences (pixel noise is set to $\sigma = 1.0$). Bottom: error with respect to pixel noise using 10 input correspondences.

the PnP problem - as being a degenerate case in the scaled Euclidean registration context - is handled properly by the SVD-based pseudo-inverse computation from Sec. 2.3.2, as all algorithms (UPnP, DLS/gDLS and our solvers) are supplied with the homogeneous matrix $\mathbf{M}_h$ from Eq. 2.9. We note again that linear parameter elimination inside the original gDLS algorithm [SFHT14] cannot be used here, because it will attempt to invert a singular matrix to obtain the pseudo-inverse $\tilde{\mathbf{A}}^{\dagger}$. Likewise, the linear parameter elimination technique from the original DLS algorithm [HR11] is not applicable whenever the scale needs to be estimated in general geometric configurations.

2.5.1.4. The Degenerate Case of Parallel Lines and Planes

We evaluated another degenerate case, where all planes and lines are parallel in one common direction $\mathbf{v}_0$. Then, the translation is ambiguous up to this common vector, so any $\mathbf{t} = \mathbf{t}_0 + \gamma \mathbf{v}_0$ represents a valid solution for arbitrary γ. Again, the linear parameter elimination techniques inside the original algorithms [HR11, SFHT14, KLS14] will fail, because they attempt to invert a singular matrix $\tilde{\mathbf{A}}^{\mathsf{T}}\tilde{\mathbf{A}}$ (see Sec. 2.3.3). By contrast, our SVD-based computation of the pseudo-inverse will handle this degeneracy correctly.

The evaluation was similar to the general case (Sec. 2.5.1.1) except that the lines and planes were constrained to be parallel to some random direction $\mathbf{v}_0$. For the translation error we projected the error vector between the translation offsets $\mathbf{t}_0$ returned by the algorithms and the ground-truth translation $\mathbf{t}_{GT}$ onto the orthogonal complement of $\mathbf{v}_0$, i.e. $err_{\mathbf{t}} = \|(\mathbf{I} - \mathbf{v}_0\mathbf{v}_0^{\mathsf{T}})(\mathbf{t}_{GT} - \mathbf{t}_0)\|$.

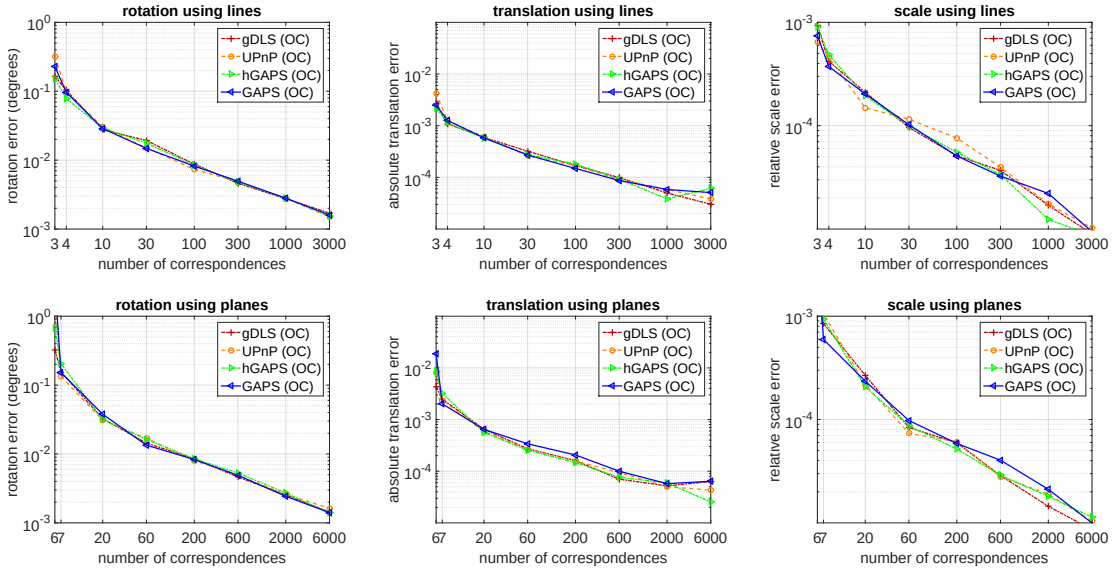


Figure 2.7.: Mean errors of the estimated rotation, translation, and scale for scaled Euclidean registration with the lines and planes being parallel to one common direction.

In Fig. 2.7 the results are depicted. It can be seen that all algorithms succeed in estimating correct results with similar accuracy. We also note that this is possible even for three point-to-line and six point-to-plane correspondences, despite the fact that we also solve for the scale. This is because once again only six DoF in total need to be estimated.

2.5.2. Numerical Stability under Strong Noise

It has already been observed in previous publications that the numerical stability of the UPnP Algorithm degrades for the central case (homogeneous case) when strong noise is present. Then it may still return very accurate solutions sometimes, but it also happens more frequently that none of the returned solutions is anywhere near the correct rotation. When evaluating the median value of the error instead of the mean value (see [KLS14, Nak15]) the algorithm still showed superior performance.

We further investigated this behavior by analyzing the stability as a function of the true rotation. We generated a set of 20 3D point-to-line correspondences using the identity matrix as initial ground truth rotation. We then added a fairly large quantity of noise to the data. The magnitude of the noise corresponded to five pixels standard deviation when projected onto the image plane in the PnP case. Next we rotated the reference points $\mathbf{x}_k$ with a smoothly varying rotation, $\mathbf{R}_0(\alpha)$. The parameter α was used for the real part q_0 and was varied in the range $[0,1]$. The imaginary values were all set to $q_i = \sqrt{(1-\alpha^2)}/\sqrt{3}$ for $i \in \{1,2,3\}$. The rest of the data, i.e. $\mathbf{y}_k$ and $\mathbf{N}_k$ and the noise, was left unchanged. For each corresponding matrix $\mathbf{M}_h(\alpha)$ we estimate the rotation using the algorithms from Sec. 2.4 and evaluated the error with respect to the ground-truth rotation $\mathbf{R}_0(\alpha)$.

Fig. 2.8 shows two independent evaluations. As expected, the DLS/gDLS algorithm fails to compute the correct solution near $\alpha = 0$, which represents an element in the set of Cayley degeneracies. The UPnP solver

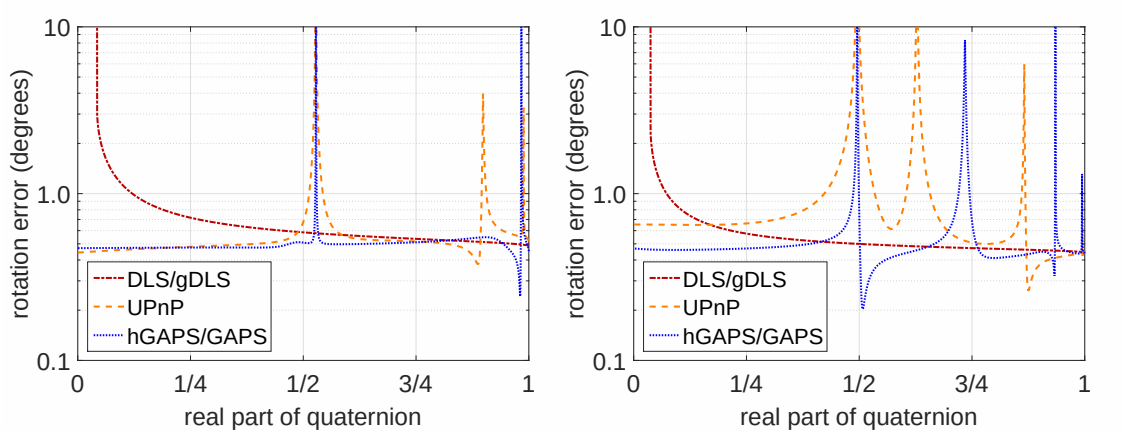


Figure 2.8.: Two independent evaluations showing the influence of the true rotation on the stability of the rotation solvers under strong noise.

Table 2.1.: Mean execution times of the different rotation solvers.

gDLS [SFHT14] (Sec. 2.4.1)	UPnP [KLS14] (Sec. 2.4.2)	hGAPS (Sec. 2.4.3)	GAPS (Sec. 2.4.3)
833.4 μsec	554.6 μsec	436.8 μsec	441.7 μsec

and our own solvers (hGAPS and GAPS)[13] also exhibit singularities. As opposed to the Cayley parameterization their location is not known in advance. However, as our analysis shows, they are also dependent on the rotation. This implies that the stability of these algorithms can be significantly improved by re-evaluating the problem for different pre-rotations of the reference points $\mathbf{x}_k$, which can be done very efficiently as shown in Sec. 2.3.6.

2.5.3. Runtime Analysis

We evaluated the runtime performance of the algorithms, which were all implemented in C++ and executed single threaded with 3.5GHz clock rate. Fig. 2.9 shows the timings for the linear parameter elimination part inside the gDLS and UPnP algorithm compared to our orthogonal complement based approach. We used 2D-3D correspondences from the PnP problem as input. Both versions exhibit linear complexity. For more than 13 input correspondences our approach is faster up to a factor of approximately 2.5.

Table 2.1 shows the mean execution times of the different algebraic solvers for 10.000 calls with varying data. The gDLS solver is the slowest and takes slightly more than 0.8 milliseconds, whereas our solvers (hGAPS and GAPS) are almost twice as fast. UPnP is slightly slower despite the fact that it uses a smaller elimination template.

[13] Both of our own solvers have the same singularities for the homogeneous case.

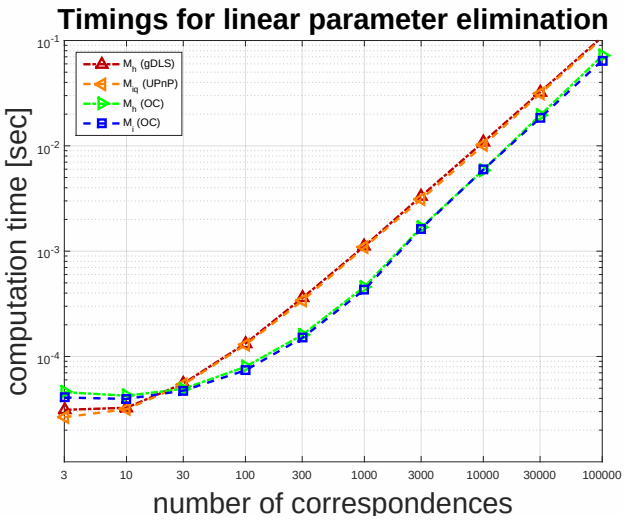


Figure 2.9.: Computational time of our linear parameter elimination step versus previous approaches.

2.6. Conclusion

In this chapter, we presented a generalization of algorithms for closed-form estimation of similarity or Euclidean transformations. Our proposed algorithm is applicable on a variety of 3D input correspondence types, namely point-to-point, point-to-line, point-to-plane, or any mixture of these. Our main motivation within this work is the spatial registration of reconstructed SLAM models to a desired virtual target coordinate system. In the further course of this work, we will frequently refer back to this algorithm as it serves as basis for the following chapters. In particular, it is used for registration by means of reference markers with partial links to the virtual coordinate system (Chap. 4) or via point correspondences obtained by a manual editing process (Chap. 5). It can also be considered as a preliminary step for bootstrapping the initial solution of the constrained bundle adjustment before final optimization (Chap. 3).

The applicability of this algorithm is not limited to the spatial registration of SLAM models only, but extends far beyond. In particular, we elucidated the analogy to the perspective-n-point (PnP) problem. Here the task is to compute the camera pose (rotation and translation) from a set of 2D points in the camera image and corresponding 3D points in the coordinate system of the tracking model. Since it is a fundamental problem of computer vision, it has attracted research efforts for many decades. We also showed that other problems, such as the perspective-n-line (PnL) problem or any variant for generalized cameras (such as for pose estimation from calibrated stereo setups) can also be solved with our proposed algorithm.

The derivation of our algorithm was inspired by other recent closed-form approaches to the PnP and GPnP problem. It is based on the insight that the PnP and GPnP problem can be interpreted as geometric 3D point-to-line registration algorithms, with the property that the orthogonal distance between points and lines is minimized. Using the state-of-the-art algorithms as starting point, we further extended the spirit of these algorithms to other geometric entities, in particular to 3D point-to-point and point-to-plane correspondences. For the 3D point-to-plane metric there exists an analogy to the PnL problem, very much like between the PnP problem and the 3D point-to-line case. Our algorithm is applicable to Euclidean as well as similarity transformations with a global

scaling parameter for the registration. Moreover, it preserves all desirable properties, in particular it is globally optimal, capable of returning all solutions in ambiguous cases, and it has a linear complexity.

The various correspondence types are represented by an orthogonal complement formulation that allows them to be used together in one single framework and even more efficiently than before. We also showed that within our formulation, existing rotation solvers (of other PnP solvers) can be re-used in a much wider context, which offered the possibility to compare them against each other on varying problem instances. It turned out that the registration using exclusively point-to-point correspondences represents a failure case for the previous ones. We additionally derived an own rotation solver which succeeds in almost all scenarios. However, our comprehensive simulations also revealed that none of the considered algorithms are completely free of singularities (including our own). A practical remedy was proposed consisting of pre-rotating reference points, but it comes at the expense of requiring multiple evaluations.

We demonstrated the broad applicability, accuracy, and numerical stability of our algorithm on various synthetic experiments including a standard evaluation framework for PnP problems. The upcoming sections show that it can also be successfully applied on real data.

3. Constrained Bundle Adjustment for Non-Rigid Registration

3.1. Introduction

The previous chapter focused on the estimation of the parameters of a single Euclidean or similarity transformation based on correspondences to known 3D data. By contrast, this chapter deals with the simultaneous determination of a *large number of Euclidean transformations* comprising a camera movement path and an *unknown 3D geometry*. Each of the Euclidean transformations represents the position and orientation of the camera relative to the reconstructed 3D structure (real model) over time. The simultaneous optimization of all structure and pose parameters from measurements of an image sequence is usually called bundle adjustment (BA) or smoothing and mapping (SAM) in robotics.

Bundle adjustment was traditionally employed as a batch method for offline applications such as the reconstruction of 3D models from internet photo collections [GSC*07, AFS*11]. Regarding simultaneous localization and mapping (SLAM), early approaches for motion and structure estimation were based on recursive filtering techniques [Dav03, ED06] until people started demonstrating the applicability of BA within real-time SLAM [DK06, KM07]. Later, it was shown that BA is even more efficient than filtering [SMD10a], because it provides more consistent estimates without necessarily being slower. State-of-the-art visual SLAM systems [MAMT15] therefore comprise two major building blocks [CCC*16]:

- A *front-end* which performs feature extraction and data association between corresponding 2D image points [GHT11, BM04] across the captured images. Further, it usually also computes initial geometric parameter estimates by employing closed-form algorithms for specific subtasks - in particular pose estimation (Chap. 2) and triangulation [HS97a] - in an interleaved manner.
- The *back-end* runs the bundle adjustment. Since the accuracy of both, motion and structure parameters, mutually depend on each other, their simultaneous optimization yields globally consistent parameter estimates which best explain the observed measurements.

However, although BA provides optimal parameter estimates in the maximum likelihood sense, it does not necessarily guarantee absolute accuracy. In fact, if the optimization problem is sparse (i.e. few observations per point), noise on the measurements can have severe effects on the outcome, resulting in low-frequency deformations of the camera path and the reconstruction of the scene, as illustrated in the synthetic toy example of Fig. 3.1. The reason for such an outcome need not necessarily be that the optimization has converged to a false, local minimum (the solutions shown in Figs. 3.1 (b) and (e) actually represent global optima with minimal reprojection error for the unconstrained BA problem). Rather, it may occur simply because in the high-dimensional parameter space the variables are highly correlated (observe in Fig. 3.1 (b) that both, cameras and points to the left and the right, are simultaneously shifted upwards), which in turn makes the location of the minimum very sensitive to the input. Problems of this kind are termed *ill-conditioned* in mathematics or statistics.

For augmented reality applications it is crucial that the reconstructed real-world model is consistent with the geometry of the virtual models. Otherwise, if the map is deformed, it will be impossible to achieve a correct

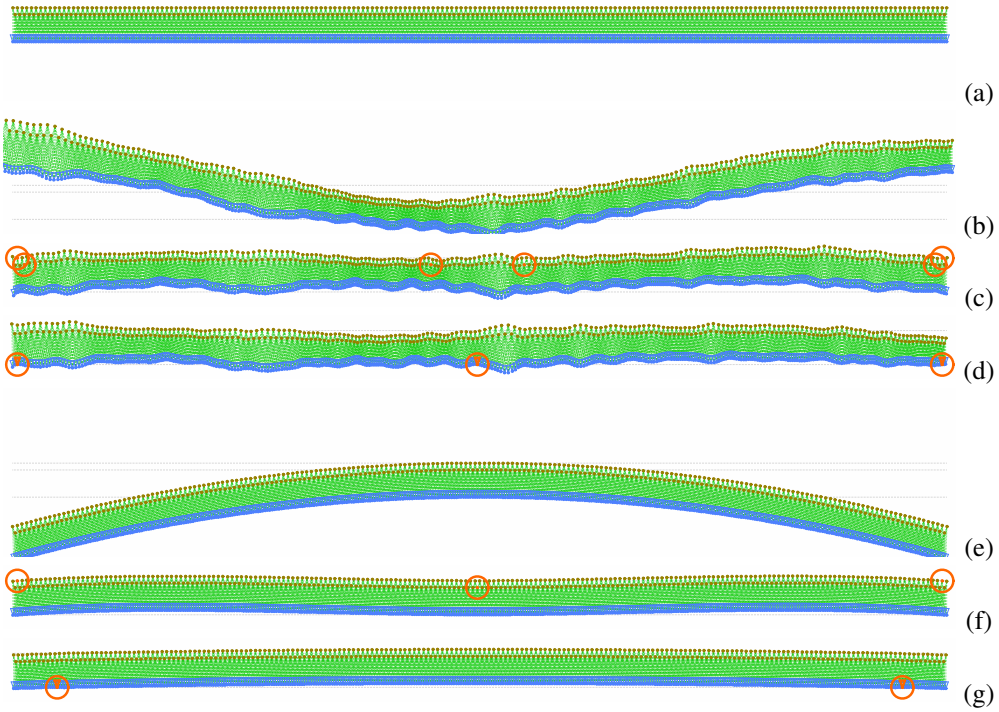


Figure 3.1.: Illustration of the constrained bundle adjustment on synthetic 2D examples. Here, all unconstrained point and camera parameters have two and three DoF, respectively (one orientation angle and a 2D translation per camera). (a) Ground-truth data with the camera path in blue, the 2D points in brown, and the 1D measurements represented by green lines connecting cameras and points. Adding random Gaussian noise (b) or systematic errors to the measurements, e.g. due to not fully corrected lens distortion as in (e), can lead to low-frequency deformations of the reconstruction and the camera path including local scale changes after unconstrained optimization. Constraining only a few points (c/f) or cameras (d/g) to their correct values (marked with red circles) can already suffice to eliminate a large portion of the estimation errors.

overlay of virtual content from all possible viewpoints. More clearly, if the virtual objects happen to appear correctly from one viewpoint they will certainly not for another.

For this purpose, we consider the internalization of knowledge about connections between the real world and the virtual model into the minimization and propose a constrained variant of BA. The goal is to achieve a better registration to the virtual domain than if it were based solely on a rigid Euclidean or similarity transformation. If, as illustrated in Fig. 3.1 (c,d,f,g), only a small number of parameters are known, this may already suffice to compensate a large portion of the deformation. Since these constraints may influence the overall shape of the camera path and the reconstruction, we label this approach *non-rigid registration*.

The traditional way for incorporating constraints is to search for stationary points of the Lagrangian (Lagrangian methods) [TMHF00, NW06, Ber99, BSS06]. Its disadvantage is, however, that the least-squares character of the problem is breached, making standard minimization methods for BA such as Gauss-Newton or

Levenberg-Marquardt [Lev44, Mar63] no longer applicable. In addition, it inflates the number of parameters due to the introduction of slack variables.

By contrast, our approach for incorporating constraints into the BA is based on the idea that any equality constraint imposed on a subset of parameters effectively reduces the dimensionality (DoF) of the parameter vector. This possibly nonlinear subspace can be interpreted as a hyper surface or manifold in parameter space. We propose a general framework for BA, with which it is possible to model constrained parameters by means of (1) their tangent spaces (linearized local approximation to the manifold), and (2) an update rule, which ensures that the parameters always remain on the constraint manifold, so that these constraints are respected at all times (strictly feasible approach). Such a generalized BA variant based on *optimization-on-manifold* techniques notably offers the following possibilities:

- *Partially constrained parameters*: The toy example in Fig. 3.1 (analogy in 2D) illustrates the case when some of the point or camera parameters are *fully known* w.r.t. all three or six DoF. In this work, we will also consider *partial knowledge*, e.g. when points are constrained to surfaces or curves in 3D space. We will exemplify the correct modeling for various special cases, including points bound to affine subspaces (planes and lines) and one DoF rotations, where the axis is fixed. This will later be relevant for markers (Chap. 4) or manually assigned feature points (Chap. 5) with partially indeterminate registration information.
- *Connected parameters*: In standard BA, the minimization variables - in particular the 3D scene points - are treated as independent parameters. We consider the case when the locations of some points may be mutually connected, e.g. when subgroups of points belong to a rigid object in the scene so that the relation between points of the same subgroup can be described by a Euclidean or similarity transform. This possibility is later exploited for the case of marker reconstruction (see Chap. 4), where each marker provides four (or more) 2D points measurements in the image, but their corresponding 3D points belong to one and the same rigid body.

As part of this work, we have implemented a flexible and generic bundle adjustment framework, which allows to combine different parameter types and to impose various constraints. We have published some of the ideas and concepts for including constraints into the bundle adjustment in 2011 [WWK11b, WWK11a]. At that time the SBA framework of Lourakis et al. [LA09] was the most advanced and most efficient publicly available bundle adjustment method. However, it was - and still is - only applicable to very specific structure-from-motion optimization problems, and therefore, it only has a limited adaptability. In particular, it is only capable of working with two different parameter types at the same time (e.g. camera extrinsics and 3D points), it is not possible to impose constraints on specific parameters, and it can only work with one measurement model for the whole problem. Due to these limitations of the SBA library, it was necessary to implement an own sparse bundle adjustment solver, with which our ideas could be realized.

Bundle adjustment in general has been an active research area [ASSS10, WACS11, AFS*11, Kon10, KRD08, KJR*11] and several other publicly available bundle adjustment frameworks have emerged in the meantime in parallel to ours [KGS*11, Del12, WACS11, AMO]. At the same time, optimization-on-manifold has gained popularity for modeling general, three-DoF rotations based on Lie group theory for efficient and singularity-free iterative minimization of rotations [SMD10b, GKS*10, WBF11, SD12] or essential matrices [MKS01]. Some of the open BA frameworks have adopted these developments and provide mechanisms for manifold-based modeling of parameters.

An important aspect of the considered constrained minimization problems is that the references to the virtual world are bound to the specific coordinate system of the virtual model. While this is not an issue if the constraints are merely used to delimit the parameters to general three-DoF rotations (a rotation remains a rotation even after a similarity transform, so the constraints remain satisfied), it becomes relevant for model-specific constraints. As we assume that no initial information about the scene was used for SLAM until spatial registration, the

coordinate system in which the initial parameters are estimated is arbitrarily chosen and predetermined by the SLAM initialization step (e.g. through Epipolar geometry [Nis04, Har97a]). In order to impose model constraints by means of an optimization-on-manifold approach in this situation, two special steps are required. These must be carried out independently of the selected sparse optimization library. First, a rigid transformation must be determined so that the reconstructed model and the camera path can be brought into the target coordinate system. For this purpose, the algorithm presented in Chap. 2 is particularly suitable, as it can also be applied to partially defined constraints or correspondences. And secondly, the constrained parameters must be projected onto their respective manifolds so that the constraints are satisfied right from the beginning. Once the feasibility of the constraints on the parameters is attained, also external libraries can be used for the actual iterative minimization on the manifolds.

The main focus in this chapter is on the essential differences to the standard BA, whereby the general idea for the specific types of parameters and constraints is made concrete in each case. The presence of existing BA libraries asks to what extent they can be reused for this purpose. We will therefore also discuss the general requirements for including partial constraints on parameters into BA by means of an optimization-on-manifold approach from a general perspective and how these open frameworks can be used for solving the constrained bundle adjustment problems considered here. The proposed approach will be the basis for the following chapters (Chap. 4 and 5). These also demonstrate the advantages of the constrained BA compared to an exclusively rigid registration on real data.

3.2. Problem Formulation and Related Work

Bundle adjustment (BA) [TMHF00, HZ04, ESN06, LA09] can be considered as a technique to *simultaneously* solve for two families of parameters within a large nonlinear minimization problem. Typically, but not necessarily, the two parameter families represent structure and motion parameters within a SLAM or SfM system. Its most widespread application is the simultaneous estimation of 3D scene points *and* the camera parameters (motion path)[1] from corresponding 2D observations in a set of images. BA aims at minimizing the sum of individual error terms, each containing the deviation between a predicted 2D point (based on the estimated parameters) and the actual 2D measurement. Since each of these error terms depends on both, motion and structure, it is clear that any technique that disregards this common dependence (e.g. by optimizing the two parameter sets in an alternating fashion), will necessarily produce suboptimal estimates or will have a slow convergence rate. By contrast, BA is able to determine the parameters optimally in a Maximum Likelihood sense.

The particularity of BA is that it addresses minimization problems that exhibit a *sparse structure*. More formally, BA is targeted towards minimization problems which are of the following type:

$$\arg\min_{\mathbf{a},\mathbf{b}}(F(\mathbf{a},\mathbf{b})) = \arg\min_{\mathbf{a},\mathbf{b}} \sum_{k=1}^{K} \rho_k(\mathbf{p}_k - f_k(\mathbf{a}_{m(k)}, \mathbf{b}_{n(k)}, \bar{\mathbf{c}}_k)), \tag{3.1}$$

where $\mathbf{a}$ and $\mathbf{b}$ refer to the two parameter families which shall be optimized and the $\mathbf{p}_k$ denote the K available measurements for the problem. Throughout this thesis, we assume that $\mathbf{p}_k \in \mathbb{R}^2$, representing 2D points in an image, e.g. obtained by feature matching or marker detections. The function $f_k(\mathbf{a},\mathbf{b},\bar{\mathbf{c}}) =: \hat{\mathbf{p}}_k$ represents the mapping from parameters $\mathbf{a}$ and $\mathbf{b}$ to a prediction on the measurement location, $\hat{\mathbf{p}}_k$, where $\bar{\mathbf{c}}$ are constant values known in advance (e.g. camera intrinsics or local marker corner coordinates). The function $\rho_k(.) : \mathbb{R}^2 \mapsto \mathbb{R}$ can be a robust metric (such as *Huber*, *Tukey* or any of the like) [Zha97], which helps to suppress the influence of outliers or to model the the distribution of measurements (if not Gaussian) more appropriately. In the further

[1] notably, the position and orientation of the camera in the scene for each recorded image.

course, we will assume that this metric is given by the squared error, $\rho_k(.) := \|.\|_2^2$, and that BA thus determines the minimum of a *least-squares* problem, $\Sigma(\mathbf{p}_k - \hat{\mathbf{p}}_k)^\mathsf{T}(\mathbf{p}_k - \hat{\mathbf{p}}_k)$ [2], or simply

$$F(\mathbf{a},\mathbf{b}) = (\mathbf{p} - f(\mathbf{a},\mathbf{b}))^\mathsf{T}(\mathbf{p} - f(\mathbf{a},\mathbf{b})), \qquad \begin{array}{ll} with & f(.) = [f_1(.)^\mathsf{T}, \cdots f_k(.)^\mathsf{T}, \cdots f_K(.)^\mathsf{T}]^\mathsf{T} \\ and & \mathbf{p} = [\mathbf{p}_1^\mathsf{T}, \cdots \mathbf{p}_k^\mathsf{T}, \cdots \mathbf{p}_K^\mathsf{T}]^\mathsf{T}. \end{array} \tag{3.2}$$

The sparsity property results from the fact that each of the individual error terms to be minimized, $\rho_k(.)$, does not depend on all optimization variables at the same time. Rather, the two parameter sets $\mathbf{a}$ and $\mathbf{b}$ can be further divided into smaller, non-overlapping subsets (blocks), $\mathbf{a} = \{\mathbf{a}_1, \cdots, \mathbf{a}_m, \cdots, \mathbf{a}_M\}$ and $\mathbf{b} = \{\mathbf{b}_1, \cdots, \mathbf{b}_n, \cdots, \mathbf{b}_N\}$, with each measurement belonging only to one subset of each family. As an example, one subset $\mathbf{a}_m$ could consist of the parameters of the camera pose, and $\mathbf{b}_n$ might be the three-dimensional coordinates of a reconstructed scene point. In this case, a prediction of a feature point location $\hat{\mathbf{p}}_k$ is only dependent on the camera pose parameters $\mathbf{a}_{m(k)}$ of the image where it was observed and on the 3D parameters of the scene point $\mathbf{b}_{n(k)}$ it belongs to. But it is not a function of other scene points or other poses corresponding to other images. In the context of this thesis, we also use BA for two other problems, namely marker-based reconstruction as well as monocular camera calibration (see Table 3.1 for an overview and the mapping of parameters), which also have the same sparsity structure.

3.2.1. Exploiting Sparsity and Efficient Computation

BA as a technique aims at exploiting the sparsity to the best extent in order to obtain an optimal solution within a feasible amount of time despite the large number of variables to be estimated.

Typically, Levenberg-Marquardt [Lev44, Mar63] is used as standard technique to find a (local) minimum to the objective function of Eq. 3.1, which can be seen as a method that combines the fast quadratic convergence property of Gauss-Newton with the robustness of Steepest Gradient methods. This is achieved by iteratively solving the set of the *augmented normal equations* to obtain parameter increments $\Delta\mathbf{a}$ and $\Delta\mathbf{b}$,

$$(\mathbf{J}^\mathsf{T}\mathbf{J} + \mu\mathbf{I}) \begin{bmatrix} \Delta\mathbf{a} \\ \Delta\mathbf{b} \end{bmatrix} = \mathbf{J}^\mathsf{T}(\mathbf{p} - f(\mathbf{a},\mathbf{b})), \tag{3.3}$$

where $\mathbf{J}$ denotes the Jacobian of $f(\mathbf{a},\mathbf{b})$ comprising the partial derivatives of all $f_k(.)$ with regard to all parameters $\mathbf{a}$ and $\mathbf{b}$. In order to obtain the solution to the increments, it is essentially necessary to invert the *approximated Hessian*, $\mathbf{J}^\mathsf{T}\mathbf{J} + \mu\mathbf{I}$, which, when using dense-matrix inversion algorithms, requires cubic complexity in terms of the size of this matrix. The actual computation of the derivatives and the inversion are the most resource intensive parts of BA.

As described above, the individual error terms $\mathbf{p}_k - f_k(.)$ are independent of the majority of parameters. As a consequence, $\mathbf{J}$ and $\mathbf{J}^\mathsf{T}\mathbf{J} + \mu\mathbf{I}$ are mostly filled with zeros, even if all theoretically possible measurements were observed. This is often called the *primary sparsity* structure of BA. It is prevalent in any BA problem and is known in advance. It can be exploited by solving for one parameter family first via the Schur complement

[2] We make this simplification only for the sake of clarity in the exposition. In literature, robust metrics are often included into BA as an iteratively re-weighted least-squares algorithm. Alternatives including an own 'lifting technique' are presented by Zach [Zac14]. Another option - which we chose - is to determine a mapping $\sigma(.) : \mathbb{R}^2 \mapsto \mathbb{R}^2$ such that $\rho(\mathbf{x}) = \sigma(\mathbf{x})^\mathsf{T}\sigma(\mathbf{x})$ and $\sigma(-\mathbf{x}) = -\sigma(\mathbf{x})$. Then, the problem becomes least-squares, again, which can be seen, when it is rewritten as: $\Sigma(\mathbf{0} - \sigma_k(\mathbf{p}_k - \hat{\mathbf{p}}_k))^\mathsf{T}(\mathbf{0} - \sigma_k(\mathbf{p}_k - \hat{\mathbf{p}}_k))$. Here, sigma has taken on the role of the prediction function $f_k(.)$, which is now compared against zero-valued 'virtual' measurements. In practice we have always obtained good results with the *Pseudo-Huber metric* [HZ04] regarding outlier suppression and convergence behavior.

Table 3.1.: Mapping of parameter types for some ordinary bundle adjustment problems without explicit model constraints. The subscripts $\mathcal{C}_m$, $\mathcal{L}_n$, and $\mathcal{W}$ denote the coordinate system of the m-th camera, the local coordinate system of the n-th marker or object, and the coordinate system of reconstructed SLAM model of the real world, respectively.

	$\mathbf{a}_m$	$\mathbf{b}_n$	$\bar{\mathbf{c}}_{mn}$
SfM-BA	(camera extrinsics) $\mathbf{R}_{m,(\mathcal{W}\to\mathcal{C}_m)} \in SO(3)$ $\mathbf{t}_{m,(\mathcal{W}\to\mathcal{C}_m)} \in \mathbb{R}^3$	(scene points) $\mathbf{x}_{n,(\mathcal{W})} \in \mathbb{R}^3$	(camera intrinsics) f_x, f_y, c_x, c_y
Marker-BA (fixed scale)	(camera extrinsics) $\mathbf{R}_{m,(\mathcal{W}\to\mathcal{C}_m)} \in SO(3)$ $\mathbf{t}_{m,(\mathcal{W}\to\mathcal{C}_m)} \in \mathbb{R}^3$	(Euclidean transform) $\mathbf{R}_{n,(\mathcal{L}_n\to\mathcal{W})} \in SO(3)$ $\mathbf{t}_{n,(\mathcal{L}_n\to\mathcal{W})} \in \mathbb{R}^3$	(camera intrinsics) f_x, f_y, c_x, c_y (local marker points) $\mathbf{x}_{ln,(\mathcal{L}_n)} \in \mathbb{R}^3, \quad l \in [1 \cdots 4]$
Marker-BA (variable scale)	(camera extrinsics) $\mathbf{R}_{m,(\mathcal{W}\to\mathcal{C}_m)} \in SO(3)$ $\mathbf{t}_{m,(\mathcal{W}\to\mathcal{C}_m)} \in \mathbb{R}^3$	(similarity transform) $\mathbf{R}_{n,(\mathcal{L}_n\to\mathcal{W})} \in SO(3)$ $\mathbf{t}_{n,(\mathcal{L}_n\to\mathcal{W})} \in \mathbb{R}^3$ $s_{n,(\mathcal{L}_n\to\mathcal{W})} \in \mathbb{R}_+$	(camera intrinsics) f_x, f_y, c_x, c_y (local marker points) $\mathbf{x}_{ln,(\mathcal{L}_n)} \in \mathbb{R}^3, \quad l \in [1 \cdots 4]$
Camera Calibration	(camera extrinsics) $\mathbf{R}_{m,(\mathcal{L}\to\mathcal{C}_m)} \in SO(3)$ $\mathbf{t}_{m,(\mathcal{L}\to\mathcal{C}_m)} \in \mathbb{R}^3$	(pinhole camera param.) f_x, f_y, c_x, c_y (distortion coefficients) d_1, d_2, d_3, d_4, d_5	(calibration pattern points) $\mathbf{x}_{l,(\mathcal{L})} \in \mathbb{R}^3, \quad l \in [1 \cdots L]$

and then for the other by back-substitution, which reduces the effort to linear complexity with regard to one parameter set and cubic complexity for the other (*'Schur complement trick'*). The SBA library of Lourakis and Argyros [LA09] was the first publicly available BA implementation BA that included this Schur complement based variable elimination scheme.

Secondary sparsity refers to the case, when not all measurements have been observed, e.g. when the camera travels along corridors and features are only visible for a short period of time. As a consequence, additional parts of the Jacobian remain empty, even though they belong to valid combinations between parameters and measurements. The secondary sparsity varies from case to case and requires an adaptive handling. A possible strategy consists of reordering the parameters, which corresponds to swapping columns in the Jacobian $\mathbf{J}$. In the ideal case (depending on the problem input) the approximated Hessian, $\mathbf{J}^\mathsf{T}\mathbf{J}$, admits a banded-like structure, having only few entries far apart from the matrix diagonal, so that the full inversion only requires linear effort. The problem of finding an optimal, fill-reducing reordering is NP-complete [Yan81], but heuristics exist to obtain good ordering results [DGLN04, Dav06b] (COLAMD), or to directly perform matrix decomposition [CDH08] (CHOLMOD), [Dav06a] (CSparse). Their use within SLAM has been demonstrated by Konolige [Kon10] and Dellaert and Kaess [DK06].

Instead of inverting the approximated Hessian of Eq. 3.3, it is also possible to consider

$$\mathbf{J}\begin{bmatrix}\Delta\mathbf{a}\\ \Delta\mathbf{b}\end{bmatrix} = (\mathbf{p} - f(\mathbf{a},\mathbf{b})), \tag{3.4}$$

and to directly solve for the increments via QR decomposition of $\mathbf{J}$. This is particularly useful in incremental SLAM, when the problem size continuously increases due to new measurements and variables [KRD08, KJR*11, PIS*13b]. For new frames the Jacobian is only partially re-evaluated or extended for the newly created variables, which allows to re-use the QR decomposition of the previous frame and update it by means of a few Givens rotations. Also, an explicit computation of $\mathbf{J}^\mathsf{T}\mathbf{J}$ can be avoided. Essentially this strategy boils down to only one GN iteration per frame. However, after frequent recurring intervals, full evaluation and multiple iterations are necessary in order to account for changes in the linearization point. In general, QR decompositions of $\mathbf{J}$ are approximately three times slower than a full Cholesky or LDL-based inversion of $\mathbf{J}^\mathsf{T}\mathbf{J}$.

Another performance gain can be achieved from the observation that the Jacobian and the Hessian both have a block-wise structure. Larger blocks corresponding to parameter subsets, $\mathbf{a}_m$ or $\mathbf{b}_n$, and measurements, $\mathbf{p}_k$, are either fully dense or contain only zeros. Thus, it is beneficial to perform matrix manipulation on blocks instead of single elements. Polok et al. [PIS13a] proposed a BLAS-like library for manipulation of block-wise sparse matrices for nonlinear least-squares problems [PSI*13] and SLAM [IPSS17]. With a block-wise approach also better results are achieved for fill-reducing variable reordering [DK06].

Several alternatives to Levenberg-Marquardt or Gauss-Newton have been proposed. Some authors argue that Powell's Dog-Leg algorithm is better suited to BA problems [LA05, RKL14]. Another popular strategy is to use conjugated gradients (CG) for solving Eq. 3.3 of large problems in order to avoid a full matrix inversion or QR decomposition [BÅ09, BÅ10, ASS*09, ASSS10]. This is particularly useful when making use of graphics hardware or other parallel architectures [WACS11], because the matrix inversion cannot be well parallelized. For the convergence speed of CG, a low condition number of $\mathbf{J}^\mathsf{T}\mathbf{J}$ is needed, which therefore requires a preconditioning of the matrix. This may often not be enough for problems with low connectivity (secondary sparsity), so convergence can often be much slower than in a full-inversion based approach [BÅ10].

Regarding our implementation of BA, we confined ourselves to exploiting the primary sparsity structure within Levenberg-Marquardt iterations. We will therefore briefly review the Schur complement trick together with other implementation aspects of our approach in Sec. 3.6.3.

3.2.2. Constrained Minimization

In this work, we are interested in internalizing information about the coordinate system of the virtual domain inside BA. Therefore, we consider a constrained variant of BA and the objective function of Eq. 3.1 modifies to:

$$\arg\min_{\mathbf{a},\mathbf{b}}(F(\mathbf{a},\mathbf{b})) \qquad s.t. \quad \mathbf{a}_m \in \mathcal{M}_{\mathbf{a}_m}, \mathbf{b}_n \in \mathcal{M}_{\mathbf{b}_n}, \tag{3.5}$$

where the right-hand side of Eq. 3.5 refers to the constraints imposed on the parameters, and $\mathcal{M}_{\mathbf{a}_m}$ and $\mathcal{M}_{\mathbf{b}_n}$ denote the feasible set for each of the parameter subsets $\mathbf{a}_m$ and $\mathbf{b}_n$, respectively. A possible way of expressing constraints mathematically can be via a set of homogeneous equalities

$$g_i(\mathbf{a},\mathbf{b}) = 0, \tag{3.6}$$

so that the feasible sets are given by $\{\mathcal{M}_\mathbf{a}, \mathcal{M}_\mathbf{b}\} = \{\mathbf{a},\mathbf{b} \,|\, g_i(\mathbf{a},\mathbf{b}) = 0 \;\; \forall i\}$.

In standard textbooks on continuous numerical optimization [NW06, Ber99, BSS06] little can be found on how constraints can be integrated efficiently into sparse problems like BA. However, they provide an overview upon general strategies for embedding equality constraints (and also bound i.e. inequality constraints) into minimization problems. Among these the most popular are *penalty*, *Lagrangian* and *strictly feasible* methods. Approaches in computer vision and related fields that incorporate constraints in sparse BA problems can be categorized accordingly.

Penalty method: In the *penalty method* the objective function is extended by an additive, weighted squared penalization term that is larger than zero whenever the constraints are not satisfied, e.g.

$$\arg\min_{\mathbf{a},\mathbf{b}} \;\; F(\mathbf{a},\mathbf{b}) + \mu \sum_i \|g_i(\mathbf{a},\mathbf{b})\|_2^2 \tag{3.7}$$

Thus, the problem is converted into an unconstrained least-squares problem, again.

In computer vision literature, penalties are often used to model 'soft constraints' and μ is used to experimentally tune their influence on the final output. For example, Fua [Fua99] uses BA to fit a generic mesh-based face model to real faces captured in three images. Based on an initial image alignment of this model, they project the vertex points of the mesh into the images and reconstruct them with BA. A penalty term ('stiffness' regularization) is added to the objective function which measures the deviations between the reconstructed points and the mesh vertices to ensure a good alignment and to prevent excessive deformation. Sato et al. [SIY04, SKYT02] propose a SfM approach for reconstructing outdoor scenes with omni-directional or handheld cameras and incorporate knowledge on the position of known 'landmark points' (e.g. marker positions) in the scene. In their BA approach the motion and structure parameters are optimized based on the re-projection error of both, landmark and unknown feature points in the scene. Measurements in frames with observed landmarks receive a higher penalty than those containing only unknown parts of the environment. Lhuillier [Lhu11, Lhu12] discusses several variants for incorporating noisy GPS measurements inside a vision-based SfM-BA. In one of them, the squared error between the GPS position and the estimated camera position is added as penalty function to the standard BA problem. In another variant the author in fact minimizes this positional error and a penalty term comprises a reciprocal function of the image residual error, $g_i(\mathbf{a},\mathbf{b}) = 1/(e_{th} - F(\mathbf{a},\mathbf{b}))$, which sharply increases, as soon as the image residuals begin to increase. Inspired by this, Larnaout et al. [LGBBD16] and Salehi et al. [SGBBC16] use the same penalization inside a SLAM system for vehicle navigation in urban environments. In an older publication, Szeliski [ST98] used the penalty method inside BA for enforcing coplanarity ('hard') constraints in order

to reconstruct piece-wise planar scenes. In their approach they alternately solve three steps: point-to-plane assignment, plane regression (via SVD), and a single (gradient-based) BA iteration with penalization. The penalty factor μ for the squared error of coplanarity equations is fixed to a very large value.

As a disadvantage, penalty methods are not well-suited for representing hard constraints that require an exact fulfillment. This can never be achieved unless the weighting factor is set to infinity, which cannot be done in practice.

Lagrangian methods: Lagrangian methods attempt find stationary points of the Lagrangian, $L(\mathbf{a}, \mathbf{b}, \boldsymbol{\lambda})$, which is constructed by multiplying each constraint equation, g_i, with a slack variable, λ_i (Lagrange multipliers), and by adding these terms to the objective function $F(\mathbf{a}, \mathbf{b})$:

$$L(\mathbf{a}, \mathbf{b}, \boldsymbol{\lambda}) = F(\mathbf{a}, \mathbf{b}) - \sum_i \lambda_i \cdot g_i(\mathbf{a}, \mathbf{b}). \tag{3.8}$$

The necessary optimality conditions, which represent stationary points of the Lagrangian, are given by:

$$\frac{\partial}{\partial \mathbf{a}, \mathbf{b}, \boldsymbol{\lambda}} L(\mathbf{a}, \mathbf{b}, \boldsymbol{\lambda}) = \mathbf{0}. \tag{3.9}$$

In his well-known reference on bundle adjustment, Triggs [TMHF00] promoted the Lagrange method as a tool for modeling constraints in BA. Börlin [BLE03] elaborated this method further and exemplified it on a synthetic BA problem where the points are bound to the inner side of a sphere ('Dome'). The Lagrange method was also used by Shi et al. [SSSZ17] for simultaneous scene reconstruction and camera calibration of multi-view stereo rigs.

The major drawback of Lagrangian methods is that they do not correspond to a minimization of a least-squares error function, anymore. As a consequence, Gauss-Newton or Levenberg-Marquardt are no longer applicable and other algorithms, such as the augmented Lagrangian method or Sequential Quadratic Programming (SQP), must be employed. These require the full evaluation of the Hessian (second order derivatives) of the Lagrangian, $L(\mathbf{a}, \mathbf{b}, \boldsymbol{\lambda})$, or of the objective function, $F(\mathbf{a}, \mathbf{b})$, in each iteration. Another disadvantage is that the number of unknowns is actually increased due to the introduction of the Lagrangian multipliers λ_i, and not decreased[3], which inflates the problem size.

Strictly feasible or direct approaches: In contrast to Lagrangian methods, direct approaches do not use additional slack variables. Rather, they attempt to incorporate the constraints $g_i(\mathbf{a}, \mathbf{b})$ directly into the objective function (e.g. by algebraic substitution), thereby reducing the dimensionality of the parameters and converting the minimization problem into an unconstrained one. If the constraints are nonlinear and a closed-form substitution is not possible, then a linearization (tangent spaces) of the constraints around the current parameter estimates is performed. After each iteration the feasibility of the variables is enforced. The Generalized Reduced Gradient (GRG) method or Gradient Projection techniques [LWJR78, FLWW98, SL92] represent some of the early approaches that follow this idea. In GRG methods typically steepest gradient methods are used for minimization.

Apart from variable substitution, another common instance of direct approaches consists of parameterizing the variables directly along their feasible sets, e.g. by using local coordinates for points or camera positions

[3] As an example, consider that quaternions, $\mathbf{q}$, were used for representing rotation parameters. For the quaternion to be a rotation, the unit norm must be preserved, $\mathbf{q}^\mathsf{T}\mathbf{q} - 1 = 0$. One could add equations of this kind as constraints to the minimization problem. In this case, however, one would finally end up with five parameters per rotation instead of three, because for each constraint one additional Lagrange multiplier must be added.

on planes in 3D space [TGBC*11, LBGBD12], so that they respect the constraints at all times. Tamaazousti et al. [TGBC*11] present a real-time SLAM approach with constraints to a CAD Model. They reconstruct both, features in the environment and on the real object belonging to the model. While the environment features remain free, the model features are only allowed to change their coordinates along the surface of the model mesh. In this way, they maintain the SLAM aligned with the CAD model and avoid drifting. The same idea is used for online vehicle geo-localization in urban environments using GIS models [LBGBD12]. Façade features are bound to vertical planes corresponding to building walls in the city model and the camera position is constrained to a virtual plane at known height from ground level. The latter approach was later extended to also include GPS measurements [LGBBD16, SGBBC16] (see above). Shan [SLZ01] proposed a BA-based approach to align a face model to real faces in images. In contrast to Fua's method [Fua99] (see above), Shan's face model is parameterized consisting of a neutral (mean) face and 65 face parameters, which may change the vertex locations of the mesh. Reconstructed points may either evolve along the model's surface or are bound to semantic points (lip corners, eyes). After some algebraic manipulations the author succeeds in internalizing all parameters, including the camera poses, point positions and face parameters, in one least-squares problem.

Optimization-on-manifold (OOM) and Lie algebraic methods [AMS07, RW12] can be considered as another, modern variant of strictly feasible methods, especially for nonlinear parameters. It is often used for representing matrix groups with special group properties, such as rotation matrices, essential matrices [HHLM07] or homographies [EvdH09]. These matrix groups can be interpreted as manifolds in a higher dimensional space. Lie group theory connects operations on local tangent spaces (*Lie algebra*) of the manifolds with the elements of the group via the exponential map. Iterative minimization algorithms such as Gauss-Newton or Levenberg-Marquardt constantly perform linearizations around current parameter estimates, i.e. they require the knowledge of their tangent spaces, which is the reason, why Lie algebra is a suitable and valuable tool for these algorithms

Rotations in $\mathbb{R}^3$ represent a good example of the usefulness of Lie algebra. From the fact that rotation matrices have nine entries, but only three DoF, it can be seen that the set of all rotations forms a three-dimensional manifold, $\mathcal{M}_{SO(3)}$, in the higher nine-dimensional space. This manifold (or feasible set) can be described using the orthogonality property of rotation matrices, i.e. $\mathbf{R}^\mathsf{T}\mathbf{R} = \mathbf{I}$, which yields six independent quadratic equations on the nine elements. However, using these as constraints inside the minimization (e.g. by means of the Lagrangian method) is not a practical approach. Another possibility is to describe rotation parameters via a global and minimal (i.e. 3 DoF) parameterizations, as done in earlier implementations of BA. Common choices in BA include Euler angles [ESN06], axis-angle [ASSS10], or imaginary parts of a quaternion [LA09]. But global, minimal parameterizations regularly exhibit degeneracies (e.g. 'Gimbal locks' for Euler angles), which may lead to convergence failures during minimization. Optimizing the nine entries of a rotation matrix directly or using other over-parameterized representations such as quaternions, is also not a good idea, because the resulting 'optimal' parameters will almost certainly not be a rotation, but some arbitrary linear transformation (or a scaled rotation in case of quaternions). These problems are overcome in a natural way, when using Lie algebra in an OOM based approach. From a practical point of view, OOM consists of a dual parameterization: a local and minimal parameterization for computing the derivatives and the parameter increments along the tangent spaces and an over-parameterized representation (e.g. rotation matrices) for storing the results between the iterations. Both parametrizations are connected with an update rule (Rodrigues' formula for rotations or the exponential map in general) that transforms the parameter increments from local tangent space ('charts') to elements of the rotation group along geodesics on their manifold. In this way, singularities are avoided and the constraints for the rotation (feasibility) are respected at all times.

Szeliski [ST98] was probably the first who used Lie algebra for rotation groups [TK94] inside BA, but at that time not within an efficient BA implementation that simultaneously solved for all parameters while exploiting the problem sparsity (i.e. structure and motion parameters were solved repeatedly in an alternating fashion).

Drummond and Cipolla [DC02, DC00] promoted to use Lie algebra inside an EKF based approach for tracking using a line-based model of an object. Sarkis and Diepold [SD12] present a pose estimation method using Lie algebra for second order minimization (Newton algorithm). Today, it is a widely used technique that is also found in modern SLAM implementations [IPSS17, ESC14, MAMT15, KM07].

Within this chapter, we use this technique not only for efficient parametrization of rotations, but rather extend its main idea to also include other constraints tied to a fixed external coordinate system within a general framework.

3.3. Approach Overview

In our work, we aim at internalizing knowledge about the virtual domain as constraints within an efficient BA implementation. Our approach for constrained BA is also based on an optimization-on-manifold strategy. But in contrast to existing approaches, we do not merely use it to preserve the intrinsic, nonlinear parameter structure of rotations (or other groups). Rather, we also integrate constraints which are bound to an externally given coordinate system. In particular, we consider the following types of constraints:

1. *Full knowledge* of some of the parameters defined in the target coordinate system.
2. Points or translations bound to *affine subspaces*, such as lines or planes in the virtual domain (partial parameter information).
3. *Axis constrained rotations* (1 DoF), where the axis in known in the target coordinate system (partial parameter information).
4. *Free 3 DoF rotations* (classical case).

An overview of the full algorithm is given in Alg. 3.3.1.

As we are proposing an OOM approach also for the externally enforced (explicit) constraints (case 1.-3.), all types be handled in an equal manner during the actual Levenberg-Marquardt iterations of the BA. In all four cases we interpret the constraints as Riemannian manifolds, $\mathcal{M}_{\mathbf{a}_m}$ and $\mathcal{M}_{\mathbf{b}_n}$, which are embedded in the higher dimensional space of the original, over-parametrized parameter subsets $\mathbf{a}_m$ and $\mathbf{b}_n$ (see Fig. 3.2). In each iteration we identify the tangent spaces of these constraint manifolds, $\mathcal{T}_{\mathbf{a}}\mathcal{M}_{\mathbf{a}}$, $\mathcal{T}_{\mathbf{b}}\mathcal{M}_{\mathbf{b}}$, for the current estimates of the parameters. The derivatives and parameter increments are computed for a set of local parameters, $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$, which represent coordinates along orthogonal bases $\mathbf{V}_{\mathbf{a}}$ and $\mathbf{V}_{\mathbf{b}}$ of the tangent spaces (L.6 in Alg. 3.3.1). The actual parameter increments are obtained by means of an implicit inversion of the Hessian using the Schur complement and LDL decomposition (L.7). Finally, the original over-parameterized variables are updated along geodesics on the constraint manifold using the increments of the local parameters, $\Delta\boldsymbol{\alpha}$ and $\Delta\boldsymbol{\beta}$ (L.8).

The introduction of constraints tied to an external coordinate system requires two extra steps. First, the parameters have to be transformed with a similarity transformation (L.2), because the constraints are defined in the virtual coordinate system (different to the SLAM). Otherwise the algorithm may converge to a wrong local minimum. To this end, the computation of the parameters can be achieved using the algorithm of Chap. (2) (L.1). Since our proposed approach is 'strictly feasible', which means that the constraints are fulfilled at all times during the LM iterations (as opposed to penalty and Lagrangian methods), it is also necessary to ensure this right from the beginning. The transformation of the parameters alone does not generally guarantee this. Therefore, as a second preliminary step, the parameters need to be projected onto their constraint manifolds (L.3). This increases the error for measurements associated with these constraint. During LM iterations they are distributed across all other parameters, which results in a uniform adaptation of the whole BA problem to the enforced external knowledge.

Algorithm 3.3.1 Levenberg-Marquardt / Gauss-Newton variant for optimization on constraint manifolds

Input:

$\mathbf{a}_{\mathcal{W}}, \mathbf{b}_{\mathcal{W}}$ ▷ parameters in SLAM-world ($\mathcal{W}$) coordinate system

$\mathcal{M}_{\mathbf{a},(\mathcal{V})}, \mathcal{M}_{\mathbf{b},(\mathcal{V})}$ ▷ constraints on the parameters in the coordinate system of the virtual model, $\mathcal{V}$.

μ^0 ▷ initial damping factor for Levenberg-Marquardt iterations ($\mu^0 = 0$ for Gauss-Newton)

Output:

$\mathbf{a}_{\mathcal{V}}, \mathbf{b}_{\mathcal{V}}$ ▷ parameters in virtual coordinates $\mathcal{V}$, optimized w.r.t. the constraints.

▷ Compute a rigid transform (Euclidean or similarity) from a subset of parameters and their associated feasible sets (explicit constraints):

$\mathbf{S}_{\mathcal{W}\to\mathcal{V}} \leftarrow$ COMPUTERIGIDTRANSFORM$(\mathbf{a}_{\mathcal{W}}, \mathbf{b}_{\mathcal{W}}, \mathcal{M}_{\mathbf{a},(\mathcal{V})}, \mathcal{M}_{\mathbf{b},(\mathcal{V})})$

▷ Apply the rigid transform on all parameters:

$\begin{bmatrix}\tilde{\mathbf{a}}_{\mathcal{V}} \\ \tilde{\mathbf{b}}_{\mathcal{V}}\end{bmatrix} \leftarrow \mathbf{S}_{\mathcal{W}\to\mathcal{V}} \begin{bmatrix}\mathbf{a}_{\mathcal{W}} \\ \mathbf{b}_{\mathcal{W}}\end{bmatrix}$

▷ Project the constrained parameters onto their associated feasible subset (retraction):

$\mathbf{a}^{(0)}, \mathbf{b}^{(0)} \leftarrow$ RETRACTPARAMETERS$(\tilde{\mathbf{a}}_{\mathcal{V}}, \tilde{\mathbf{b}}_{\mathcal{V}}, \mathcal{M}_{\mathbf{a},(\mathcal{V})}, \mathcal{M}_{\mathbf{b},(\mathcal{V})})$

▷ Do the Levenberg-Marquardt iterations:

$i \leftarrow 0$

repeat

▷ Compute Jacobians w.r.t. a local parametrization $\boldsymbol{\alpha}, \boldsymbol{\beta}$ on the constraint manifold (tangent space):

$\begin{bmatrix}\mathbf{J}_{\boldsymbol{\alpha}} & \mathbf{J}_{\boldsymbol{\beta}}\end{bmatrix} \leftarrow \frac{\partial}{\partial \boldsymbol{\alpha}, \boldsymbol{\beta}} f(\mathbf{a}^{(i)} + \mathbf{V}_{\mathbf{a}^{(i)}} \cdot \boldsymbol{\alpha}, \quad \mathbf{b}^{(i)} + \mathbf{V}_{\mathbf{b}^{(i)}} \cdot \boldsymbol{\beta}) \mid_{\boldsymbol{\alpha}, \boldsymbol{\beta} = \mathbf{0}}$

▷ Compute local parameter increments:

$\begin{bmatrix}\Delta\boldsymbol{\alpha} \\ \Delta\boldsymbol{\beta}\end{bmatrix} \leftarrow \left(\begin{bmatrix}\mathbf{J}_{\boldsymbol{\alpha}}^{\mathsf{T}} \\ \mathbf{J}_{\boldsymbol{\beta}}^{\mathsf{T}}\end{bmatrix} \begin{bmatrix}\mathbf{J}_{\boldsymbol{\alpha}} & \mathbf{J}_{\boldsymbol{\beta}}\end{bmatrix} + \mu^{(i)} \mathbf{I} \right)^{-1} \begin{bmatrix}\mathbf{J}_{\boldsymbol{\alpha}}^{\mathsf{T}} \\ \mathbf{J}_{\boldsymbol{\beta}}^{\mathsf{T}}\end{bmatrix} \left[\mathbf{p} - f(\mathbf{a}^{(i)}, \mathbf{b}^{(i)})\right]$

▷ Update the parameters along geodesics on the manifolds to ensure strict feasibility:

$\begin{bmatrix}\mathbf{a}^{(i+1)} \\ \mathbf{b}^{(i+1)}\end{bmatrix} \leftarrow$ GEODESICUPDATE$\left(\begin{bmatrix}\mathbf{a}^{(i)} \\ \mathbf{b}^{(i)}\end{bmatrix}, \{\mathcal{M}_{\mathbf{a}}, \mathcal{M}_{\mathbf{b}}\}, \begin{bmatrix}\Delta\boldsymbol{\alpha} \\ \Delta\boldsymbol{\beta}\end{bmatrix}\right)$

▷ Check if update was successful:

if $\|\mathbf{p} - f(\mathbf{a}^{(i+1)}, \mathbf{b}^{(i+1)})\| < \|\mathbf{p} - f(\mathbf{a}^{(i)}, \mathbf{b}^{(i)})\|$ **then**

$\mu^{(i+1)} \leftarrow \mu^{(i)}/3$ ▷ decrease damping factor (typically $1/3$)

else

$\begin{bmatrix}\mathbf{a}^{(i+1)} \\ \mathbf{b}^{(i+1)}\end{bmatrix} \leftarrow \begin{bmatrix}\mathbf{a}^{(i)} \\ \mathbf{b}^{(i)}\end{bmatrix}$ ▷ undo update and increase damping factor

$\mu^{(i+1)} \leftarrow 2\mu^{(i)}$

end if

$i \leftarrow i + 1$

until $convergence$

$\begin{bmatrix}\mathbf{a}_{\mathcal{V}} \\ \mathbf{b}_{\mathcal{V}}\end{bmatrix} \leftarrow \begin{bmatrix}\mathbf{a}^{(i)} \\ \mathbf{b}^{(i)}\end{bmatrix}$

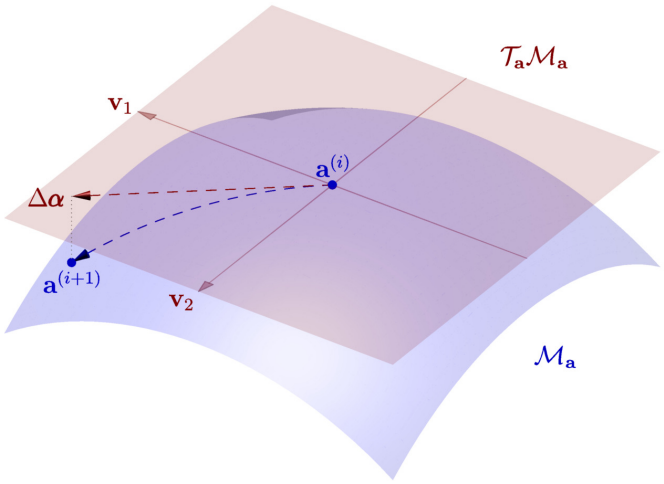


Figure 3.2.: Optimization on a manifold. The global parameters, $\mathbf{a}_{(i)}$, at iteration i and the feasible set (manifold), $\mathcal{M}_{\mathbf{a}}$, induced by the constraints are shown in blue. The local parameter increments $\Delta\boldsymbol{\alpha}$ are computed as elements of the tangent space, $\mathcal{M}_{\mathbf{a}}\mathcal{T}_{\mathbf{a}}$ (red), represented by a subspace basis, $\mathbf{V} = [\mathbf{v}_1, \mathbf{v}_2]$. The constrained parameter values for the next iteration, $\mathbf{a}_{(i+1)}$, are updated on geodesics of the manifold along the direction of $\Delta\boldsymbol{\alpha}$.

In the practical use-cases (as later described in Chaps. 4 and 5) the imposed constraints on the parameters can also involve a combination of the four types given above. For example, the knowledge that a marker which must lie on a planar surface corresponding to the virtual model may simultaneously affect the translation (plane constraint) as well as the rotation, i.e. the rotation axis is always parallel to the plane normal.

We have implemented this BA variant and evaluated in the use-cases of the chapters that follow. Due to the fact, that Lie algebra and has gained a high level of popularity for parameterizing rotations and other groups, several existing BA libraries - which have appeared in parallel to ours - also provide mechanisms to represent such an OOM approach. We will formulate general requirements and analyze to what extend they are fulfilled in these implementations.

3.4. Gauge Freedom and Coarse Initial Alignment

Iterative minimization approaches usually require an initial estimate of the parameters, which should be close enough to the global minimum. This is especially important for large minimization problems such as the bundle adjustment problems considered here, since the large amount of nonlinear parameters also results in a large number of local minima.

In Chapters 4 and 5 we will describe how an initial reconstruction is obtained for marker-based and feature-based SLAM, respectively. In both cases, the initial parameter estimates are obtained without assuming any prior information on the scene. In particular, no assumption on the availability of user knowledge for the registration is made at the beginning, either because the user provides his knowledge only later at the end of a preparative stage (feature-based) or because it is necessary to wait until observations for a minimal number of parameters with associated constraints are available (marker-based).

This lack of pre-knowledge leaves some arbitrariness on the choice of a coordinate system. But as the numerical estimation of points or other parameters in space requires to use coordinates for their representation, one has to deliberately decide for one specific coordinate system to work with. For instance, in feature-based SLAM usually the location of the first camera is chosen as the origin, but any other choice would be equally valid for the unconstrained minimization problem. This arbitrariness in reconstruction problems is long known, both by researchers from photogrammetry [Der94] and computer vision [TMHF00], and is usually termed *gauge freedom* or *datum problem*.

A consequence of the gauge freedom is that the explicit constraints given by the user cannot be included into the minimization algorithm directly. They are described in the coordinate system of the virtual model which may be considerably different to the one temporarily used by the SLAM algorithm for initial reconstruction. Even if the parameters are initially estimated with high accuracy in the unconstrained case, their final optimal values of the constrained problem may be completely different due to the fact that the coordinate systems do not match. Thus, a direct inclusion may result in convergence to a wrong local minimum or in a complete failure of the minimization algorithm.

It is therefore necessary to reconcile both coordinate systems, first. To this end, a similarity (or Euclidean) transformation, $S_{\mathcal{W}\to\mathcal{V}}$, is computed that connects both coordinate systems. With this similarity transformation all other parameters are also transformed into the coordinate system of the virtual model. For the estimation we use correspondences established from a subset of parameters for which constraints exist. Specifically, we use information on the linear components in the structure parameters $\mathbf{b}_n$, i.e. scene points $\mathbf{x}_n$ (features) or translation vectors $\mathbf{t}_n$ (markers). We assume that the constraints for these parameter subsets may be specified completely as target or anchor points or partially in the form of affine subspaces such as lines or planes in Euclidean space. Then, for the estimation of the similarity transform the closed-form algorithms of Chap. 2 can be used. Their application is use-case dependent and will be presented in more detail in Chapters 4 and 5.

For the moment we will assume that a similarity transform, $S_{\mathcal{W}\to\mathcal{V}}$, has been computed, which connects both coordinate systems, $\mathcal{W}$ and $\mathcal{V}$. In order to correctly apply this transformation to the optimization variables, $\mathbf{a}_m, \mathbf{b}_n$, it is important to recognize that some parameters represent points in $\mathcal{W}$ (such as the scene points $x_\mathcal{W}$ in SfM-BA), whereas others represent transformation parameters between $\mathcal{W}$ and some other local coordinate systems ($\mathcal{C}_m$ or $\mathcal{L}_n$). The subscript notation introduced at beginning of this thesis in Section *Mathematical Notation* becomes a useful tool at this point, as it helps to identify all involved coordinate systems, and it reveals the relations between them. The transformation of the motion and structure parameters of Table 3.1 can be summarized as follows.

- *Scene points of SfM-BA, $\mathbf{x}_n$, (structure parameters).* These represent the simplest case. In order to transform them into the coordinate system of the virtual model $\mathcal{V}$, the similarity transformation can be directly applied onto the points:

$$\begin{aligned}\mathbf{x}_{n,(\mathcal{V})} &= S_{\mathcal{W}\to\mathcal{V}} \cdot \mathbf{x}_{n,(\mathcal{W})} \\ &= s_{\mathcal{W}\to\mathcal{V}} \cdot \mathbf{R}_{\mathcal{W}\to\mathcal{V}} \cdot \mathbf{x}_{n,(\mathcal{W})} + \mathbf{t}_{\mathcal{W}\to\mathcal{V}}.\end{aligned} \tag{3.10}$$

- *Marker positions and orientations in marker BA, S_n or E_n, (structure parameters).* As we will discuss later, the geometry of a marker is usually represented by four 3D points corresponding to the outer corners of its binary pattern. However, these corner points are not optimized directly or individually. Rather, the marker is modeled as a rigid body, where the coordinates of the points $\mathbf{x}_{ln,(\mathcal{L}_n)}$ are fixed and defined in a local coordinate system $\mathcal{L}_n$, with each of the markers having its own origin e.g. at the center of the pattern. The location and orientation of the markers is represented by similarity transformations $S_{n,(\mathcal{L}_n\to\mathcal{W})}$ (alternatively Euclidean), which connect each local coordinate system with $\mathcal{W}$. The transformation of these

parameters to the coordinate system of the virtual model $\mathcal{V}$ amounts to a concatenation with $S_{\mathcal{W}\to\mathcal{V}}$, i.e.:

$$\begin{aligned} & S_{n,(\mathcal{L}_n\to\mathcal{V})} = S_{\mathcal{W}\to\mathcal{V}} \cdot S_{n,(\mathcal{L}_n\to\mathcal{W})}. \\ \Leftrightarrow \quad & \begin{cases} \mathbf{R}_{n,(\mathcal{L}_n\to\mathcal{V})} & = \mathbf{R}_{\mathcal{W}\to\mathcal{V}} \cdot \mathbf{R}_{n,(\mathcal{L}_n\to\mathcal{W})}. \\ \mathbf{t}_{n,(\mathcal{L}_n\to\mathcal{V})} & = s_{\mathcal{W}\to\mathcal{V}} \cdot \mathbf{R}_{\mathcal{W}\to\mathcal{V}} \cdot \mathbf{t}_{n,(\mathcal{L}_n\to\mathcal{W})} + \mathbf{t}_{\mathcal{W}\to\mathcal{V}}. \\ s_{n,(\mathcal{L}_n\to\mathcal{V}} & = s_{\mathcal{W}\to\mathcal{V}} \cdot s_{n,(\mathcal{L}_n\to\mathcal{W})}. \end{cases} \end{aligned} \tag{3.11}$$

- *Camera extrinsics, E_m, (motion parameters).* The extrinsic parameters for all frames m are given by Euclidean transformations, $E_{\mathcal{W}\to\mathcal{C}_m}$, from the real world model coordinate system $\mathcal{W}$ to the local coordinate system $\mathcal{C}_m$ of each camera. After transformation with $S_{\mathcal{W}\to\mathcal{V}}$ the extrinsics should relate points in $\mathcal{V}$ to points in $\mathcal{C}_m$, which means that the transformation of the extrinsic parameters must be as follows:

$$\begin{aligned} S_{m,(\mathcal{V}\to\mathcal{C}_m)} &= E_{m,(\mathcal{W}\to\mathcal{C}_m)} \cdot S_{\mathcal{V}\to\mathcal{W}} \\ &= E_{m,(\mathcal{W}\to\mathcal{C}_m)} \cdot S^{-1}_{\mathcal{W}\to\mathcal{V}}. \\ \Leftrightarrow \quad & \begin{cases} \mathbf{R}_{m,(\mathcal{V}\to\mathcal{C}_m)} & = \mathbf{R}_{m,(\mathcal{W}\to\mathcal{C}_m)} \cdot \mathbf{R}^{\mathsf{T}}_{\mathcal{W}\to\mathcal{V}}. \\ \mathbf{t}_{m,(\mathcal{V}\to\mathcal{C}_m)} & = -s^{-1}_{\mathcal{W}\to\mathcal{V}} \cdot \mathbf{R}_{m,(\mathcal{W}\to\mathcal{C}_m)} \cdot \mathbf{R}^{\mathsf{T}}_{\mathcal{W}\to\mathcal{V}} \cdot \mathbf{t}_{\mathcal{W}\to\mathcal{V}} + \mathbf{t}_{m,(\mathcal{W}\to\mathcal{C}_m)}. \\ s_{m,(\mathcal{V}\to\mathcal{C}_m)} & = s^{-1}_{\mathcal{W}\to\mathcal{V}}. \end{cases} \end{aligned} \tag{3.12}$$

  Since we are working with pinhole models for the cameras, images points remain invariant to global scale changes of their associated 3D points $\mathbf{x}$ in camera coordinates $\mathcal{C}_n$. This is because of the perspective division, $P(\mathbf{x}) = [x_1/x_3, x_2/x_3]^{\mathsf{T}}$, as part of the image generation process, which has the property that $P(s\mathbf{x}) = P(\mathbf{x})$ for any $s \in \mathbb{R} \setminus 0$. Therefore it is convenient to re-express the resulting camera extrinsics again as Euclidean transformations with unit scale, which can be achieved by multiplication of the parameters with $s_{\mathcal{W}\to\mathcal{V}}$:

$$E_{m,(\mathcal{V}\to\mathcal{C}_m)}: \quad \begin{cases} \mathbf{R}_{m,(\mathcal{V}\to\mathcal{C}_m)} & = \mathbf{R}_{m,(\mathcal{W}\to\mathcal{C}_m)} \cdot \mathbf{R}^{\mathsf{T}}_{\mathcal{W}\to\mathcal{V}}. \\ \mathbf{t}_{m,(\mathcal{V}\to\mathcal{C}_m)} & = -\mathbf{R}_{m,(\mathcal{W}\to\mathcal{C}_m)} \cdot \mathbf{R}^{\mathsf{T}}_{\mathcal{W}\to\mathcal{V}} \cdot \mathbf{t}_{\mathcal{W}\to\mathcal{V}} + s_{\mathcal{W}\to\mathcal{V}} \cdot \mathbf{t}_{m,(\mathcal{W}\to\mathcal{C}_m)}. \end{cases} \tag{3.13}$$

Based on these relations it is easy recognize why the *gauge freedom* is always prevalent in unconstrained bundle adjustment problems. If both, the motion and structure parameters, are transformed simultaneously as in Eqs. 3.10 to 3.13 using *any* similarity transformation, the error of the minimization problem will remain invariant to this coordinate system change. This can be seen by writing the coordinates of a scene point in camera coordinates after the transformation:

$$\begin{aligned} \mathbf{x}_{n,(\mathcal{C}_m)} &= E_{m,(\mathcal{V}\to\mathcal{C}_m)} \cdot \mathbf{x}_{n,(\mathcal{V})} \\ &= s_{\mathcal{W}\to\mathcal{V}} \cdot E_{m,(\mathcal{W}\to\mathcal{C}_m)} \cdot S^{-1}_{\mathcal{W}\to\mathcal{V}} \quad \cdot \quad S_{\mathcal{W}\to\mathcal{V}} \cdot \mathbf{x}_{n,(\mathcal{W})} \\ &= s_{\mathcal{W}\to\mathcal{V}} \cdot E_{m,(\mathcal{W}\to\mathcal{C}_m)} \cdot \mathbf{x}_{n,(\mathcal{W})}. \end{aligned} \tag{3.14}$$

As can be seen, the values of any point in camera coordinates remain the same after the transformation, apart from the scaling factor $s_{\mathcal{W}\to\mathcal{V}}$. Since the scaling factor has no influence on the final image points after perspective division, the effect of the transformation on the structure and on the motion parameters is completely canceled out.

At this point we would like to point out another property of the gauge freedom and the explicit constraints: their impact on possible choices for the actual minimization algorithm. As we showed above, the actual choice of

the coordinate system does not affect the direction at which scene points are located with respect to the cameras, and thus, the final re-projection error of the unconstrained minimization problem is invariant to the reference coordinate system. This means that the minimum of the objective function is not a single point, but rather a seven-dimensional manifold[4] that evolves through the parameter space. As a consequence, this results in a degeneracy of the Jacobian matrix, $[\mathbf{J}_a, \mathbf{J}_b]$, of line 6 in Listing 3.3.1 and also of the approximated Hessian. The rank of these matrices is always by seven smaller than the number of minimization parameters and the number of columns of $[\mathbf{J}_a, \mathbf{J}_b]$, and therefore the Hessian or its Gauss-Newton approximation, $[\mathbf{J}_a, \mathbf{J}_b]^\mathsf{T}[\mathbf{J}_a, \mathbf{J}_b]$, is not directly invertible. This property is completely independent of the size of the bundle adjustment problem and the number of measurements. This circumstance is rarely addressed in literature, but it is the actual reason why a damped minimization algorithm such as Levenberg-Marquardt must be used in the unconstrained case. The damping term, $\mu\mathbf{I}$, has the role of regulizer and ensures that these matrices become invertible again.

If external constraints are added the situation changes. If a similarity transformation can be unambiguously computed from the given constraints, it also means that the coordinate system (gauge) is in fact fixed. This again results in invertible matrices and the damping term mechanism is not needed anymore. As a result, undamped algorithms such as Gauss-Newton can be used, having in general faster convergence rates than damped variants.

3.5. Parameter Projection onto Constrained Parameter Manifold

In the previous section, we discussed how to transform the parameters into the desired coordinate system ($\mathcal{V}$) of the virtual model. This transformation is applied equally for all parameters including those with associated constraints. However, even after transformation the constrained parameters will not necessarily lie on their respective feasible sets or constraint manifolds. This is particularly true if the initial reconstruction is inaccurate and if more constraints are imposed than actually needed to merely resolve the gauge ambiguity. In fact, a remaining deviation is even necessary in order for the constrained minimization to have an effect and to compensate non-rigid deformations of the reconstruction.

Our constrained minimization is strictly feasible, which means that the constraints must be satisfied at all times during the minimization. The effect of these constraints will then distribute across all other parameters resulting in a common, global adaptation to these constraints. The feasibility of the constrained parameters must already be attained right from the beginning by projecting them onto their respective constraint manifolds. In literature, this operation is often also called *retraction*. In the following we will discuss some concrete retraction operations which are required for the realization of the use-cases in Chapters 4 and 5:

- *Complete knowledge on parameters*: If some of the parameters are completely known and fixed in all of their dimensions, then the projection simply amounts to a replacement of parameter values. As an example, in Chap. 5 we will consider the case where the user selects a subset of points from a reconstructed feature map and manually assigns them coordinates on the surface of a virtual model.
- *Partially known linear parameters*: For partially known linear parameters (scene points or translations) we assume that their affine subspaces are represented by an offset vector, $\mathbf{y}_\mathcal{V} \in \mathbb{R}^3$, and a basis $\mathbf{V}_\mathcal{V}$. If the parameters are constrained to a plane, then $\mathbf{V}_\mathcal{V} \in \mathbb{R}^{3\times 2}$ consists of two orthonormal column vectors spanning the plane. For a line, $\mathbf{V}_\mathcal{V} \in \mathbb{R}^{3\times 1}$ represents a normalized direction vector. For a parameter $\mathbf{x}' \in \mathbb{R}^3$ (point or translation), which after the transformation of Sec. 3.4 is close to, but not directly on the constraint subspace, the projection consists of finding the closest point, $\mathbf{x}$, in this subspace. This is

[4] Depending on the problem type the dimensionality of the gauge invariance can also be different, e.g. six-dimensional for marker reconstruction with fixed and known marker sizes.

achieved by the following operation:

$$\mathbf{x} = \mathbf{V}\mathbf{V}^{\mathsf{T}}(\mathbf{x}' - \mathbf{y}) + \mathbf{y}. \tag{3.15}$$

- *Known rotation axes*: For partially constrained rotations we consider 1-DoF rotations around a known axis vector, $\mathbf{w}_{\mathcal{V}}$. As an example, this situation arises in Chap. 4 for markers that are placed freely on a planar surface or along the edge of an object. In this case, the unknown part of $\mathbf{R}_{\mathcal{L}_n \to \mathcal{V}}$ comprises a rotation, whose axis must necessarily be parallel to the plane normal or the edge direction. For the projection of the initially transformed rotation matrix, $\mathbf{R}_{\mathcal{L}_n \to \mathcal{V}'}$, we exploit the knowledge on the directions of the rotation axis, $\mathbf{w}_{\mathcal{L}_n}$ in local coordinates and $\mathbf{w}_{\mathcal{V}}$ in virtual coordinates, respectively. The projected rotation, $\mathbf{R}_{\mathcal{L}_n \to \mathcal{V}}$, must ensure parallelism after transformation, i.e. $\mathbf{w}_{\mathcal{V}} \propto \mathbf{R}_{\mathcal{L}_n \to \mathcal{V}} \cdot \mathbf{w}_{\mathcal{L}_n}$. Initially, this may not be satisfied for $\mathbf{R}_{\mathcal{L}_n \to \mathcal{V}'}$, so the projection involves a correction with $\mathbf{R}_{\mathcal{V}' \to \mathcal{V}}$ of the form:

$$\mathbf{R}_{\mathcal{L}_n \to \mathcal{V}} = \mathbf{R}_{\mathcal{V}' \to \mathcal{V}} \cdot \mathbf{R}_{\mathcal{L}_n \to \mathcal{V}'}. \tag{3.16}$$

Given that $\mathbf{w}_{\mathcal{V}}$ and $\mathbf{w}_{\mathcal{L}_n}$ are normalized unit vectors, the correcting rotation can be computed as:

$$\mathbf{R}_{\mathcal{V}' \to \mathcal{V}} = \mathrm{rod}\left(\arccos(\mathbf{w}_{\mathcal{V}}^{\mathsf{T}} \mathbf{R}_{\mathcal{L}_n \to \mathcal{V}'} \mathbf{w}_{\mathcal{L}_n}) \cdot \frac{\mathbf{w}_{\mathcal{V}} \times \mathbf{R}_{\mathcal{L}_n \to \mathcal{V}'} \mathbf{w}_{\mathcal{L}_n}}{\| \mathbf{w}_{\mathcal{V}} \times \mathbf{R}_{\mathcal{L}_n \to \mathcal{V}'} \mathbf{w}_{\mathcal{L}_n} \|} \right), \tag{3.17}$$

where $\mathrm{rod}(.)$ denotes the Rodrigues' formula:

$$\mathrm{rod}(\mathbf{w}) = \mathbf{I} + \frac{\sin(\|\mathbf{w}\|)}{\|\mathbf{w}\|} [\mathbf{w}]_{\times} + \frac{(1 - \cos(\|\mathbf{w}\|))}{\|\mathbf{w}\|^2} [\mathbf{w}]_{\times}^2, \tag{3.18}$$

Once the alignment of the rotation axis (feasibility of axis constraint) is achieved, it will not be spoiled anymore during minimization, as long as the updates, $\Delta\mathbf{R}$, comprise a sequence of left-handed rotations around the axis $\mathbf{w}_{\mathcal{V}}$ of the form $\mathbf{R}^{(i)}_{\mathcal{L}_n \to \mathcal{V}} = \Delta\mathbf{R}^{(i)}(\mathbf{w}_{\mathcal{V}}) \cdot \ldots \cdot \Delta\mathbf{R}^{(0)}(\mathbf{w}_{\mathcal{V}}) \cdot \mathbf{R}^{(0)}_{\mathcal{L}_n \to \mathcal{V}}$. This is because for any rotation matrix $\mathbf{R}(\mathbf{w})$ its corresponding rotation vector $\mathbf{w}$ remains invariant under rotation, i.e. $\mathbf{w} = \mathbf{R}(\mathbf{w}) \cdot \mathbf{w}$. Similarly, right-handed rotation increments around the axis, $\mathbf{w}_{\mathcal{L}_n}$, in local coordinates, i.e. $\mathbf{R}^{(0)}_{\mathcal{L}_n \to \mathcal{V}} \cdot \Delta\mathbf{R}^{(0)}(\mathbf{w}_{\mathcal{L}_n}) \cdot \ldots \cdot \Delta\mathbf{R}^{(i)}(\mathbf{w}_{\mathcal{L}_n})$, also preserve the alignment for the same reason.

3.6. Jacobian Setup and Minimization on Geodesics

We will now turn towards the inclusion of constraints into the actual iterations for minimizing the objective function Eq. 3.5. Unlike the standard approach (see Triggs et al. [TMHF00]), which consists of searching for stationary points of the Lagrangian with Sequential Quadratic Programming (SQP), our approach is based on the minimization-on-manifold paradigm. This preserves the least-squares character of the problem so that Gauss-Newton or Levenberg-Marquardt iterations can still be used.

In general, each iteration in sparse Gauss-Newton or Levenberg-Marquardt involves three steps.

- *Setting up the Jacobian* $\mathbf{J}$ *or directly the approximated Hessian* $\mathbf{J}^{\mathsf{T}}\mathbf{J}$: The composition of these matrices can be achieved by an independent computation of partial derivatives per measurement $\mathbf{p}_k$ and adding each of these contributions to the final matrix $\mathbf{J}^{\mathsf{T}}\mathbf{J}$. In sparse BA, each measurement is only dependent on two distinct subsets of fixed size, $\mathbf{a}_{m(k)}$ and $\mathbf{b}_{m(k)}$ (e.g. one camera extrinsic and one 3D point), and functions for computing the derivatives are re-used for each measurement to model BA problems of varying size. Constrained parameters, however, do need individual derivative functions, which we obtain

by a linearization (tangent hyper-plane) of the manifold induced by the constraint. The mathematical modeling is described in Sec. 3.6.1 for various examples and Sec. 3.6.2 outlines how corresponding code for the miminizer is derived.

- *Computing the parameter increments*: The computation of the increments along their tangent spaces essentially requires the inversion of the matrix $\mathbf{J}^\mathsf{T}\mathbf{J}$. An efficient BA implementation must exploit its sparsity. A standard technique is the Schur complement trick, which we will shortly review in Sec. 3.6.3 together with an outline of other implementation aspects directly related to the manifold representation of constrained parameters. We will then discuss the implementations of existing sparse minimization libraries with regard to their the applicability to model-constrained manifold-based minimization 3.7.
- *Parameter update*: Traditionally, GN or LM solvers follow a simple additive update rule for the parameters. However, since the dimensionality of the constraint manifold and the parameter increments differs from the size of the global parameter vector, such an additive update is not possible for an optimization-on-manifold approach. Moreover, the parameter update must ensure that the constraints remain satisfied after the update. The linearized approximation (tangent space) of the parameters generally violates these constraints unless the changes are ininitesimally small. Therefore the update must also include a projection (also called *retraction*) of the parameters back onto the manifold. Certain constraints or parameter types allow for a closed-form update on curves on the manifold (geodesics) along the increment direction. We will discuss the geodesic updates for the relevant parameter types of this work together with the associated tangent spaces in Sec. 3.6.1.

3.6.1. Local Parametrization and Geodesic Parameter Update

Our constrained minimization approach is based on the idea that every constraint imposed on the parameters reduces the dimensionality of the parameters, so that the remaining feasible set forms a differentiable manifold in the original higher dimensional parameter space. In each iteration we identify the tangent space for every constrained parameter. Then, we compute the parameter increments in local coordinates of the tangent space along properly chosen basis vectors. Finally, we update the original parameters on geodesics of the manifold along the direction of the increments. Again, we exemplify this approach on the various parameter and constraint types, as used later on in this chapter:

- *Full, three-DoF rotations*: Even if no explicit constraints are imposed on rotation parameters, we must nevertheless ensure that we only minimize along their three DoF and that the parameters always represent a valid rotation. For rotation matrices, $\mathbf{R}$, it means that $\mathbf{R}^\mathsf{T}\mathbf{R} = \mathbf{I}$ and $\det(\mathbf{R}) = 1$ is satisfied. In many earlier BA implementations a global parametrization with additive update is used, which may result in singularities.[5] We use a local parametrization, $\boldsymbol{\omega} \in \mathbb{R}^3$, with multiplicative update based on the theory of Lie groups / algebra for rotations [TK94, AMS07, SMD10b]:

$$\mathbf{R}^{(i+1)} = \underbrace{\exp(\Delta\boldsymbol{\Omega})}_{\Delta\mathbf{R}} \cdot \mathbf{R}^{(i)}, \tag{3.19}$$

[5] For example, in the well-known SBA library [LA09] the parametrization is based on the imaginary parts, $\check{\mathbf{q}} := [q_1, q_2, q_3]^\mathsf{T}$, of a quaternion, $\mathbf{q} := [q_0, q_1, q_2, q_3]^\mathsf{T}$, and the update in iteration i follows the rule: $\mathbf{q}^{(i+1)} = [\sqrt{1 - (\check{\mathbf{q}}^{(i)} + \Delta\check{\mathbf{q}})^\mathsf{T}(\check{\mathbf{q}}^{(i)} + \Delta\check{\mathbf{q}})}, (\check{\mathbf{q}}^{(i)} + \Delta\check{\mathbf{q}})^\mathsf{T}]^\mathsf{T}$, i.e. the real part is recalculated implicitly based on the updated imaginary part. This additive update becomes problematic whenever the real part is already close to zero (rotations with an angle of π). In this case the norm of the updated imaginary part, $\check{\mathbf{q}}^{(i)} + \Delta\check{\mathbf{q}}$, may become larger than one, so that no real-valued q_0 can be computed and the update fails. Similar problems arise for other global parameterizations (Euler angles, axis-angle) with additive update.

where

$$\exp(\Delta\boldsymbol{\Omega}) := \sum_{j=0}^{\infty} \tfrac{1}{j!}\Delta\boldsymbol{\Omega}^j = \mathbf{I} + \Delta\boldsymbol{\Omega} + \tfrac{1}{2}\Delta\boldsymbol{\Omega}\Delta\boldsymbol{\Omega} + \tfrac{1}{6}\Delta\boldsymbol{\Omega}\Delta\boldsymbol{\Omega}\Delta\boldsymbol{\Omega} + \cdots \tag{3.20}$$

denotes the matrix exponential of $\Delta\boldsymbol{\Omega}$, which can be computed explicitly using the Rodrigues' formula of Eq. 3.18, and

$$\Delta\boldsymbol{\Omega} := [\Delta\boldsymbol{\omega}]_\times = \begin{bmatrix} 0 & -\Delta\omega_3 & \Delta\omega_2 \\ \Delta\omega_3 & 0 & -\Delta\omega_1 \\ -\Delta\omega_2 & \Delta\omega_1 & 0 \end{bmatrix} \tag{3.21}$$

is an element of the set of skew-symmetric matrices, which comprises the Lie algebra $\mathfrak{so}(3)$ of the group of rotations $SO(3)$. Since we are only interested in first-order partial derivatives during minimization, we can drop any higher order terms in Eqs. 3.19 and 3.20. Thus, we compute the derivatives of $f(.)$ with regard to unconstrained rotation parameters for $\boldsymbol{\omega} = \mathbf{0}$ (around identity: $\Delta\mathbf{R} = \mathbf{I}$) as:

$$\frac{\partial f(\mathbf{R}(\boldsymbol{\omega}))}{\partial \boldsymbol{\omega}} = \frac{\partial f((\mathbf{I} + \boldsymbol{\Omega}) \cdot \mathbf{R})}{\partial \boldsymbol{\omega}}\bigg|_{\boldsymbol{\omega}=\mathbf{0}}, \tag{3.22}$$

which leads to very simple and efficient expressions. From this we can readily identify that the set of matrices

$$\left\{ \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix} \mathbf{R}^{(i)}, \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ -1 & 0 & 0 \end{bmatrix} \mathbf{R}^{(i)}, \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \mathbf{R}^{(i)} \right\} \tag{3.23}$$

forms a basis for the tangent space of a rotation at the current estimate $\mathbf{R}^{(i)}$ and the elements of $\Delta\boldsymbol{\omega}$ denote local coordinates in that space (which in Alg. 3.3.1 correspond to the generic variable names $\mathbf{V_a}$ / $\mathbf{V_b}$ and $\boldsymbol{\alpha}$ / $\boldsymbol{\beta}$, respectively). The linearization is only used for the derivatives, whereas updates with $\Delta\boldsymbol{\omega}$ are carried out using the exact (non-linearized) Eq. 3.20 together with Eq. 3.18, so that the result always remains a rotation. The set of points, $\exp(\lambda\Delta\boldsymbol{\Omega}) \cdot \mathbf{R}^{(i)}$ with $\lambda \in [0, 1]$, represents the shortest path on the manifold of rotations that connects $\mathbf{R}^{(i)}$ with $\mathbf{R}^{(i+1)}$, which is why the formula of Eq. 3.20 can be considered as an update along a manifold geodesic. Finally, we note that one can just as well execute right-handed updates, i.e. $\mathbf{R}^{(i+1)} = \mathbf{R}^{(i)} \cdot \exp(\Delta\boldsymbol{\Omega})$, as long as the computation of the derivatives and local tangent spaces (Eqs. 3.22 and 3.23) is adapted and matches the update function appropriately.

- *Axis-constrained rotations*: In order to preserve the alignment of the rotation axis, the update can either be applied with a left-handed multiplication with a rotation matrix around $\mathbf{w}_\mathcal{V}$ or right-handed around $\mathbf{w}_{\mathcal{L}_n}$,

$$\mathbf{R}^{(i+1)} = \begin{cases} \exp(\Delta\omega[\mathbf{w}_\mathcal{V}]_\times) & \mathbf{R}^{(i)} \\ & \mathbf{R}^{(i)} \exp(\Delta\omega[\mathbf{w}_{\mathcal{L}_n}]_\times) \end{cases}, \tag{3.24}$$

where the optimization variable ω is a scalar representing the angle of rotation (in Alg. 3.3.1 denoted by generic variables α or β). For the computation of the derivatives, $\partial f(.)/\partial\omega$, all higher order terms in the matrix exponential can be neglected again. The corresponding tangent spaces of the constrained rotation are thus given by

$$\begin{cases} [\mathbf{w}_\mathcal{V}]_\times \cdot \mathbf{R}^{(i)} & \textit{for left-handed updates} \\ \mathbf{R}^{(i)} \cdot [\mathbf{w}_{\mathcal{L}_n}]_\times & \textit{for right-handed updates} \end{cases}, \tag{3.25}$$

- *Partially constrained linear parameters*: for scene points, $\mathbf{x}$, or translations, $\mathbf{t}$, which are bound to an affine subspace, $\mathbf{y}_\mathcal{V} + \mathbf{V}_\mathcal{V} \cdot \boldsymbol{\beta}$ (plane or line), the update follows the rule,

$$\mathbf{x}^{(i+1)} = \mathbf{x}^{(i)} + \mathbf{V}_\mathcal{V} \Delta\boldsymbol{\beta}, \tag{3.26}$$

from which it follows that the orthonormal column vectors in $\mathbf{V}_\mathcal{V}$ spanning the plane or line also represent basis vectors for the tangent space of the constrained parameters, and the elements of $\boldsymbol{\beta}$ are coordinates in that space.

- *Scale parameters*: Scale parameters inside similarity transformations are parametrized using exponential coordinates with multiplicative update in analogy to parametrization of rotations:

$$s^{(i+1)} = s^{(i)} e^{\Delta\sigma}, \tag{3.27}$$

  which has the advantage that the scale will always remain positive. For small $\Delta\sigma$ near zero we have $s^{(i+1)} \approx s^{(i)} + s^{(i)}\Delta\sigma$, so we use $s^{(i)}$ as basis value that 'spans' the tangent space (trivially it is $\mathbb{R}$).
- *Complete knowledge on parameters*: fully constrained parameters have a tangent space of dimension zero. They neither have associated local variables nor do they possess an update rule.

We note at this point that we always treat rotations, translations and scalings as independent parameters. In literature, people have also derived and applied the Lie algebra for full Euclidean or similarity transformations. A good overview about the major transformation groups in computer vision and their associated Lie algebra is given by Zacur [ZBO14] and Eade [Ead]. Coupling the rotation, translation and scale parameters in this way leads to different update functions and derivatives (tangent spaces) for the translation and scale part. However, according to our experience it will not improve convergence speed of the minimization, it merely increases the complexity and computational time for the update and derivative computation steps.

3.6.2. Symbolically Computing Partial Derivatives and Code Generation

The implementation of the Levenberg-Marquardt algorithm requires the computation of the derivatives,$\partial f/\partial \boldsymbol{\alpha}$ and $\partial f/\partial \boldsymbol{\beta}$, of the measurement prediction function with regard to the local parameters $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ in each iteration, as indicated in line 6 of Algorithm 3.3.1. Since the computation of Jacobians in BA is one of major the bottlenecks, an efficient implementation is desirable. In general, there are four different ways for obtaining partial derivatives for functions [BPR15]:

- *Figuring out the derivatives manually* and coding the result,
- *Numerical differentiation*,
- *Automatic (algorithmic) differentiation* [GW08], and
- *Symbolic differentiation* using expression manipulation with an external computer algebra software (CAS) such as Mathematica, Maple or the Symbolic Math Toolbox of Matlab.

Manually working out the derivatives can be time consuming and error-prone and does not always produce optimal results. When using *numerical differentiation*, the derivatives are approximated by finite differences, $\partial f(x)/\partial x \approx (f(x) - f(x+h))/h$, i.e. the function is simply evaluated multiple times for varying inputs, so no knowledge of the internal computations is necessary. For multivariate vector-valued functions, $f : \mathbb{R}^p \to \mathbb{R}^q$, it will however require at least $pq+1$ evaluations, which may be very inefficient. Moreover, numerical approximations of derivatives need a careful selection of the step size h, otherwise it results in inaccuracies, either because of round-off errors due to limited machine precision when h is too small, or because of truncation errors when h is to large. *Automatic differentiation* (AD) refers to a technique which determines exact derivatives at specific numerical values of the function arguments and their internal intermediate variables. A function or a program is considered to be a series of atomic operations such as additions or multiplications, for which the derivatives are assumed to be known. The derivatives of the outputs of the complete function with regard to its inputs are determined by propagating sub-derivatives of intermediate variables along the program flow by means of the chain rule of differential calculus. For this purpose AD engines typically provide own base-types and operator

overloads. Functions, for which derivatives are needed, must be implemented using these proprietary types. Existing software packages for AD, which can be used for C++ code, include ADOL-C [GJU96], FADBAD [BS96] and CppAD [Bel03]. More recent libraries like Adept [Hog14] and the AD support of Ceres [AMO] are programmed using the C++ expression template technique [Vel95], which permits a simple usage and at the same time transfers some of the computational load to compile time. There are two basic implementation variants or operation modes of AD engines: *forward differentiation* and *reverse differentiation*. The forward mode is efficient for univariate vector-valued functions, $f : \mathbb{R}^1 \to \mathbb{R}^q$, while for multivariate scalar functions, $f : \mathbb{R}^p \to \mathbb{R}^1$, it creates an overhead of factor p regarding the number of necessary floating point operations, very much like the numerical differentiation. The opposite is true for reverse differentiation [Gue07]. In general, the task of expressing or computing Jacobians of multivariate vector-valued functions with a minimal sequence of floating point operations is known to be NP-complete [Nau08], so it is illusive to expect any technique to produce optimal results for arbitrary inputs.

We have decided to use *symbolic differentiation* to obtain the derivatives. This approach is suitable for functions up to a moderately low complexity and generally results in more efficient code for the derivatives than the automatic differentiation technique. However, for complex functions with several hundreds or thousands of intermediate variables, symbolic derivations will quickly explode regarding memory consumption and computation time for the optimization of the expressions. In this case one needs to use one of the other techniques. In our case this is not an issue, because the BA problems have a sparse structure. This means that derivatives of image points are zero for almost all parameters of the whole problem. It is only necessary to compute the derivative with respect to those variables that have a direct influence on the predicted image point.

To this end, the optimization algorithm needs to be equipped with certain types of derivative functions: one for each possible combination of constraints that might potentially exist between pairs of motion and structure parameter subsets. In order to ease the generation of derivative functions, we have implemented an automatic C++ code generator in Python using the SymPy package[6]. It accepts a model definition as input and produces C++ code as compilable functions. Model definitions describe the functions $f_k(\mathbf{a}_m, \mathbf{b}_n)$ in symbolic form and are kept in separate files. Every model definition contains the following sections:

1. *Expressing the prediction function symbolically* in terms of the tangent spaces and local parameters according to the constraints.
2. *Marking of parameters* as global, local or constant variables and whether they belong to the motion $\mathbf{a}_m$ or the structure $\mathbf{b}_n$ components.
3. Definition of the *evaluation point for local parameters*, i.e. the actual values for $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ at which $\partial f / \partial \boldsymbol{\alpha}, \boldsymbol{\beta}$ are determined. Usually the evaluation point is set to zero (as indicated in line 6 of Alg. 3.3.1), e.g. for rotation parameters we evaluate the derivatives at $\boldsymbol{\omega} = 0$ (see Eq. 3.22) or $\Delta\mathbf{R} = \mathbf{I}$.

Based on a model definition the main script of the code generator performs the following sequence of operations:

1. *Computation of the Jacobians* as a matrix with symbolic entries.
2. *Substitution of local parameter symbols* in the Jacobians with their evaluation points.
3. Application of the *Common Subexpression Elimination (CSE)*, which converts the Jacobian into a list of statements in symbolic form. The CSE module attempts to optimize the list by means of heuristics (due to NP-completeness of the problem), so that it ideally comprises a minimal amount of floating-point operations.

[6] We have also experimented with the Symbolic Math Toolbox of Matlab using `simplify()` and `ccode()` for expression optimization and C-code generation. But the resulting code was much less compact than then one produced by our generator, which used the CSE routine of SymPy for optimization.

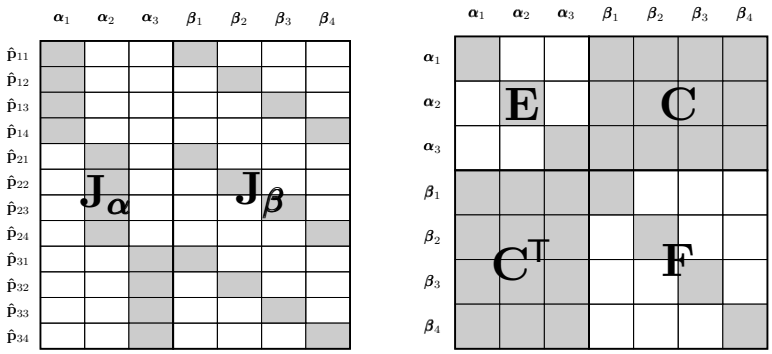
4. Performing a *C++ code conversion* of the statement list and writing it into an output file as a C++ callable function with a predefined signature.

The Levenberg-Marquardt algorithm not only needs to evaluate the Jacobian, but also the prediction function for each measurement. Usually both operations share similar floating point expressions, e.g. $f(x) = x^3$ and $\partial f(x)/\partial x = 3x^2$ both have $x \cdot x$ in common. We therefore also provide the possibility to derive functions, which compute both, the derivative and the prediction in one single run. In this case, the CSE module identifies a large portion of common expressions and often produces code that is significantly more compact and efficient than calling the two functions individually.

3.6.3. Schur Complement Trick and Implementation Aspects

For an efficient implementation of the Levenberg-Marquardt or Gauss-Newton algorithm, it is important to exploit the sparsity of typical BA problems. As discussed at the beginning of this chapter on page 41, this means that the predicted location, $\hat{\mathbf{p}}_k$, of a measurement is directly influenced by only a very small subset among all optimization variables. As a consequence the partial derivatives w.r.t any of the other parameters are zero. Fig. 3.3 shows an example of the sparsity pattern of the Jacobian, $\mathbf{J}$, and approximated Hessian, $\mathbf{J}^\mathsf{T}\mathbf{J}$, where the zero-valued sub-matrices are left blank. Note that this example represents a maximally dense structure, i.e. the sparsity can only further increase, e.g. whenever some measurements for parameter combinations are not observable, e.g. due to occlusion or because the scene points are outside of the camera frustum.

GN or LM effectively require to invert the approximated Hessian, $\mathbf{J}^\mathsf{T}\mathbf{J} \in \mathbb{R}^{D\times D}$, where the size $D = D_{\boldsymbol{\alpha}} + D_{\boldsymbol{\beta}}$ of the matrix corresponds to the total number of DoF of all motion ($D_{\boldsymbol{\alpha}}$) and structure parameters ($D_{\boldsymbol{\beta}}$). Applying a standard matrix inversion or equation solving algorithm for dense matrices directly on the entire matrix $\mathbf{J}^\mathsf{T}\mathbf{J}$ will result in a cubic effort, $\mathcal{O}(D_{\boldsymbol{\alpha}} + D_{\boldsymbol{\beta}})^3 \approx \mathcal{O}(M+N)^3$, for this step.



(a) (b)

Figure 3.3.: Schematic representation of the sparse structure of (a) the Jacobian, $\mathbf{J} = [\mathbf{J}_{\boldsymbol{\alpha}}, \mathbf{J}_{\boldsymbol{\beta}}]$, and (b) the approximated Hessian, $\mathbf{H} = \mathbf{J}^\mathsf{T}\mathbf{J} + \mu\mathbf{I}$, for an example with three subsets of motion and four subsets of structure parameters. Non-zero blocks are marked in grey.

The *Schur complement trick* reduces this effort significantly by exploiting the fact that the sub-matrices $\mathbf{E}$ and $\mathbf{F}$ (see Fig. 3.3 (b)) have a block-diagonal structure. The basic idea is to first solve for the smaller set of parameters, and then to determine the others by back-substitution. For example, the motion increments can be obtained via

$$\Delta\boldsymbol{\alpha} = \left(\mathbf{E} - \mathbf{C}\mathbf{F}^{-1}\mathbf{C}^\mathsf{T}\right)^{-1}\mathbf{J}_{\boldsymbol{\alpha}}^\mathsf{T}\Delta\mathbf{p}, \tag{3.28}$$

with $\Delta\mathbf{p} = \mathbf{p} - f(\mathbf{a}, \mathbf{b})$. Since $\mathbf{F}$ is block-diagonal, its inverse, $\mathbf{F}^{-1}$, will also be block-diagonal and can be obtained by inverting each of the sub-blocks independently. Thus the effort for this operation is linear w.r.t. the number of structure parameter subsets, N. The same remains true for the subsequent operations for computing the Schur complement, $\mathbf{S} = \mathbf{E} - \mathbf{C}\mathbf{F}^{-1}\mathbf{C}^\mathsf{T}$, of $\mathbf{F}$, if one carefully avoids to operate on zero-valued entries during the matrix multiplications. The most expensive operation therefore is the inversion of $\mathbf{S} \in \mathbb{R}^{D_{\boldsymbol{\alpha}} \times D_{\boldsymbol{\alpha}}}$.[7] Since this matrix is dense in general, the effort will remain cubic with regard to the DoF of the motion parameters, $D_{\boldsymbol{\alpha}}$. The structure parameter increments can then be determined as

$$\Delta\boldsymbol{\beta} = \mathbf{F}^{-1}\left(\mathbf{J}_{\boldsymbol{\beta}}^\mathsf{T}\Delta\mathbf{p} - \mathbf{C}^\mathsf{T}\Delta\boldsymbol{\alpha}\right), \tag{3.29}$$

which requires a bi-linear ($\mathcal{O}(D_{\boldsymbol{\alpha}} D_{\boldsymbol{\beta}})$) effort and thus is negligible. In a similiar fashion, one can also solve for the structure parameter increments, $\Delta\boldsymbol{\beta}$, first, and substituting them to obtain $\Delta\boldsymbol{\alpha}$, which simply reverses the complexity analysis. In summary, the Schur complement trick reduces the overall effort for the LM iterations to either $\mathcal{O}(D_{\boldsymbol{\alpha}}^3 + D_{\boldsymbol{\beta}}) \approx \mathcal{O}(M^3 + N)$ or $\mathcal{O}(D_{\boldsymbol{\alpha}} + D_{\boldsymbol{\beta}}^3) \approx \mathcal{O}(M + N^3)$.

The SBA library by Lourakis et al. [LA09] was one of the first publicly available implementations of BA, in which this technique was realized, and it was therefore also most widely used for a long time. There even exist very recent, light-weight SLAM implementations such as ATAM [UTIdML15] that are still using it. However, it also has two major limitations that prevent to include constraints in the way which we are proposing in this work. First, it does not allow to associate different measurement prediction and Jacobian functions to individual measurements. Second, all motion and all structure parameters are supposed to have the same dimensionality or DoF (e.g. 6 DoF for all motion and 3 DoF for all structure parameters). This simplifies the implementation for unconstrained BA, because it results in a direct mapping between the parameter indexes $m \in [1, ..., M]$, $n \in [1, ..., N]$ and the corresponding position of sub-matrices inside the Jacobian, $\mathbf{J}$, that are affected by the parameters. However, in the constrained case the parameters may have different DoF depending on the dimensionality of their associated tangent spaces and their local parametrizations. As a result, the position of affected sub-matrices inside the Jacobian is not a function of the indexes m and n anymore, but rather of the accumulated DoF of all preceding elements in the parameter vector.

We therefore implemented a more agile version of the sparse LM algorithm, which addresses these limitations of the SBA library. Our implementation allows the association of individual prediction / Jacobian functions for each measurement and a flexible linkage to associated parameters. Programmatically this is realized in C++ by introducing two types of classes: measurement objects and parameter objects. Each *measurement object* contains (1) the values of the measured image points p_k, (2) pointers to the associated parameter objects and constant values ($\bar{\mathbf{c}}_k$), and (3) function pointers to the measurement prediction function, f_k, and the corresponding jacobian function, $\partial f_k / \partial(\boldsymbol{\alpha}_{m(k)}, \boldsymbol{\beta}_{n(k)})$, or alternatively a function which computes both in a single run. Each *parameter object* contains (1) the values of the global parameters $\mathbf{a}_m$ or $\mathbf{b}_n$, (2) function pointers for the geodesic update of $\mathbf{a}_m$ or $\mathbf{b}_m$ based on the local increments $\boldsymbol{\alpha}_m$, $\boldsymbol{\beta}_n$, (3) indicators of the number of DoF, $d_{\boldsymbol{\alpha}_m}$ or $d_{\boldsymbol{\beta}_n}$, per parameter subset, and (4) an index ($D_{\boldsymbol{\alpha}_m}$ or $D_{\boldsymbol{\beta}_n}$) of the corresponding column in the Jacobian, which is pre-computed automatically by accumulation of DoF of preceding parameters based on on a fixed order among them.

Our generic LM optimization routine contains a list of parameter objects and a list of measurement objects. Alg. 3.6.1 sketches the main part of the LM procedure, which can be seen as a more detailed view into lines 6 and 7 of Alg. 3.3.1. In order to construct the sub-matrices, $\mathbf{E}$, $\mathbf{F}$, $\mathbf{C}$, $\mathbf{J}_{\boldsymbol{\alpha}}^\mathsf{T}\Delta\mathbf{p}$, and $\mathbf{J}_{\boldsymbol{\alpha}}^\mathsf{T}\Delta\mathbf{p}$, it first iterates once through the list of measurement objects (Lines 1-9).[8] For each measurement it calls the associated prediction and Jacobian functions with the associated parameters as arguments. The resulting measurement error vector

[7] In practice, we do not compute the inverse of $\mathbf{S}$ explicitly. Rather, we use appropriate routines from LAPACK, which directly solve the system of equations, $\mathbf{S}\Delta\boldsymbol{\alpha} = \mathbf{J}_{\mathbf{a}}\Delta\mathbf{p}$, either by means of a Cholesky (`DPOSV`) or Bunch-Kaufman (`DSYSV`) factorization of $\mathbf{S}$.

[8] The block-diagonal matrices $\mathbf{E}$ and $\mathbf{F}$ is stored as a list of the diagonal sub-matrix blocks, not as full matrices.

Algorithm 3.6.1 Construction of sparse Jacobian / Hessian and the computation of parameter increments via Schur complements (further specification of lines 6 and 7 in Alg. 3.3.1).

Input:

$\mathbf{a}_m, \mathbf{b}_n, \bar{\mathbf{c}}_k$ ▷ Motion, structure and complementary variables as global (over-parametrized) parameters.

$\mathbf{p}_k$ ▷ Image measurements.

$f_k, \partial f_k / \partial(\boldsymbol{\alpha}_m, \boldsymbol{\beta}_n)$ ▷ Measurement prediction and Jacobian functions for each measurement.

$n(k), m(k)$ ▷ association / mapping between measurement (k) and parameter subset indexes (n / m).

$d_{\boldsymbol{\alpha}_m}, d_{\boldsymbol{\beta}_n}$ ▷ Dimensionality of tangent spaces / DoF of constraint parameters $\mathbf{a}_m$ and $\mathbf{b}_n$.

$D_{\boldsymbol{\alpha}_m} = \sum_{i=1}^{m-1} d_{\boldsymbol{\alpha}_i}$,

$D_{\boldsymbol{\beta}_n} = \sum_{j=1}^{n-1} d_{\boldsymbol{\beta}_j}$ ▷ Precomputed cumulative DoF / associated starting column of Jacobian.

Output:

$\Delta\boldsymbol{\alpha}_m, \Delta\boldsymbol{\beta}_n$ ▷ Local parameter increments w.r.t. the tangent spaces.

▷ Iterate over all measurements:

1: **for all** $k \in \{1, ..., K\}$ **do**

▷ Compute measurement error:

2: $\Delta\mathbf{p}_k \leftarrow \mathbf{p}_k - f_k(\mathbf{a}_{m(k)}, \mathbf{b}_{n(k)}, \bar{\mathbf{c}}_k)$

▷ Compute sub-Jacobian contributions for each measurement, $\mathbf{A} \in \mathbb{R}^{2 \times d_{\boldsymbol{\alpha}_m}}$ and $\mathbf{B} \in \mathbb{R}^{2 \times d_{\boldsymbol{\beta}_n}}$:

3: $[\mathbf{A}, \mathbf{B}] \leftarrow \partial f_k(.) / \partial(\boldsymbol{\alpha}_{m(k)}, \boldsymbol{\beta}_{n(k)}) \,|_{\boldsymbol{\alpha}_m, \boldsymbol{\beta}_n = \mathbf{0}}$

▷ Update diagonal block, $\mathbf{E}_{D_{\boldsymbol{\alpha}_m}, D_{\boldsymbol{\alpha}_m}} \in \mathbb{R}^{d_{\boldsymbol{\alpha}_m} \times d_{\boldsymbol{\alpha}_m}}$, inside $\mathbf{E}$ at position $(D_{\boldsymbol{\alpha}_m}, D_{\boldsymbol{\alpha}_m})$:

4: $\mathbf{E}_{D_{\boldsymbol{\alpha}_m}, D_{\boldsymbol{\alpha}_m}} \leftarrow \mathbf{E}_{D_{\boldsymbol{\alpha}_m}, D_{\boldsymbol{\alpha}_m}} + \mathbf{A}^\mathsf{T}\mathbf{A}$

▷ Update diagonal block, $\mathbf{F}_{D_{\boldsymbol{\beta}_n}, D_{\boldsymbol{\beta}_n}} \in \mathbb{R}^{d_{\boldsymbol{\beta}_n} \times d_{\boldsymbol{\beta}_n}}$, inside $\mathbf{F}$:

5: $\mathbf{F}_{D_{\boldsymbol{\beta}_n}, D_{\boldsymbol{\beta}_n}} \leftarrow \mathbf{F}_{D_{\boldsymbol{\beta}_n}, D_{\boldsymbol{\beta}_n}} + \mathbf{B}^\mathsf{T}\mathbf{B}$

▷ Update off-diagonal block, $\mathbf{C}_{D_{\boldsymbol{\alpha}_m}, D_{\boldsymbol{\beta}_n}} \in \mathbb{R}^{d_{\boldsymbol{\alpha}_m} \times d_{\boldsymbol{\beta}_n}}$, inside $\mathbf{C}$:

6: $\mathbf{C}_{D_{\boldsymbol{\alpha}_m}, D_{\boldsymbol{\beta}_n}} \leftarrow \mathbf{C}_{D_{\boldsymbol{\alpha}_m}, D_{\boldsymbol{\beta}_n}} + \mathbf{A}^\mathsf{T}\mathbf{B}$

▷ Update the sub-vector, $\mathbf{e}_{\mathbf{a}, D_{\boldsymbol{\alpha}_m}} \in \mathbb{R}^{d_{\boldsymbol{\alpha}_m}}$, inside $\mathbf{e}_\mathbf{a} := \mathbf{J}_{\boldsymbol{\alpha}}^\mathsf{T}\Delta\mathbf{p}$, at position $D_{\boldsymbol{\alpha}_m}$:

7: $\mathbf{e}_{\mathbf{a}, D_{\boldsymbol{\alpha}_m}} \leftarrow \mathbf{e}_{\mathbf{a}, D_{\boldsymbol{\alpha}_m}} + \mathbf{A}^\mathsf{T}\Delta\mathbf{p}_k$

▷ Update the sub-vector, $\mathbf{e}_{\mathbf{b}, D_{\boldsymbol{\beta}_n}} \in \mathbb{R}^{d_{\boldsymbol{\beta}_n}}$, inside $\mathbf{e}_\mathbf{b} := \mathbf{J}_{\boldsymbol{\beta}}^\mathsf{T}\Delta\mathbf{p}$, at position $D_{\boldsymbol{\beta}_n}$:

8: $\mathbf{e}_{\mathbf{b}, D_{\boldsymbol{\beta}_n}} \leftarrow \mathbf{e}_{\mathbf{b}, D_{\boldsymbol{\beta}_n}} + \mathbf{B}^\mathsf{T}\Delta\mathbf{p}_k$

9: **end for**

▷ Add LM damping factor:

10: $\mathbf{E} \leftarrow \mathbf{E} + \mu\mathbf{I}$

11: $\mathbf{F} \leftarrow \mathbf{F} + \mu\mathbf{I}$

▷ Solve for the increments via the Schur complement:

12: **if** $D_{\boldsymbol{\alpha}} < D_{\boldsymbol{\beta}}$ **then**

13: $\Delta\boldsymbol{\alpha} \leftarrow \left(\mathbf{E} - \mathbf{C}\mathbf{F}^{-1}\mathbf{C}^\mathsf{T}\right)^{-1}\mathbf{e}_\mathbf{a}$

14: $\Delta\boldsymbol{\beta} \leftarrow \mathbf{F}^{-1}\left(\mathbf{e}_\mathbf{b} - \mathbf{C}^\mathsf{T}\Delta\boldsymbol{\alpha}\right)$

15: **else**

16: $\Delta\boldsymbol{\beta} \leftarrow \left(\mathbf{F} - \mathbf{C}^\mathsf{T}\mathbf{E}^{-1}\mathbf{C}\right)^{-1}\mathbf{e}_\mathbf{b}$

17: $\Delta\boldsymbol{\alpha} \leftarrow \mathbf{E}^{-1}\left(\mathbf{e}_\mathbf{a} - \mathbf{C}\Delta\boldsymbol{\beta}\right)$

18: **end if**

$\Delta\mathbf{p}_k$ and sub-Jacobian contributions, $\mathbf{A}$ and $\mathbf{B}$, are then accumulated into $\mathbf{E}$, $\mathbf{F}$, $\mathbf{C}$, $\mathbf{J}_{\boldsymbol{\alpha}}^{\mathsf{T}}\Delta\mathbf{p}$ and $\mathbf{J}_{\boldsymbol{\alpha}}^{\mathsf{T}}\Delta\mathbf{p}$. The information on the correct positions is taken from the indexes $D_{\boldsymbol{\alpha}_m}$ and $D_{\boldsymbol{\beta}_n}$ of the parameters that are linked to the measurement objects. Next, the increments are solved via the Schur complement. We make the decision on which increments to solve first ($\Delta\boldsymbol{\alpha}$ or $\Delta\boldsymbol{\beta}$) based on the total number of DoF of each parameter family, $D_{\boldsymbol{\alpha}}$ and $D_{\boldsymbol{\beta}}$ (Line 12-18). Finally, we iterate through the list of parameter objects and call each associated update function with $\Delta\boldsymbol{\alpha}_m$ or $\Delta\boldsymbol{\beta}_n$ as arguments (not shown in Alg. 3.6.1).

3.7. Alternative Realization with Other Sparse Optimization Libraries

Our implementation of the sparse LM optimizer was motivated by the limitations of the SBA library [LA09]. Our initial version was capable of incorporating constraints in the form of complete knowledge on certain parameter subsets $\mathbf{a}_m$ and $\mathbf{b}_n$. It was targeted at the use-case of the later Chapter 5, where the user can interactively define anchor points as constraints inside a traditional SfM-BA problem, and as such it was part of our publications in 2011 [WWK11b, WWK11a]. Since then, our minimization framework has been extended and applied to many other sparse optimization problems, including the other use-cases in this thesis, namely the camera calibration (Sec. 3.8) and the marker-based SLAM with partial constraints (Chap. 4).

Parallel to our work several other publicly available sparse optimization libraries have emerged [KGS*11, Del12, WACS11, AMO]. From the discussion of the implementation aspects in Sec. 3.6.3, one can deduce several necessary properties for external libraries that are required in order to persistently ensure the feasibility of the parameter constraints as part of an optimization-on-manifold approach. Therefore, we will now discuss the existing libraries with regard to the following criteria:

- *Variable parameter types:* The library should support different types of parameters for structure and motion, respectively. In each parameter family, it should be possible to devise varying numbers of DoF for each parameter subset.
- *Distinction between local and global parametrization:* The library should support a dual representation with *local parameters* along the tangent space of the constrained parameters for the Jacobians and increments, and *global parameters* for inter-iteration parameter storage to circumvent singularities.
- *Geodesic update rule:* It should offer the possibility of providing an own update rule that maps the local increments to the global parameter space while respecting the constraints (feasibility).
- *Individual error and Jacobian evaluators*: For each measurement it should be possible to assign individual error function types and Jacobians[9]. Even if the library offers means for automatically computing the derivatives either by numeric or algorithmic differentiation, the definition of self-derived Jacobians can further improve the overall performance as discussed in Sec. 3.6.2.
- *Sparsity exploitation*: The libraries should exploit the sparsity to reduce the overall computational effort, either by means of explicit Schur complement computation (Sec. 3.6.3), via the implicit Schur complement and / or by using other sparse matrix decomposition techniques [CDH08].

Except for the last, the *SBA library* [LA09] fulfills none of these criteria. The *Parallel BA* routine (pba) [WACS11] is probably the most optimized and specialized BA solver, as it excessively exploits the possibilities offered by parallel architectures such as SIMD 2, multithreading and GPUs. Like SBA, however, it is only specialized on sparse feature-based SfM problems and only offers two types of parameters, 3D points and cameras (with and without unknown intrinsics). The other libraries offer the following possibilities as needed for the realization of the constrained optimization:

[9] In the SBA library it is only possible to define one error function type and Jacobian for the whole optimization problem, which is applied equally to all measurements.

- **g2o**: The *General Graph Optimization* framework by Kümmerle et al. [KGS*11] is designed to optimize nonlinear least squares problems that can be embedded as a graph or in a hyper-graph. Parameters are interpreted as vertices and measurements as edges or hyper edges. Own parameter types can be defined by inheritance from the `BaseVertex<.>` template with the dimension of the tangent space (DoF) and the global parameter type as template arguments. The virtual function `oplusImpl(.)` can be overloaded to define the geodesic update rule for the parameters. Measurements are derived from the template `BaseBinaryEdge<.>`[10] with the parameter types as arguments and an overload of the function `computeError()`, which computes the difference vector between the prediction $\hat{\mathbf{p}}_k$ and the actual measurement $\mathbf{p}_k$. By default, g2o computes the derivatives by means of numeric differentiation, but it is also possible to redefine `linearizeOplus()` inside the measurement class and insert own code for efficiency purposes. The g2o framework implements the Schur-complement solver inside `BlockSolver<.>` and offers additional techniques for sparse equation solving via sparse QR factorization (CHOLMOD [CDH08] or CSparse) or preconditioned gradient descent (PCG).
- **GTSAM**: *Georgia Tech Smoothing and Mapping* [Del12] is an optimization framework based on the idea of factor graphs and has strong focus on probabilistic (Bayesian) modeling. Nodes of the graph are either measurements (called *Factors*) or parameters (called *Values*). Edges represent connections / dependencies between measurements and parameters. An own parameter type can be created by inheriting from `DerivedValue<.>` with the own class as template parameter. For the geodesic update, one can overload the function `retract_(.)` and the function `dim()`. The latter defines the dimension of the tangent space. For the definition of own measurement types, one can inherit from `NoiseModelFactor2<.>` with the two dependent parameter subset types as template arguments. Inside of an overload of the `evaluateError()`, one can also specify how the derivatives are computed. In order to exploit the sparsity of the problem, GTSAM does not necessarily solve for the increments by means of the Schur complement, but rather implements an adaptive, problem specific elimination strategy, for which the order of elimination is managed by so-called *smart factors*. Unfortunately, the library is not well documented, which renders its application to novel use-cases cumbersome.
- **Ceres**: Currently, Ceres [AMO] can be considered as the most advanced open-source library to solve sparse least squares optimization problems. It was originally developed at the University of Washington [ASSS10, AFS*11, WACS11] by Agarwal et al. and is now supported by Google. The parameters for optimization can be passed on to the library as pointers to raw floating-point arrays, so it is not necessary to use any built-in class hierarchy for their representation, as in other frameworks. Measurement types can be specified by inheriting from the template `SizedCostFunction<.>` with the dimension of the residual and the dimensions of the global parameter subsets $\mathbf{a}_m$ and $\mathbf{b}_n$ as template arguments. The actual measurement values are supposed to be stored inside instantiations of this derived class. The function `Evaluate(.)` must be overloaded and should define how the residual vectors, $\mathbf{p}_k - \hat{\mathbf{p}}_k$, and the sub-Jacobians (in Alg. 3.6.1 $\mathbf{A}$ and $\mathbf{B}$) with regard to the global parameters are computed. One can also let Ceres compute the derivatives with their built-in AD-technique by using `AutoDiffCostFunction<.>` with an own, templated cost-functor class as template argument. A complete minimization problem is then constructed by adding all `CostFunction` objects together with associated global parameter pointers to an instance of a `Problem` class via `AddResidualBlock()`. In order model parameter constraints by means of tangent spaces, it is possible to create classes that inherit from `LocalParameterization`, for which the methods `GlobalSize()` (dimension of global parameters), `LocalSize()` (dimension of tangent space), `Plus(.)` (geodesic update) and `ComputeJacobian(.)` must be overloaded. The latter must

[10] A measurement represented by a binary edge is influenced by two subsets of two parameter families as in our case. In g2o, it is also possible to model measurements that are dependent on three or more parameter subsets, for which one would use the template `BaseMultiEdge<.>`.

define the derivatives of the global parameters with regard to the local parameters, which is the matrix $\mathbf{V}_{\boldsymbol{\alpha}}$ (or $\mathbf{V}_{\boldsymbol{\beta}}$) of Alg. 3.3.1, line 6. Instances of these parametrization classes can be associated to global parameter pointers by means of the function `Problem::SetParameterization()`. For completely known (fixed) parameter subsets one can use `Problem::SetParameterBlockConstant()`, i.e. it is not allowed to set the tangent space dimension to zero inside a `LocalParameterization` object. The derivatives of the residuals with regard to the local parameterization are computed internally by concatenation of the two provided Jacobians (global parameters with regard to local parameters and residual vectors with regard to global parameters) via the chain rule of differential calculus. In Ceres it is possible to choose from rich set of optimization algorithms, such as Levenberg-Marquardt (default), Dogleg, line search, and many others. It also provides many options for sparse equation solving like the explicit Schur complement method (`DENSE_SCHUR` or `SPARSE_SCHUR`), sparse Cholesky factorization using SuiteSparse / CXSparse [DHD] (including CHOLMOD [CDH08]), or preconditioned conjugate gradients (option `CGNR`).

To summarize, by now there exist sparse least squares minimization frameworks that - in contrast to the SBA library [LA09] - can be used for constrained minimization with an optimization-on-manifold approach. Although they differ by their concepts and architecture, the important part is here that they allow to use many different parameter types and error models in one minimization problem, and they also support the minimization of parameters on manifolds embedded in a higher dimensional (global) space. Thus, the strictly feasible, constrained BA proposed in this work can be realized without having to re-implement the actual minimization algorithm that exploits the sparsity of BA problems (lines 5-16 in Alg. 3.3.1 and Alg. 3.6.1). When imposing explicit constraints, it is nevertheless important to transform the parameters close to their constraint manifolds (gauge resolution, Sec. 3.4) and to ensure the feasibility of the constraints (parameter projection, Sec. 3.5) before starting the minimization.

3.8. Application of Sparse BA to Monocular Camera Calibration

In this final section, we will make a short excursion and address the process of calibrating the intrinsic parameters of a camera. Our motivation in this regard is twofold. First, a well calibrated camera is an essential prerequisite for any geometric calculation in computer vision. All the methods presented in this thesis assume that the intrinsic parameters are known. It is therefore important to be able to determine these in advance with a high degree of precision. Second, camera calibration can be formulated as a minimization problem having a sparsity structure that is very similar to SLAM or SfM problems, and thus represents another application example of our sparse minimization library. In order to suppress the influence of measurement noise and to obtain reproducible results, it is advantageous to be able to perform the camera calibration on a larger data basis (more than a few dozens of input images) while still obtaining the result within a reasonable period of time. It is therefore important that the optimization algorithm efficiently exploits the sparsity of the minimization problem.

The objective of camera calibration is to retrieve the perspective parameters of the camera, in particular the focal lengths, f_x, f_y, and the principal points, c_x, c_y, as well as the coefficients of a distortion model that describes the lens-induced deviations from an ideal perspective image projection. To this end, the calibration is typically carried out in to steps. In the first step, a sequence of images of a known calibration target (e.g. a planar, printed-out chessboard pattern) is collected with the camera from various viewpoints including oblique angles. The target pattern is detected in the images, point measurements at reference points (e.g. chessboard corners) are refined to sub-pixel accuracy, and initial estimates of the unknown viewpoints (extrinsic parameters) are coarsely estimated for each image (e.g. by using default intrinsics and a PnP algorithm). These tasks are performed in real-time and can be executed while the images and measurements are recorded and stored in memory.

In the second step, all collected measurements are used for an accurate estimation of the intrinsic parameters. Based on a model that connects points on the calibration target with point predictions in the image, a large minimization problem (BA) is solved, in which also the approximate extrinsic parameters are further refined. In the commonly used camera model by Brown [Bro71], the distortions are described by means of two-variate polynomials on normalized image coordinates with so-called 'radial' and 'tangential' terms using five coefficients, $d_1, ..., d_5$, for parametrization. Using this model the complete mapping of a point $\mathbf{x}_\mathcal{L}$ of a calibration target to a predicted image point $\hat{\mathbf{p}}$ in camera pixel coordinates can be described by the following sequence of operations:

$$
\begin{aligned}
1.\quad & \begin{bmatrix} x \\ y \\ z \end{bmatrix} && := \mathbf{R}_{\mathcal{L}\to\mathcal{C}_m}\mathbf{x}_\mathcal{L} + \mathbf{t}_{\mathcal{L}\to\mathcal{C}_m} && \text{(transf. to camera coordinate frame } \mathcal{C}_m\text{)} \\
2.\quad & \begin{bmatrix} u \\ v \end{bmatrix} && := z^{-1}\begin{bmatrix} x \\ y \end{bmatrix} && \text{(perspective division)} \\
3.\quad & \begin{bmatrix} \tilde{u} \\ \tilde{v} \end{bmatrix} && := (1 + d_1 r^2 + d_2 r^4 + d_5 r^6)\begin{bmatrix} u \\ v \end{bmatrix} && \text{(distortion mapping)} \\
& && \quad + \begin{bmatrix} 2d_3uv + d_4(r^2+2u^2) \\ d_3(r^2+2v^2) + 2d_4uv \end{bmatrix}, \quad \text{with } r^2 = u^2+v^2 && \\
4.\quad & \hat{\mathbf{p}} && := \begin{bmatrix} f_x\tilde{u} + c_x \\ f_y\tilde{v} + c_y \end{bmatrix}. && \text{(conversion to pixel coordinates)}
\end{aligned}
\tag{3.30}
$$

Here, $d_1, ..., d_5$ denote the distortion coefficients to be estimated together with the perspective pinhole model parameters. The full minimization problem aims at minimizing the error between predicted and measured 2D reference points across all M images and has the following form:

$$
\underset{\mathbf{a},\mathbf{b}}{\arg\min}(F(\mathbf{a},\mathbf{b})) = \underset{\mathbf{a},\mathbf{b}}{\arg\min} \sum_{\substack{m=1,...,M \\ l=1,...,L}} \|\mathbf{p}_{ml} - f_{ml}(\underbrace{\mathbf{R}_m, \mathbf{t}_m}_{\mathbf{a}_m}, \underbrace{f_x, f_y, c_x, c_y, d_1, ..., d_5}_{\mathbf{b}}, \underbrace{\mathbf{x}_l}_{\bar{\mathbf{c}}})\|_2^2, \tag{3.31}
$$

As indicated in the equation above and in Table 3.1 on page 42, the complete set of minimization parameters can be divided into subsets (blocks) of two distinct parameter families, $\{\mathbf{a}_1, ..., \mathbf{a}_m, ..., \mathbf{a}_M\}$ and $\mathbf{b}$. And each predicted measurement point $\hat{\mathbf{p}}$ only depends on one specific subset of each family, which allows to exploit the primary sparsity structure by means of the 'Schur complement trick' for inverting of the approximated Hessian $\mathbf{J}^\mathsf{T}\mathbf{J}$ within the Levenberg-Marquardt iterations.

The important part here is that the number of blocks for the second parameter family $\mathbf{b}$ is constant, which means that all measurements are connected to the same constant number of intrinsic parameters. So, referring back to the graphical representation of the Jacobian and Hessian in Fig. 3.3 and the complexity analysis in Sec. 3.6.3, one can see that in this case the size of the dense sub-matrix $\mathbf{F}$ in $\mathbf{J}^\mathsf{T}\mathbf{J}$ equals the number of intrinsic parameters and is also constant, whereas $\mathbf{E}$ (associated to the extrinsics) is block-diagonal and grows with the number of frames. An efficient implicit inversion of $\mathbf{J}^\mathsf{T}\mathbf{J}$ can be achieved by first solving for the intrinsic parameter increments via the Schur complement (cubic effort w.r.t the constant number of intrinsic parameters) and then by back-substituting the result to obtain the extrinsic parameter updates (linear w.r.t. extrinsic parameters). Thus, applying this elimination strategy in the context of camera calibration results in a significant performance gain, because then *the total effort in each LM iteration remains linear* with regard to the number of images, whereas standard dense-matrix solvers reveal a cubic effort with increasing problem size.

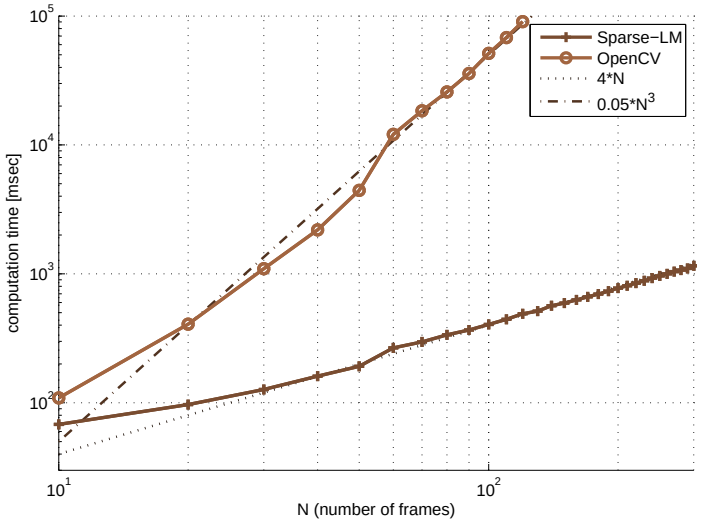


Figure 3.4.: Speed comparison between our BA implementation (Sparse-LM) and the calibration of OpenCV with regard to the number of processed images.

There exist several implementations and tools for camera calibration. Notably, OpenCV's version of the Camera Calibration Toolbox for Matlab by Bouguet [Bou] and its original Matlab implementation are most commonly used in academia.[11] However, they are not implemented efficiently as they disregard the sparseness of the minimization problem. In Fig. 3.4 we compare the mean execution times of the minimizer of OpenCV's calibration with our implementation, from which the cubic increase of computation time with OpenCV's version opposed to the linear increase for our implementation can be observed. As a consequence, with OpenCV it takes almost a minute to process 100 images and even half a day for 1000 images. By contrast, our minimizer determines the solution for thousand of images within a few seconds.

To summarize, in this section we have presented another application of our sparse minimization framework to intrinsic camera calibration. Because of its efficient implementation, it outperforms existing tools which are commonly used for this task. A fast solver with linear complexity allows to process large sets of input images and to obtain consistent and reproducible calibration results, which form the basis for all other methods presented in this thesis.

3.9. Conclusion

In this chapter, we have studied and proposed a variant to the classical BA in which known connections to a target coordinate system can be integrated as constraints for non-rigid registration. BA itself is seen today as an integral part of any SLAM system. Compared with earlier, recursive EKF-based approaches that attempt to factor out a large part of the measurements and accumulate them in parameter uncertainties (covariances), BA integrates all available measurements in a global, least-squares minimization formulation in order to obtain the

[11] Google Scholar reports that the Camera Calibration Toolbox for Matlab by Bouguet [Bou] has been cited more than 3000 times since it was released online (approx. in 2001). In 2017 and 2018 it was still cited 185 and 145 times, respectively, which indicates that it still is one of the most used software tools for camera calibration.

best possible estimation of the camera path and the reconstruction of the environment. The particularity of BA as one instance of least-squares problems solvers consists in explicitly accounting for the sparse structure of the problem ('*Schur complement trick*' or fill-reducing variable reordering) in order to allow efficient calculation of the optimal solution despite the large number of parameters to be estimated.

Typically, BA and SLAM generally have seven degrees of freedom, which is reflected in the fact that the coordinate system and the scaling are completely free. For numerical calculation purposes, an initial choice of the coordinate system must be made, but otherwise this choice is arbitrary. Therefore, for AR applications, the selected and the virtual coordinate system must be aligned with each other. However, a rigid transformation of the parameters after unconstrained BA is not sufficient. Due to the high number of parameters to be estimated and their mutual correlation, BA often represents a poorly-conditioned minimization problem. As a result, the calculated reconstruction and the camera path can exhibit low-frequency deformations, which in AR applications lead to systematic misalignments of the augmentations.

To compensate for these errors, we propose to integrate connections between the SLAM and the virtual domain mathematically as constraints to the least-squares formulation. Instead of modeling such constraints via the standard Lagrangian method, we propose a strictly feasible, optimization-on-manifold approach that uses Lie algebraic methods. On the one hand, this has the advantage of naturally reducing the number of optimization variables according to their associated constraints, instead of inflating the overall problem by introducing additional Lagrange parameters. On the other hand, the least-squares character of the overall optimization problem is preserved so that the Levenberg-Marquardt or Gauss-Newton algorithm, which has traditionally been employed successfully for BA, can still be used.

We have implemented a corresponding BA framework for this purpose. Parallel to our work, several more optimization libraries have emerged that also address sparse least-squares minimization problems or BA. We elaborated general requirements for the realization of a constrained BA of the kind intended by us. And we analyzed to what extent these existing libraries meet these requirements. In particular, a BA library must offer the following possibilities for realizing such an optimization-on-manifold approach: it must permit a dual parameterization by means of local and global variables; within each parameter family, it must be possible to model varying degrees-of-freedom among the parameter subsets; it must be possible to assign different error functions and derivatives to each error term (measuring function); and the updating step (from local to global parameters) must be overloadable. Three of the considered external libraries fulfill these requirements and can therefore be used for the iterative minimization step. In addition, regardless of the chosen minimization library, two further steps have to be carried out in advance in case the imposed constraints are expressed in the target coordinate system. First, a coordinate transformation on the parameters is necessary, so that the constraints are at least approximately fulfilled. And second, the transformed parameters must be projected onto their respective constraint manifolds. Both operations have been exemplified on several concrete cases in this chapter.

The method proposed in this chapter has been used in the applications as described in the upcoming chapters. Therein, we also demonstrate the advantage of internalizing constraints on real use-cases.

4. Marker-Based Reconstruction and Alignment

4.1. Introduction

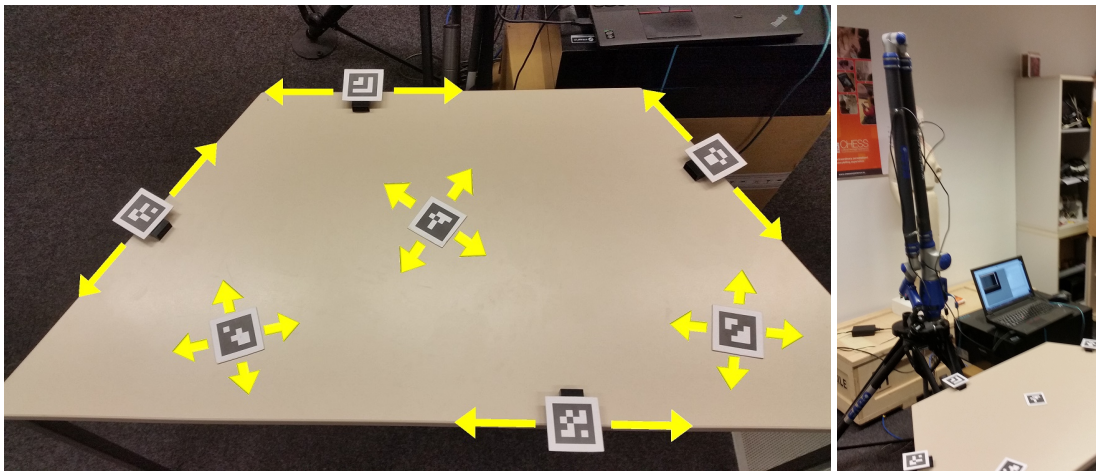
We recall that SLAM in general - whether marker or feature based - is targeted towards a *relative* ego-motion estimation in unknown environments. If no prior information on the scene is available, the coordinate system and scale for reconstructing the scene geometry and the camera trajectory is subject to some arbitrariness, as the minimized error is independent on any particular choice (gauge freedom) [TMHF00]. Augmented reality applications, however, require a geometric linkage between the reconstructed and virtual models in order to ensure that the visualized virtual content is rendered from the correct viewpoint, so it appears properly aligned and seamlessly integrated with the real environment.

In this chapter, we present a marker-based SLAM pipeline, where the reconstructed model and the camera trajectory are automatically registered to a predefined coordinate system of a given virtual model during runtime. To this end, we assume that the registration is based on partial and dispersed registration information linked to some of the observed markers. It is not necessary to know the location and orientation of any marker with regard to all six DoF. Rather, the reference markers may only contain partial associations to lines and planes of the virtual model. During setup, these markers can be placed freely on the corresponding planar surfaces or along the edges of the real object (see Fig. 4.1). The complete information for registration is encoded in the spatial configuration of whole marker ensemble.

We believe that the possibility of using partial information for registration - particularly of edges and surfaces of the environment - can greatly simplify the registration process. Planar surfaces and straight edges are very prevalent in industrial or man-made environments, and thus offer a rich source of information for registration. Furthermore, their virtual counterparts can be easily extracted from given CAD models. We will focus only on a pure marker-based SLAM. However, the idea of using partially defined reference markers for spatial registration can easily be combined with any SLAM system that additionally uses natural features.

Once the markers are initially reconstructed in the local SLAM coordinate system (Sec. 4.3), we can compute the global (rigid) transformation to the virtual coordinate system (Sec 4.4). For estimating this transformation, 3D points of the reconstructed reference markers and their associated lines and planes in the virtual coordinate system are used. This can be interpreted as an instance of a mixed point-to-plane and point-to-line absolute pose problem. Thus, it represents an application of the presented algorithms of Chap. 2 that goes beyond the classical camera pose computation from image measurements (PnP or PnL problems).

In order to further increase the accuracy, we additionally propose to incorporate the registration information into the bundle adjustment. We follow the general procedure outlined in Chap. 3: after a rigid reconciliation of the local SLAM coordinate system and the virtual coordinate system in which the constraints are defined, the parameters of the reference markers are projected onto the constraint manifolds and subsequently optimized along geodesics of their feasible (possibly nonlinear) subspaces. Markers attached to an edge or a surface are constrained in both, translation and rotation. The latter is defined by a fixed rotation axis, which must remain parallel to the edge direction or the plane normal.



(a) (b)

Figure 4.1.: (a) Exemplary setup consisting of markers which can be moved freely on the surface or along edges of an object (table). We show that this information is sufficient to achieve a highly accurate registration to a given virtual model of this object. Using a marker-based SLAM approach as back-end, it can be interpreted as a registration problem from point-to-line and point-to-plane correspondences. (b) A FARO® measurement arm with the camera mounted on it is used for evaluating the accuracy of the registered tracking and the marker ensemble reconstruction.

The presented registration method offers a quick and accurate in-situ setup of a SLAM system to an absolute reference. It was studied and developed as part of an augmented reality quality inspection system in industrial environments as outlined in the introduction (Chap. 1). In this chapter, we will evaluate the accuracy quantitatively by using a FARO® measurement arm, which generates highly precise ground-truth. The evaluation covers both rigid and non-rigid (constrained BA) registration, revealing the additional benefits of the latter.

4.2. Related Work

Marker-based camera tracking has a long tradition, in particular for augmented reality applications [KB99]. Despite the tremendous advances in natural feature recognition, markers still play an important role, especially when the scene is poorly textured or predominantly made of metallic or specular objects, such as in industrial manufacturing environments. Under these circumstances, tracking or SLAM using natural features is often not applicable or results in poor accuracy. Markers can provide a robust and precise alternative in these situations. For this purpose, a huge number of possible marker types have proposed [Fia10, BAC*16, US11, KB99, WLS08, Ols11].

Using only a single marker for tracking has limitations regarding the accuracy and the tracking range. Therefore several researchers both from robotics and augmented reality communities have proposed to reconstruct a set of multiple markers from the image data [BNR02, KUY04, KS07, LL09, YYBM09, WSHN10, NBB16, MMYC16].

Baratoff et al. [BNR02] present a system for reconstructing and tracking multiple markers placed temporarily in the scene. Based on a coarse estimate, the reconstruction is further refined. However, the optimization is carried out in an decoupled manner, switching alternately between camera and marker pose optimization. As this

results in a relatively slow convergence rate, this optimization is carried out at the end of a preparative workflow. The system is used as part of an AR application for air flow visualization in the interior of an airplane.

Klopschitz et al. [KS07] combine natural features and markers in their reconstruction. Their goal is to reconstruct sets of markers over a relatively large indoor area in order to support an augmented reality based building information management (BIM) system and to visualize subsurface structures. The reconstruction of the markers is primarily driven by the natural feature based SLAM including a global BA. Marker measurements are discarded in the first step until the full camera path is available. The marker positions and orientations are 'triangulated' afterwards given these estimates. One reason for this two-step approach could be that they used the SBA library for bundle adjustment [LA09], which does not allow to combine different measurement models and parameter types (e.g. markers as rigid bodies) in one optimization problem, as discussed in Chap. 3.

Some works consider to use extended Kalman filters (EKF) for marker-based SLAM [LL09,YYBM09,NBB16]. While the approaches of Lim et al. [LL09] and Yamada et al. [YYBM09] are purely vision based, Neunert et al. [NBB16] combine measurements from an inertial measurement unit (IMU) for a visual-inertial marker based SLAM system. As an advantage of this system, SLAM and tracking can be sustained, even if no markers are visible for a certain period of time. However, as noted earlier, most probabilistic formulations of SLAM can be translated into a (possibly weighted) sparse minimization problem [DK06] (bundle adjustment or smoothing and mapping), even if the estimation involves measurements from different modalities with heterogeneous uncertainties. There is a general consent that sparse minimization should be the preferred choice as it provides more consistent estimates of the parameters [SMD10a].

Before applying bundle adjustment, it is required to have initial estimates for the structure and motion parameters, which must be sufficiently close to the optimal solution. These can be obtained by concatenating the individual marker-to-camera poses in a 'greedy' manner. Wang et al. [WSHN10] propose a method to obtain improved initial estimates by interpreting the reconstruction problem as a graph, where the nodes correspond to marker and camera poses, and the measurements are represented by edges. Their 'best' initial estimate is obtained by chaining poses along the shortest path with the lowest uncertainty in the intermediate parameters. The most severe problem during initial parameter estimation - the marker pose ambiguity - is not addressed in their work, however.

Recently, a comprehensive marker-based online SLAM approach has been proposed by Muñoz-Salinas et al. [MMYC16]. Camera and marker poses are optimized simultaneously as a sparse optimization problem (BA) similar to ours. Their minimization treats markers as rigid bodies, and rotations are parameterized using Lie algebra. They also provide solutions for handling the pose ambiguity of individual markers during initial estimation of the marker ensemble structure. They compared their approach to recent state-of-the-art marker-less SLAM systems showing its superiority in certain situations.

All of these SLAM systems have in common that tracking and reconstruction is performed in some arbitrarily selected coordinate system, where typically one of the markers is chosen as reference for the relative pose of the others. For augmented reality applications, however, the reconstructed model needs to be aligned with the coordinate system of the virtual content. The marker-SLAM approaches, which have been used in the AR-context [BNR02, KS07] (see above), both recognize that spatial registration is necessary. Nevertheless, this part is described only succinctly and corresponding details are omitted[1]. In an earlier work, Kotake et al. [KUY04] have proposed a method to integrate constraints on the marker parameters into a Gauss-Newton-based marker BA algorithm. They consider several types of constraints such as (1) markers sharing a common but unknown plane (coplanarity), (2) markers having the same normal orientation (co-normal vector), or (3) markers placed on

[1] For example Baratoff et al. [BNR02] explain that the workflow in setting up their AR-system consists of a total of six steps, one of which is the spatial registration. In the further course of the paper, they do not give any further hints on this, but concede towards the end that a more comfortable registration tool could be part of future work.

known planes or lines. Some of their constraints represent inter-marker relations (coplanarity, co-normal vector), others are fixed to a specific target coordinate system very much as in our case. The difference is, however, that they use these constraints merely to increase the consistency (relative accuracy) of the reconstruction. They do not address how the marker setup is initially reconstructed and transformed into the target coordinate system based on the constraint information. Thus, their approach will only work, if the marker ensemble is already roughly aligned with the imposed constraints.

Opposed to previous works, we propose a self-contained marker-based SLAM approach, which includes the spatial registration to a target coordinate system. The reconstruction and registration is performed in a completely automated manner and takes full advantage of all information available for optimal alignment, provided that the user has initially associated the model-based constraints to the markers. The approach is based on the constrained BA as outlined in Chap. 3, which in turn relies on our closed-form algorithm for estimating the rigid transformation from point-to-line and point-to-plane correspondences (Chap. 2). The latter is of central importance to this application because it makes the alignment based on partial and dispersed information possible at all.

4.3. Marker-based SLAM Pipeline

At the core of our SLAM pipeline we use our sparse bundle adjustment solver presented in Chap. 3. Markers, with local coordinate system $\mathcal{L}_n$, are treated as rigid bodies and are parametrized by a Euclidean transformation (6 DoF), $\mathbf{E}_{\mathcal{L}_n \to \mathcal{L}_0}$, to some other arbitrarily selected base marker, $\mathcal{L}_0$. Similar to Muñoz-Salinas et al. [MMYC16], our optimization criterion is the squared error between the predicted and measured 2D marker corner points in the images. The minimization of all rotation parameters is performed by a local parametrization using Lie algebra. For an efficient implementation of the bundle adjustment, the sparsity of the problem is considered by means of the *Schur complement* (see Sec. 3.6.3 on page 58). Since the number of markers parameters is constant and typically small, the complexity of the BA will only increase linearly with respect to the number of frames - as opposed to cubic complexity when the sparsity is left unexploited. By further reducing the optimization only to key-frames, the BA will only consume a few milliseconds. Thus, the BA can even be carried out in the main thread without loosing real-time operability.

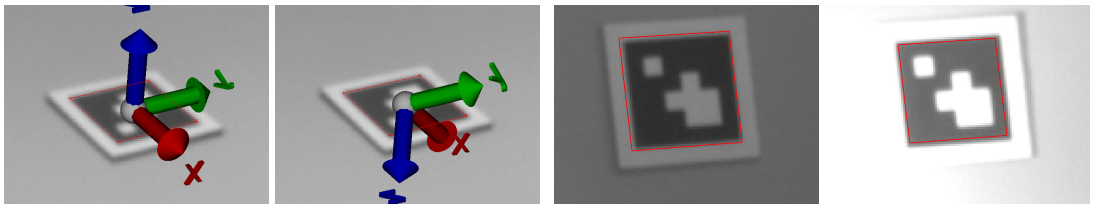
The complete marker-based SLAM pipeline consists of four major subtasks, which are executed in the following order:

- *Image preprocessing and marker detection* (on every frame),
- *Camera pose estimation* based on previously reconstructed markers (on every frame),
- *Initial reconstruction* of newly detected markers (key-frames only), and
- *Spatial registration and global bundle adjustment* (key-frames only).

For marker detection we decided to use Aruco markers [GJMSMCMJ14], which are publicly available as part of the OpenCV Library. The output of the marker detection is a set of marker identifiers and the 2D points of the four outer corners of the black marker square in the image.

The camera pose for each frame, $\mathbf{E}_{\mathcal{L}_0 \to \mathcal{C}}$, is estimated based on the subset of detected markers that have already been successfully reconstructed using the common coordinate system of the base marker $\mathcal{L}_0$. For all newly detected, unreconstructed markers, we also calculate the camera poses $\mathbf{E}_{\mathcal{L}_n \to \mathcal{C}}$ relative to their local coordinate systems $\mathcal{L}_n$. Both cases are instances of the classical PnP problem (or P4P for single markers).

Initial estimates for the reconstruction of new markers can be obtained by a concatenation of individual marker-to-camera poses. In theory, this is easily accomplished by applying the inverse of the global camera pose to the local poses of the markers: $\mathbf{E}_{\mathcal{L}_n \to \mathcal{L}_0} = \mathbf{E}^{-1}_{\mathcal{L}_0 \to \mathcal{C}} \mathbf{E}_{\mathcal{L}_n \to \mathcal{C}}$.



(a) (b)

Figure 4.2.: Particularities of the marker detection that need to be addressed during reconstruction and registration. (a) Two distinct solutions for the individual pose of a marker with almost identical error, when the marker covers only a small image region or the camera has a narrow field-of-view. (b) Lighting conditions may affect the scale at which the markers are detected (red square) depending on the used marker detection algorithm. This also results in a global scaling of the reconstruction of marker ensembles.

However, a problem which needs to be addressed is the *pose ambiguity*. This occurs whenever the markers (or planar objects in general) cover only small region in the image or the camera has a narrow field-of-view (see Fig. 4.2 (a))[2]. Even if the two solutions are considerably different, the remaining residual error of the minimized cost function may be almost identical. At this point we note that it is important to choose a PnP (P4P) algorithm which is capable of retrieving all minima at once, such as our proposed algorithm of Chap. 2. Algorithms that can only determine a single solution repeatedly pick the wrong one of the two. In our SLAM system, a pose is considered as ambiguous, whenever two solutions exist that both have a re-projection error below three pixels.

In case of such an ambiguity, the two solutions are retained as unresolved hypotheses. Fortunately, resolution becomes possible as soon as the camera movement also contains a translational component (not only pure rotations). For any reconstruction based on the wrong hypothesis, the re-projection error will increase much faster than for the correct one. In the implementation, we resort to the concept of *key-frames*, which has also been successfully used for natural feature SLAM (see PTAM [KM07] as example). For the key-frame selection we compute the angles between the viewing rays from two camera centers to each of the 3D marker points (apical angles[3]). The majority of angles must then exceed a threshold (in practice we used 5°) with regard to all other, previously tagged key-frames. In order to reconstruct a newly observed marker, we require that it must have been observed in three key-frames. We then choose the reconstruction hypothesis with the smallest error.

The final step in our marker-based SLAM comprises the rigid registration followed by the constrained BA. At the beginning, the BA operates in 'unconstrained' mode, where the only imposed constraint is that the base marker is kept fixed to its own local coordinate frame. The fixed-base-marker constraint is used to increase the convergence speed of the solver, since the fixed gauge allows us to use Gauss-Newton instead of damped Levenberg-Marquardt iterations (see Sec. 3.4 of the previous chapter). This condition is waived as soon as the rigid registration was successful and the external constraints can be taken into account. Rigid registration and constrained BA will both be discussed in more detail in the upcoming sections.

[2]For the uniqueness of the solution, it is crucial that perspective shortening effects are observable. The ambiguity always occurs when the image points can be sufficiently explained by a scaled orthographic projection (or a 2D affine image warp). In simple terms, this is the case when the projected marker square nearly forms a parallelogram and not a trapezoid.

[3]The computation of the apical angles is further discussed in the next chapter in Sec. 5.4.

4.4. Rigid Registration of the Marker Ensembles and the Camera Path

Once enough reference markers have been reconstructed, we can globally register the marker ensemble and the estimated camera trajectory to the virtual model's coordinate system. We assume that the correspondences between markers and their constraints (e.g. planes or lines) are given in advance by the user according to how the environment was instrumented.

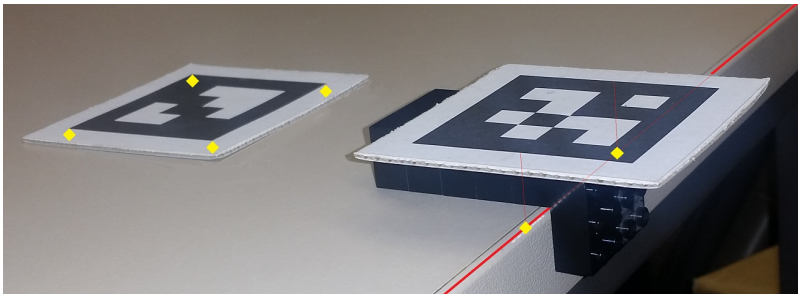


Figure 4.3.: Markers and possible adapter types used for registration. *Plane markers* (left) and *edge markers* (right) and their corresponding *rendezvouz points* (yellow).

For an easy setup, it can be a worthwhile option to mount the markers on physical adapters. For markers with associated line correspondences, we created special adapters consisting of two perpendicular surfaces (see Fig. 4.3). This allows the markers to be placed very easily and accurately at the same time. For plane-constrained reference markers a physical adapter can be dispensed with, nevertheless the material thickness between the printed pattern and the backside must be taken into account in the calculations.

The rigid registration is based on a special set of points, which we call *rendezvouz points*. Considering the rigid body consisting of marker and adapter model, these are the points which have a direct contact to the surface or the edge of the real object. There is an infinite number of such points and we choose two of them for the edge markers and four for the plane markers (yellow points in Fig. 4.3).

Having reconstructed the markers in a common coordinate system, $\mathcal{L}_0$, the registration problem amounts to finding an optimal Euclidean, $\mathbf{E}_{\mathcal{L}_0 \to \mathcal{V}}$, or similarity transformation, $\mathbf{S}_{\mathcal{L}_0 \to \mathcal{V}}$, between all *rendezvouz points*, $\mathbf{x}_{\mathcal{L}_0}$ and their associated virtual geometric entities, $\{\mathbf{y}_\mathcal{V}, \mathbf{N}_\mathcal{V}\}$ in the target coordinate system $\mathcal{V}$ according to the objective function of Eq. 2.3 of Chap. 2.

Before attempting to estimate the rigid registration, it is necessary to check, if enough reference markers have been observed and reconstructed so that the algorithm also has a chance of retrieving a correct solution. Simply counting the number of effective constraints (one DoF for point-to-plane correspondences and two DoF for point-to-line) and verifying that they exceed the number of DoF for the desired transformation (e.g. seven DoF for similarity transformations), may not be enough for two reasons. First, the used rendezvouz points may in fact be redundant, e.g. the four points of the plane marker are coplanar, so the fourth point does not add extra information. Second, such a simple check ignores the geometric configuration of the constraints. For example, three plane markers might in principle be enough to obtain a solution for a Euclidean transformation (fixed scale). However, if the three planes are parallel to one common direction, the translation cannot be determined unambiguously (see also the evaluations in Sec. 2.5.1.2 on page 29).

It might be possible to analyze all relevant geometric configurations and classify them according to their solvability, which we do not here, however. Instead, we note that the solution procedure of the proposed solver of Chap. 2 is based on certain intermediate necessary conditions for obtaining unique solutions. These can be

used as bail-out points for a quick response on solvability. First, the matrix $\mathbf{A}_{st}$ (Eq. 2.7) must have full rank (four) for the uniqueness of the linear parameters (translation and scale). Since we compute the SVD on this matrix anyway (Eq. 2.8), there is no additional overhead for doing this check. Second, the 9×9 matrix $\mathbf{M}_h$ (Eq. 2.9) after elimination of the linear parameters must also have rank four. A rank three matrix corresponds to the minimal case, and thus it will in general yield multiple solutions for the rotation. All of them will have a zero residual error, so there is no possibility to decide for the correct or 'best' solution.

These checks can also be performed during constraint definition in the setup phase of the SLAM system. The user will immediately receive feedback on whether the number and type of constraints are sufficient or whether further reference markers with model constraints must be placed in the scene to ensure a fully-defined spatial registration. The general solvability of the setup can be checked with a dry run using simulated values for the marker reconstruction.

4.5. Constrained Marker-Based Bundle Adjustment

After rigid registration, the associated Euclidean (or similarity) transformation of each marker describes a mapping from local marker coordinates $\mathcal{L}_n$ to virtual model coordinates $\mathcal{V}$, i.e. $\mathbf{E}_{\mathcal{L}_n \to \mathcal{V}}$ (see Sec. 3.4 on p.49), so it becomes possible to include the model-based constraints into the BA. The formulation of constraints is simplified, if the marker geometry is described relative to the contact surface between the adapter and the model, e.g. if one of the 'rendezvouz points' is used as origin for the local coordinate system, $\mathcal{L}_n$. Then, the two considered reference marker types, plane and line markers, comprise a combination of translational and rotational constraints as follows (cf. Sec. 3.6.1 on p.54):

- For *plane markers*, the translation is only allowed to change in two directions, which are defined by a matrix $\mathbf{V} \in \mathbb{R}^{3\times2}$ spanning the plane subspace, i.e. $\Delta\mathbf{t} = \mathbf{V}\Delta\boldsymbol{\beta}$, starting from some offset point, $\mathbf{y}_\mathcal{V}$, anywhere on the plane. Rotations are only possible around a fixed axis, $\mathbf{w}_\mathcal{V}$, which must be parallel to the plane normal, i.e. $\mathbf{w}_\mathcal{V}^\mathsf{T}\mathbf{V} = \mathbf{0}$.
- For *line markers*, the translation is bound to a single direction, $\mathbf{v} \in \mathbb{R}^3$, and an offset point, $\mathbf{y}_\mathcal{V}$, representing the line. Again, the rotation is axis-constrained with $\mathbf{w}_\mathcal{V} = \mathbf{v}$ in this case.[4]

Before optimizing the BA problem with manifold constrained Gauss-Newton iterations[5], the parameters are projected to the closest points on the feasible subsets of their associated constraints (rotation axes and affine subspaces for the translation) as explained in Sec. 3.5 on p.52.

4.6. Evaluation

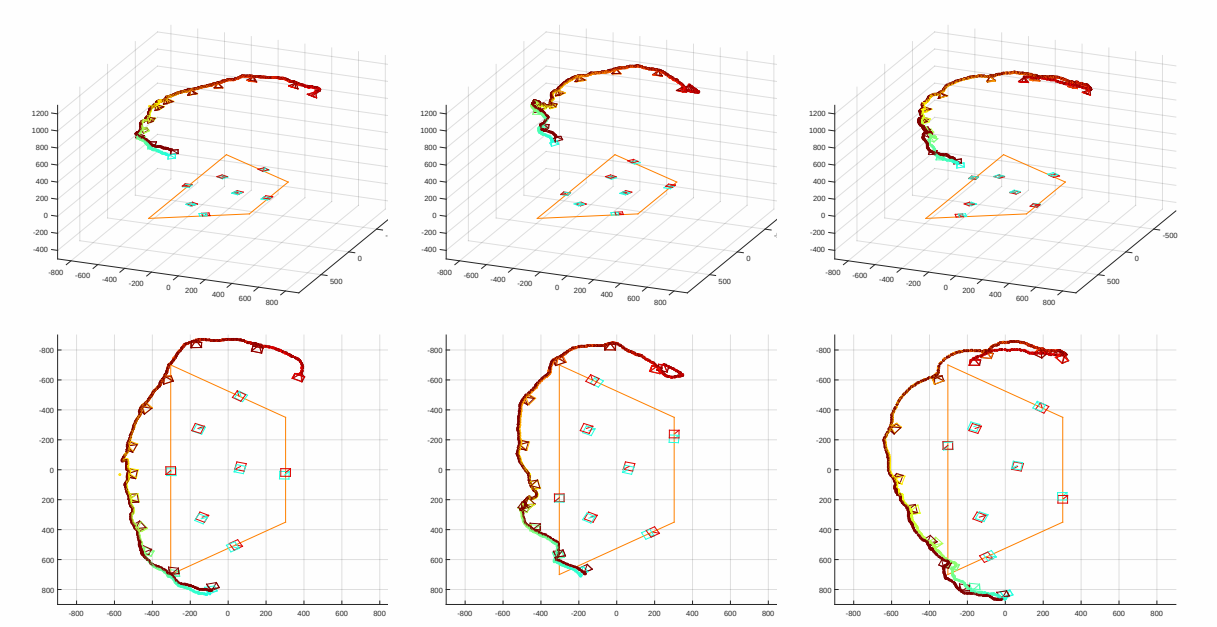
In order to point out the effects of the constrained BA, we ran the marker SLAM in two different modes for the evaluations:

- *Rigid transform mode* (Sec. 4.6.1): In this mode we reversed the order of execution. The bundle adjustment is carried out first without imposing constraints. The rigid registration is executed afterwards - every time that new results from the bundle adjustment are available, i.e. on every key-frame.

[4] We note, that the used adapters for line markers, see Fig. 4.3, in fact only allow them to be attached to the real model in two possible ways (vertical placement is also possible). So, we could also assume the rotation to be fully known.

[5] Undamped Gauss-Newton (as opposed to Levenberg-Marquardt) is possible, because the gauge is fully defined if a rigid transformation could be successfully estimated from the given constraints. Thus, the approximated Hessian, $\mathbf{J}^\mathsf{T}\mathbf{J}$, has in general full rank and no regularization is necessary before inversion or Cholesky decomposition (see Sec. 3.4).

- *Constrained BA mode* (Sec. 4.6.2): This is the default mode, where the constraints are internalized into the BA. Rigid registration and parameter projection must be done before initiating the constrained BA iterations for the first time, but only once during the entire SLAM; specifically at that point, when enough reference markers have been reconstructed. Once the feasibility of the constraints is attained, it will not be violated by the BA again later, because the parameters will evolve on their constraint manifolds.



(a) (b) (c)

Figure 4.4.: Three different evaluation runs in 'rigid transform mode', where the edge markers are placed at varying locations. The estimated camera pose is colored from cyan to red (rigid transform mode) and the ground-truth pose in brown. Camera frustums are visualized every 40th frame. The reconstructed marker positions are depicted for two time instants: at the beginning of the sequence (cyan) and at the end (red). The orange trapezoid represents the edges of the table used for evaluation. Reconstruction and registration in the 'rigid transform mode' gradually improves the more frames and marker measurements are available.

For our evaluations we mounted the camera[6] onto the tip of a FARO® measurement arm. The measurement arm is capable of providing a highly accurate full pose of the tip (including translation and rotation) at high frequency rates. However, the raw output ($\mathbf{E}_{\mathcal{F}_s \to \mathcal{F}_d}$) is in FARO®'s local coordinates of the measurement device, $\mathcal{F}_s$ and $\mathcal{F}_d$, where $\mathcal{F}_s$ denotes the static coordinate system (basement) and $\mathcal{F}_d$ is the coordinate system of the dynamically updated tip pose. So, in order to obtain ground-truth data for the camera pose in the coordinate system of the virtual model $\mathcal{V}$, we needed to perform two calibrations beforehand: a *hand-eye calibration* between the camera and the measurement arm, $\mathbf{E}_{\mathcal{F}_d \to \mathcal{C}}$, and a *scene calibration* from measurement arm coordinates to

[6] We used an IDS UEye monochromatic camera with global shutter and a resultion of 1280×1024 pixels.

the coordinate system of the virtual model, $\mathbf{E}_{\mathcal{F}_s \to \mathcal{V}}$.[7] During runtime, the ground-truth pose is given by the concatenation of these relative transformations: $\mathbf{E}_{\mathcal{V} \to \mathcal{C}} = \mathbf{E}_{\mathcal{F}_d \to \mathcal{C}} \cdot \mathbf{E}_{\mathcal{F}_s \to \mathcal{F}_d} \cdot \mathbf{E}^{-1}_{\mathcal{F}_s \to \mathcal{V}}$.

We recorded three different image sequences and ground-truth poses for evaluation, whereby we moved the camera in half-circles around the table. In each evaluation run we changed the positions of the edge markers. In the first sequence, they were roughly centered between the edge endpoints, while in the second and third sequence they were shifted in counterclockwise and clockwise direction, respectively. The plane markers were left unaltered, because we used them for further evaluations (see below). In Fig. 4.4, for the 'rigid transform mode', the ground truth and the estimated camera path are shown from the moment the markers were initially reconstructed and registered. Note, how the registration improves slowly but steadily as more marker measurements become available, and the more accurate the reconstruction by the unconstrained bundle adjustment becomes.

4.6.1. Results for Unconstrained Bundle Adjustment with Rigid Registration

We will first summarize our evaluations for the *rigid transform mode*. The corresponding quantitative results to the three evaluations runs are shown in Fig. 4.5. After each (unconstrained) BA the registration was computed as a similarity transformation including a global scale as free parameter using the *rendezvouz points* and the lines and planes, i.e. we used the homogeneous version of Eq. 2.10 (Chap. 2 p. 20). We visualized the results of all three rotation solvers of Chap. 2, i.e. UPnP [KLS14], DLS/gDLS [HR11, SFHT14] and our own solver, but they yield almost identical results. It can be observed that after 200 frames (which approximately corresponds to a 90° turn) in all three sequences an error of less than 20mm (translation) and 1° (rotation) is achieved. The final error after 440 frames is around 10mm (translation) and 0.5° (rotation).

A large portion of the remaining error is due to the fact that the final pose of the camera for each image is computed with a maximum of 28 2D-3D-correspondences (four corners of the seven markers). So, even if the markers were perfectly reconstructed and registered, the detected marker corners in the image are still noisy and therefore prevent the pose estimates from becoming more accurate. Moreover, we could not find any means for perfectly synchronizing the camera images with the measurement arm.[8] Therefore, in Fig. 4.6 we also evaluated, how well the three plane markers are reconstructed and registered. We compared their estimated center points with the center points of the real markers measured with the measurement arm. The final reconstruction and registration error is well below 2mm in all three cases. Comparing the final coordinates of the reconstructed and registered marker centers (see Table 4.1) - which excludes any remaining errors from the measurement arm calibration - we see that these values are reproduced with sub-millimeter accuracy.

We also considered the case of performing the registration with a fixed and known scale as described in Sec. 2.3.4 (Chap. 2.3.4 p. 22). This is possible, because we assume that the size of the real markers is known. This knowledge implicitly transfers to the scale to the whole marker ensemble reconstruction. Intuitively, this should produce more accurate results, since there is one parameter less for the estimation. However, depending on the used marker algorithm and parameters the opposite may be true. As shown in Fig. 4.7, the registration with fixed scale results in a clear misalignment that persists until the end of each sequence.

[7] We performed both calibrations in two separate stages. For *hand-eye calibration* we observed a checker board pattern with a 400 frames sequence from various viewpoints. Then we simultaneously optimized $\mathbf{E}_{\mathcal{F}_d \to \mathcal{C}}$ and $\mathbf{E}_{\mathcal{F}_s \to \mathcal{B}}$, so that the re-projection error in the image between the measured and transformed checker board corners, $\mathbf{x}_{\mathcal{C}} = \mathbf{E}_{\mathcal{F}_d \to \mathcal{C}} \cdot \mathbf{E}_{\mathcal{F}_s \to \mathcal{F}_d} \cdot \mathbf{E}^{-1}_{\mathcal{F}_s \to \mathcal{B}} \cdot \mathbf{x}_{\mathcal{B}}$, became minimal. $\mathcal{B}$ denotes the coordinate system of the checker board. $\mathbf{E}_{\mathcal{F}_s \to \mathcal{B}}$ was only used as an intermediate slack variable and discarded after hand-eye calibration. For *scene calibration* we measured the four corner points of the table with the measurement arm various times and averaged the results. Then we computed the relative transformation $\mathbf{E}_{\mathcal{F}_s \to \mathcal{V}}$ between the averaged points $\mathbf{x}_{\mathcal{F}_s}$ and $\mathbf{x}_{\mathcal{V}}$ with the algorithm of Umeyama [Ume91].

[8] Synchronization is only achieved in as much as we query a pose from the measurement arm immediately before triggering a new image capture, assuming that the transmission of the pixel data is the dominant bottleneck and causes the biggest latency. Effects of synchronization errors are kept small by moving the camera very slowly.

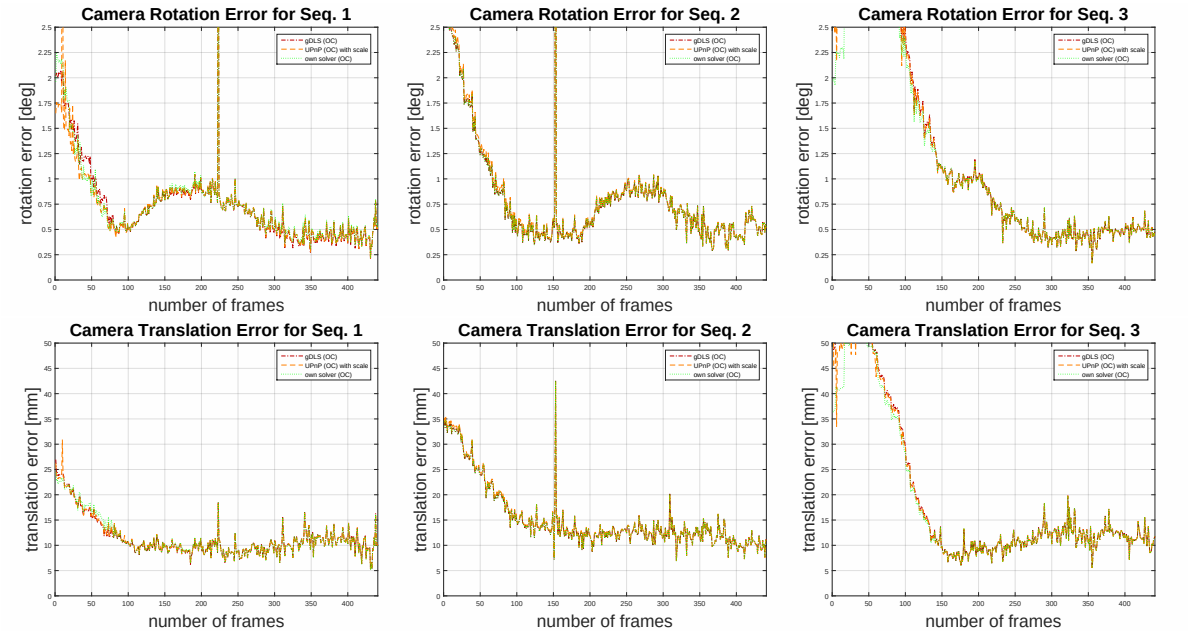


Figure 4.5.: Absolute tracking error of the camera with respect to the ground-truth data coming from the Faro-Arm. The columns (a)-(c) belong to the camera paths in Fig. 4.4, respectively. Spatial registration is performed independently with all three algorithms of Chap. 2, namely UPnP [KLS14], DLS/gDLS [HR11, SFHT14] and our own solver.

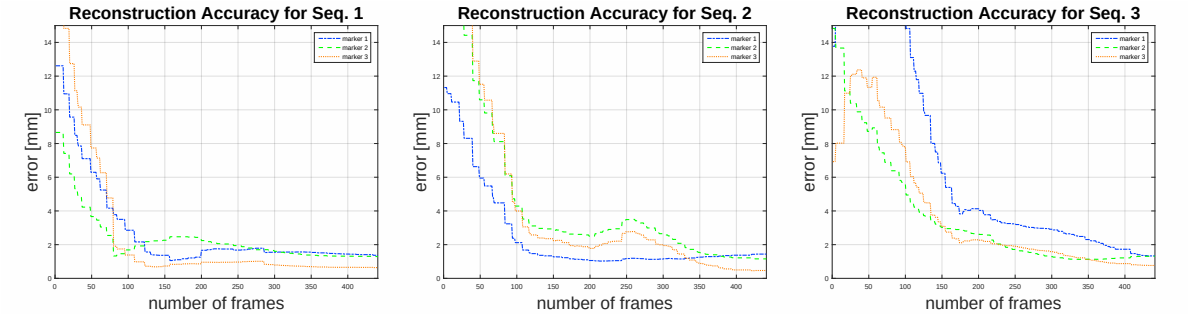
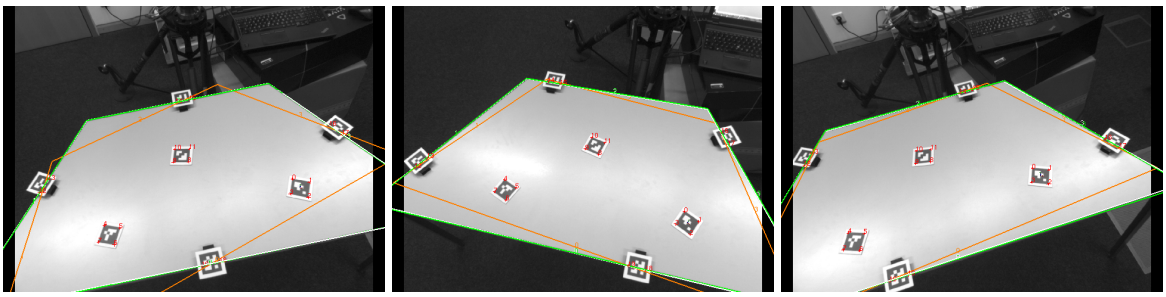


Figure 4.6.: Evolution of the absolute error in millimeters of the center points of the three reconstructed plane markers. The ground truth positions in model coordinates are obtained by measuring their locations with the FaroArm. 'marker 3' denotes the center plane marker, 'marker 1' and 'marker 2' are at the bottom-left and top-left of the table, see Fig. 4.4, bottom row.

The reason for this effect is rooted in the used marker detection algorithm, in which the black marker squares are detected by a binarization of the image with a fixed threshold. This leads to larger marker detections for darker lighting conditions and vice versa (see Fig. 4.2 (b)). The size of the detected markers also influences the global scale of the reconstruction and must be compensated during registration. Newest versions of the Aruco library are capable of performing an adaptive thresholding, which should alleviate this problem. We did not

Table 4.1.: Final coordinates in mm of the three plane markers for the three evaluation runs.

	Marker 1			Marker 2			Marker 3		
Seq.	1	2	3	1	2	3	1	2	3
x	316.597	316.545	316.099	−275.979	−275.932	−276.095	−21.649	−21.795	−22.313
y	−136.905	−136.826	−136.440	−159.139	−159.336	−159.369	64.970	64.913	64.966
z	0.912	0.853	0.967	0.946	0.956	0.883	1.315	1.464	1.312



(a) (b) (c)

Figure 4.7.: Visualization of the registration results after 200 frames in each of the three evaluation sequences. The orange outlines represent the case when the registration is computed assuming a fixed scale (using GAPS or UPnP), whereas green and white correspond to variable scale and the ground truth, respectively. The estimated scale factors for the three sequences are 1.0573, 1.0619, and 1.0604.

evaluate the behaviour in this case. However, fixed thresholding is also a common strategy in many other marker libraries, so it is worth pointing out this problem. We have shown that by leaving the scaling as free parameter, accurate registrations can still be attained despite this defect.

4.6.2. Results for the Constrained Bundle Adjustment Case

We will now present the results for the 'constrained BA mode'. The BA with internalized constraints is executed on every key-frame once the rigid registration, transformation (Sec. 3.4), and parameter projection (Sec. 3.5) have been accomplished. Due to the observed fact that the lighting conditions affect the scale with which the markers are detected, we parameterize each marker with an additional scaling parameter, so their individual reconstructions comprise a similarity transformation, $\mathbf{S}_{\mathcal{L}_n \to \mathcal{V}}$, instead of only Euclidean.

The blue plot in Fig. 4.8 depicts the error of the registered camera pose with regard to the measurement arm ground-truth. For comparison, we also added the error of the 'rigid transform mode' (same as in Fig. 4.5) in green. As can be seen, the 'constrained BA mode' attains its final error value right from the beginning, i.e. after the initialization phase after enough translation is present to resolve the pose ambiguities of the marker poses. No further 100-200 frames to collect additional measurements to increase the accuracy. These observations are confirmed for the reconstruction accuracy as visualized in Fig. 4.9 and the reproducibility plots (Fig. 4.10).

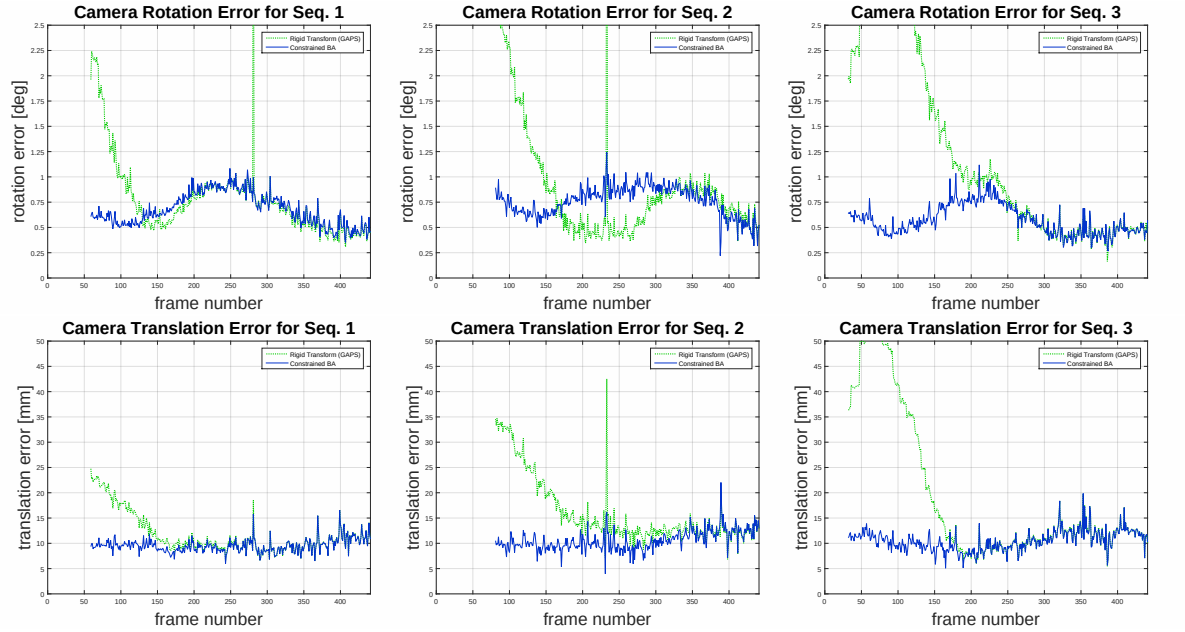


Figure 4.8.: Absolute tracking error of the marker-based SLAM with constrained BA (blue) compared to rigid registration only (green, cf. Fig. 4.5). Again, the columns (a)-(c) correspond to the camera paths in Fig. 4.4.

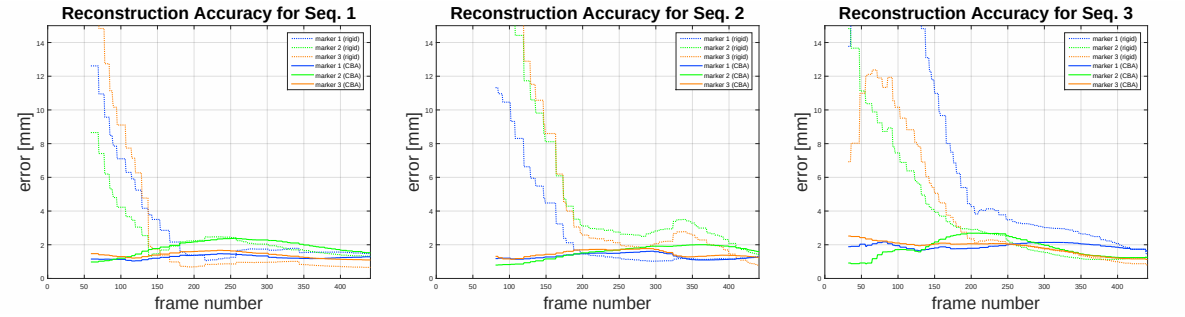


Figure 4.9.: Comparison of the constrained bundle adjustment (solid) and the unconstrained version with rigid registration (dotted line plots) for the absolute reconstruction error of the three reconstructed plane markers. The error of the center points is given in millimeters and denotes the distance to the ground truth positions in model coordinates as measured with the FaroArm. Again, 'marker 3' denotes the center plane marker, 'marker 1' and 'marker 2' are at the bottom-left and top-left of the table, see Fig. 4.4, bottom row.

We conclude this evaluation by making some final remarks on the scale computation. In the 'rigid transform mode' we have estimated a single scaling parameter for all markers. This modeling assumption represents a sufficiently precise solution for the given experimental setup, where all markers are exposed to approximately the

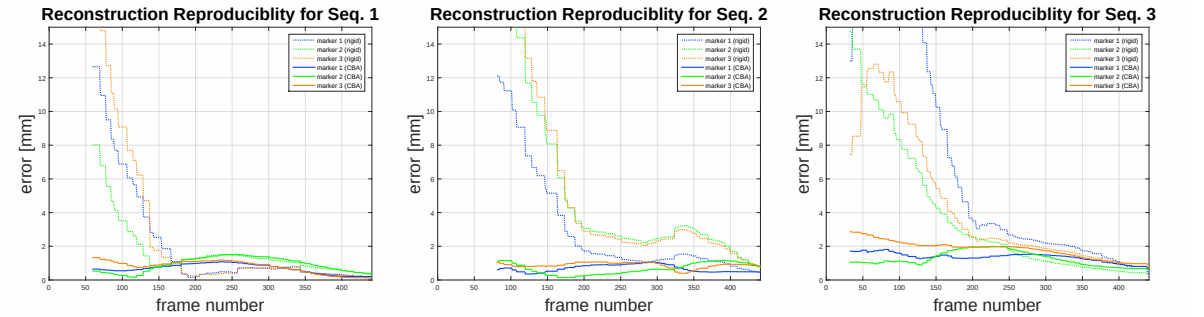


Figure 4.10.: Comparison of the reproducibility of the reconstruction between the constrained bundle adjustment (solid) and the unconstrained version with rigid registration (dotted line plots). In contrast to Fig. 4.9 we used the final, reconstructed center points of the two other sequences as reference for measuring the deviation of the reconstructed center points over time, which excludes any errors from manual measurement of the marker centers as well as calibration errors of the measurement arm to the whole setup. The plots for 'marker 1', 'marker 2' and 'marker 3' refer to the same plane markers as in previous figures.

same lighting conditions and therefore their black-and-white patterns are observed in the image with comparable brightness. In real situations, however, the markers might also be positioned in shadowed areas or they might have different orientations with regard to the main light source in the scene. As the different brightness levels also result in detections at varying sizes, an individual scaling parameter per marker - as done in the 'constrained BA mode' - is the better way of modeling. Nevertheless, scaling in general remains not more than a workaround to handle systematic errors of the used marker detection algorithm.

4.7. Conclusion

In this chapter, we presented a marker-based SLAM approach with automatic spatial registration to a target coordinate system of a virtual model. The information for registration is assumed to be associated to a few reference markers. The position and orientation for each of the reference markers only needs to be defined partially, and our approach combines this distributed information with the geometric arrangement of the whole marker setup for spatial alignment. This makes an easy setup possible as the markers do not have to be accurately measured with regard to all six DoF.

In particular, we considered two ways of how the partial links between the real and the virtual world can be specified: using markers attached arbitrarily on object surfaces or planes and / or markers placed along edges or lines. This information is optimally exploited by first computing a rigid transformation of the reconstructed markers and the estimated camera path, followed by a constrained bundle adjustment in which the translation parameters are bound to affine subspaces and the rotations are constrained by fixed axes. The presented marker-based registration thus represents a concretization of the general approach from Chap. 3. Notably, the first step, i.e. the rigid registration based on the partial constraints, is interpreted as an absolute pose problem from a mix of point-to-line and point-to-plane correspondences. It thus represents and application example of our closed-form algorithm of Chap. 2 that goes beyond classical perspective-n-point (PnP) or perspective-n-line (PnL) problems. In other words, we show that the algorithm may not only be used for camera pose estimation

from image measurements, but can also be applied to spatial alignment of a reconstructed real-world model to its virtual counterpart.

For the initial reconstruction of the markers in a common coordinate system, we follow the typical approach of determining the individual marker-to-camera poses and chaining these Euclidean transformations. A robust SLAM solution requires to handle the pose ambiguity, which is mostly left unadressed in previous marker-based slam approaches apart from one exception. For planar targets that occupy only a small region in the image, there typically exist two solutions, of which the correct one cannot be uniquely identified from a single view. This requires to use a PnP algorithm which is capable of retrieving all or at least two solutions per observation. The collected hypotheses can only be resolved as soon as the camera movement contains a sufficiently large translational component. Again, this represents an application example of the algorithm of Chap. 2, due to its ability to determine all existing minima at once.

We also identified a problem with marker detection algorithms that are based on an image binarization using a fixed threshold. In this case, the lighting conditions and the brightness levels at which the markers appear in the image have an influence on the size of the detected marker squares. As presented, a possible workaround for still obtaining accurate results consists in parameterizing the marker geometry by means of a similarity transformation, which additionally adds an individual scaling parameter per marker.

In our evaluations we showed the general feasibility of our approach. In the chosen experimental setup we attain a reconstruction and registration accuracy of one millimeter, if compared to measurements of an external device (Faro arm), and sub-millimeter accuracy for the case of cross-evaluation. Furthermore, we highlighted the additional advantages of internalizing the constraints into the BA. In this case, the same level of accuracy is achieved much earlier (directly after SLAM initialization), i.e. it requires a significantly lower total number of image measurements.

5. Reconstruction and Alignment of Feature Maps for Ready-to-Use Natural Feature Tracking

5.1. Introduction

In this chapter, we will focus on a preparative process for setting up and registering a tracking system based on *natural features*. This preparative stage is based on a pre-recorded video sequence, in which the object or scene is observed from various viewpoints. Based on the provided image sequence, all information relevant for tracking is automatically retrieved, including the geometry, appearance models, and the viewpoint-dependent visibility of individual features. This allows for the realization of a fast, robust and highly accurate ready-to-use camera tracking during runtime, for which no additional user involvement is needed once the preparation is completed.

The rigid registration (Chap. 2) and constrained BA algorithm (Chap. 3) play an important part during preparation, because they allow to transform the retrieved feature map into a desired coordinate frame. The link between the coordinate systems of the virtual model and the reconstructed feature map is established by means of a simple user interface. With that the user can browse through the sequence, select reconstructed feature points, and assign these to 3D points corresponding to a virtual model. These user-defined 3D-3D-correspondences (*anchors*) are then used for rigid transformation and as constraints inside the bundle adjustment. Having the tracking model registered and optimally aligned with the virtual model adds a substantial amount of flexibility to the creation of AR applications, since map generation and the authoring of possibly complex and animated augmentations can be decoupled from each other.

As discussed earlier, using a model-based tracking approach [WWS07, BWS06, TGBC*11] for automatic registration is not always possible or desirable due to potential limitations of such a method in certain scenarios. Therefore, in order to keep the tracking setup process as generic as possible, we do not assume that the scene contains a real object that can be tracked and whose geometry is identical to the virtual model. For the spatial registration, we only require the visibility of a small number of points which have a direct connection to the virtual coordinate system. And we resort to the capabilities of the human user to correctly identify and establish these correspondences while making use of his use-case specific contextual knowledge, e.g. regarding geometric differences between the real and virtual objects.

In general, monocular tracking for augmented reality applications is a challenging task, particularly in uncontrolled, industrial environments. The realization of a fully functional tracking system does not only require accurate and fast frame-to-frame pose estimation based on a geometrically consistent 3D model of the scene. Also, a robust tracking initialization technique is needed to support instant pose estimation from arbitrary viewpoints and to recover from tracking failures. Methods for handling limited visibility of scene parts due to self-occlusions or reflective surfaces are needed as well. All these requirements must be met in real-time, which becomes difficult when computational resources are scarce.

We address these challenges by further exploiting the preparative stage for saving computational resources during runtime of the application. We chose to use algorithms that work offline and that can be embedded in a natural way into the preparative phase, but at the same time allow a faster, more accurate, and more robust track-

ing later in the online stage. In this way, we also address the problem of dealing with limited processing power on small devices, for which usually one goal has to be traded against another. Of course, this is complemented by the possibility to use different hardware during preparation and online tracking, e.g. a PC may be used from which the tracking model is transferred later to a portable device.

For tracking initialization, Lepetit et al. [LLF05] propose to use randomized trees as a machine learning method to train a *wide-baseline keypoint recognizer* based on a large collection of different views of the object. As the main computational load is transferred into the preparative phase, they achieve a substantial speed-up compared to classical, single template based matching schemes, such as SIFT, HOG, or SURF. However, their original approach is only applicable to planar scenes (of which one image can be warped to create the training data), or a realistically textured CAD model is needed for training. In a succeeding work by Özuysal [OLFF06], the training is based on a previously captured real video sequence, but it assumes that the object's geometry can be approximated by an ellipsoid. We extend the applicability of this approach to arbitrary geometries by fusing it with our reconstruction scheme. During the online stage, this serves for fast and robust tracking initialization.

Furthermore, we train a *feature management* system [WPS07] in order to handle limited visibility of parts of the map in complex and self-occluding environments. This serves for preselecting features during runtime according to their visibility and ensures real-time capability of the tracking even for large scale feature maps.

The result of the preparative phase is a tracking model that can be directly used in AR applications, since it is already accurately registered to the same coordinate system as the virtual content to be displayed. Moreover, we show that the generated tracking model is particularly well suited for hybrid pose estimation with additional measurements from inertial sensors (IMU), which nowadays are commonly installed and accessible on mobile devices such as smart phones and tablets. By exploiting this sensor data, it is not only possible to detect fast and erratic movements, but also to reduce the number of required image features. In particular, combining hybrid tracking with a smart feature selection strategy (feature management), reveals the complementary benefits of both techniques. This leads to an even more efficient tracking, as we can show in our experiments.

The presented approach was applied in various academic and industrial research projects (see introductory Chap. 1). Furthermore, we benchmarked it against other systems from academia and industry at various tracking competitions at ISMAR conferences and other events. The results will also be presented in this chapter.

5.2. Related Work

Geometric registration for augmented reality, including camera tracking, has always been considered one of the core problems of augmented reality. Image-based reconstruction of visual-geometric representations of the scene (*reality models*) directly from the camera images have already been discussed as a promising path in the early stages of AR [KSR98], while also identifying the necessity to align the reconstructed real models with the virtual content, possibly supported by means of user-friendly, interactive tools. In the further course, SLAM or SfM has emerged as an important building block for AR.

The reconstruction of an unknown environment can be achieved via offline [SSS08, ASSS10, AFS*11, JNS*12] or online methods. While the former are targeted towards large scale (nonetheless sparse) models using batch (BA) techniques [TMHF00], online methods aim at providing the estimate of the camera movement in real-time. Online methods initially were based on recursive filtering (EKF) [Dav03, ED06] until it was shown that BA can also be used in this context, provided that a further reduction of input data to the most informative parts is ensured. To this end, Mouragnon et al. [MLD*06] propose a sliding window approach for BA (local BA), where only the most recent frames and associated measurements are used for optimization. Klein et al. [KM07] (PTAM) perform the BA optimization on the basis of key-frames, which are separated by a certain amount of

translational movement. Further, they outsource the environment mapping part (triangulation and BA) into a separate thread running in parallel, in order to not hamper the reactiveness of the camera tracking. With further simplifications, PTAM has also been shown to run on mobile devices [KM09]. These ideas (local BA, keyframes, and parallel threads) have become recurring elements in the most recent approaches to online SLAM [MAMT15, ESC14, IPSS17], in which BA can be considered as a standard technique [SMD10a]. Several surveys have been published recently for both, online SLAM [YASZ17, TUI17, CCC*16] and offline SfM [OVBS17].

Nevertheless, online SLAM still remains restricted to small workspaces, particularly on mobile devices (phones and tablets) due to memory or processing power limitations. In order to meet the real-time requirements for AR, a possible and natural solution consists of further externalizing the reconstruction to an offline preparative phase and of using a server with substantially higher computational resources for expensive tasks. Ventura et al. [VH12] reconstruct outdoor scenes with an omnidirectional camera in advance. During runtime, the server performs the global localization (tracking initialization) for a query image from the mobile client using extensive SIFT matching [Low04] together with RANSAC, and finally sends back the computed initial camera pose. Subsequently, the pose is estimated continuously on the mobile with light-weight frame-to-frame tracking techniques using the pre-reconstructed map. Due to the high latency of this system during initialization, large displacements between the query image and the current frame may occur in the meantime, so the received pose may become unusable for continuation with frame-to-frame tracking. To mitigate this problem, Middelberg et al. [MSUK14] and Ventura et al. [VARS14a] also perform a local, temporary SLAM on the mobile device. Instead of registering individual frames to the offline model, they compute a global similarity transformation using reconstructed and matched 3D points from both (online and offline) feature maps. This allows to spatially align the complete camera path of the local SLAM including the frames of the past. Since the poses of the frames of the local SLAM-model are all known relative to each other, the point in time at which the information for registration arrives on the client becomes unimportant, so that the latency is no longer an issue. Similar to our approach (as presented in our earlier publications [WWK11b, WWK11a]), they also use fixed points (anchors) to constrain the local BA and for drift reduction. However, as opposed to our case, these anchors do not come from the virtual reference model but from an independently reconstructed real-world model, which implicitly assumes that this model has absolute accuracy or is at least much more accurate than the local reconstruction.

A central aspect not covered in the approaches above - although many of them directly address augmented reality - is, how the geometrical linkage between the virtual content and the reconstructed model of the real environment is achieved. *AR authoring*, i.e. the process of populating the scene with virtual elements (annotations, highlights, billboard, etc.), essentially requires the definition of spatial relations among these objects with regard to a common coordinate system. Depending on the actual choice of the coordinate system and the degree of interlacing between SLAM and authoring, we distinguish two different strategies to define the spatial alignment of the virtual content, which we denote *SLAM-centric* and *virtual-centric* approaches.

- In *SLAM-centric* registration or authoring, the arbitrarily selected coordinate system of the SLAM algorithm forms the main reference datum for the AR application. The virtual content is interactively inserted into the recorded images or the reconstructed real world model. Typically, the placement is manually refined until a plausible or visually appealing result is achieved. Due to the dependence on the SLAM coordinate system, authoring can not be performed until the environment has been captured and reconstructed.
- *Virtual-model centric* approaches assume the existence of a static and predefined virtual reference coordinate system. This may come from a CAD reference geometry or be predetermined by some other general basis (e.g. a geographic coordinate system). The authoring of the AR content is accomplished relatively to this reference model and is therefore procedurally and temporally decoupled from the setup of the track-

ing system. However, this also requires special techniques to register the entire SLAM data including the camera trajectory and the reconstructed real world model to this reference coordinate system as well.

SLAM-centric registration: SLAM-centric approaches can be further categorized into *offline*, *online*, or *distributed* authoring approaches.

In *offline-based* SLAM-centric approaches a pre-recorded image sequence is used for the reconstruction of the real-world model and the authoring of the virtual content on a desktop environment. Skrypnyk and Lowe [SL04] were probably the first to employ SfM techniques for setting up a tracking system for AR. During the offline stage a feature map and all camera poses are reconstructed with SIFT matching. For placing virtual 3D objects into the scene (they use a teapot, a cube and a 3D billboard as examples) the user can browse through the images with a graphical interface. One and the same scene point is clicked from two different views and its triangulated 3D point is used to define the position of the virtual object's origin in the augmented scene, which afterwards can be fine-tuned together with the object's rotation. During online tracking, the offline captured feature map is used together with the transformed object for augmentation. In a more recent approach by Martin et al. [MMHM14] the reconstruction is further densified in the offline stage. The reconstructed point cloud then is visualized in a 3D editor, where the virtual objects can be placed relatively to the reconstructed ones. As in our case, the offline stage is also used to make further optimizations for more efficient relocalization during runtime. Petersen et al. [PS12, PPS13, PS15] present an AR authoring system, which not only allows to extract a static model of the real environment and to place virtual annotations relative to it. During a training phase of the system, they also capture manual workflows and scene changes that typically occur in assembly or disassembly tasks of mechanical objects. These are then compared at runtime with the performed work to give feedback to the user on the correct execution. All virtual content including annotations and the workflow captures is defined relatively to a simplified real-world model consisting of reference images (views) of the training scene (no full SLAM is used).

Online or in-situ methods allow to author and register the virtual content on the mobile system during SLAM and possibly even at runtime of the application. In order to simplify the placement, the number of adjustable DoF for each virtual object is further reduced by making use of special assumptions or additional sensory data. Reitmayr et al. [RED07, RLW*10] present a SLAM-based system, which semi-automatically extracts and reconstructs 'hyper-features' from the images. These are rectangles, polygons, or ellipses in the scene, for which it is valid to assume that they have a planar 3D surface shape inside their outlines. The knowledge of the surface normals may simplify the placement of 3D content in certain cases, e.g. annotation arrows are automatically oriented parallel to the surface normal and can be placed by simply tipping on a point near such a hyper-feature. Inside the PTAM approach [KM07], Klein performs a dominant plane regression using a subset of reconstructed feature points, which most likely belong to the floor ground, and on which animated virtual scenes can be easily attached to. In the in-situ indoor authoring approach of Kim et al. [KRW13], a room is approximated by a cuboid. With a gyroscope supported panoramic tracker, the user first interactively determines the dimensions of this cuboid by marking horizontal intersections between the walls, floors and ceilings in the images. Then, the annotations or virtual objects can be attached to walls or placed on the floor. Using the smartphone built-in accelerometer sensor, all data including the room cuboid is automatically aligned to gravity, which simplifies the placement and orientation. Similar ideas are exploited and further elaborated in the IKEA Place App [Int17] based on Apple's ARKit [App17] full SLAM approach. Here the end-user can interactively place virtual furniture models into the live view of his home, in order to asses their fitness before actually buying the respective real furniture. The SLAM system also determines the floor ground level by plane regression and makes use of the accelerometer sensor to obtain an estimate of the correct scaling and the gravity direction. The placement of the digital furniture models thus only requires a horizontal shift in two directions and a rotation around the estimated gravitational vector.

The third variant of SLAM-centric systems (*distibuted registration*) refers to AR-based remote collaboration between two or more people. Gauglitz et al. [GLTH12, GNTH14] present an AR system for remote maintenance, in which a field operator is assisted in his task by a remotely connected expert. The expert can place simple annotations (virtual crossmarks or 2D spheres) into the view of the field operator in order to guide him through a sequence of steps. The live video feed and the SLAM model of the real environment, which is automatically reconstructed from the field operator's movements, are both shared with the remote expert. The authoring is performed by the remote expert on a desktop computer. Several techniques are possible to achieve independence from the field operator's current view [TB15], in order to ease the accurate placement of the annotations; he can freeze the current view, browse back and forth within the image sequence recorded so far, or he can even visualize a densified and meshed 3D model in a 3D editor. The virtual content is placed relative to the coordinate system of the SLAM. On the client's side these virtual objects then appear appropriately based on the current viewpoint of the operator.

If the SLAM uses only one monocular video feed as input, the quality of the reconstructed environment and camera path strongly depends on the end-user's movements with the device. Monocular SLAM always requires a certain amount of translation ('*SLAM dance*'). If the motion is dominated by rotations around a fixed observer position, it may lead to false reconstructions or even a complete tracking failure. It is not only difficult to explain the type of allowed movements to an inexperienced end-user, but also the acceptance of the AR-application is reduced significantly, simply by the fact that his movements are constrained. Typically the AR application should help the end-users to achieve their personal goal, but they should not become distracted trying to help the tracking algorithm to work. Some authors have addressed this problem [GSV*12, PSR13], but it remains a challenging issue, unless the SLAM is complemented with additional sensory input like depth cameras and IMUs, as done in Microsoft's Hololens [Mic16], Apple's ARKit [App17], and Google's ARCore [Goo17].

From a general perspective, SLAM-centric authoring approaches suffer from the following drawbacks, which arise due to the fact that the user has little or no control about the coordinate system in which the SLAM model or tracking data is built. The possibility of inserting the created virtual 3D content into this extracted visual-geometric representation of the real scene allows a quick creation of AR-applications for demonstration purposes. But it also restricts its applicability in practical scenarios, because then the created virtual content becomes highly dependent on this tracking model. As a result, the reusability of the authored content may be impaired, in particular when the corresponding real world model becomes invalid. For example when drastic changes in the scene occur, or the lighting and other environmental conditions alter significantly, the scene may no longer be recognized based on the captured tracking model. Also, if the generated virtual AR content is to be shared with other people, it is insufficient to provide only the content alone; the corresponding tracking model must always be shipped along with it. Moreover, aligning the 3D content so that it looks plausible or visually appealing is often insufficient to meet industrial requirements regarding accuracy. It becomes a subjective measure, and mostly dependent on how much effort the user is willing to invest in aligning the content. In particular, when AR is used in quality inspection, the accuracy of visualized virtual content often needs to be higher than what can be perceived by human operator. Otherwise it may become impossible to decide whether any discrepancies are the result of a misplacement of the content by the author or whether they can be attributed to quality faults of the product to be checked. Furthermore, SLAM-centric authoring approaches conceptually do not offer any means for correcting errors of the reconstructed SLAM model, since the SLAM model itself is treated as the authoritative source of truth. As a consequence, virtual objects may tend to float around their designated positions, when the camera pose is affected by systematic errors due to low-frequency map deformations of the SLAM model. Finally, existing desktop-based PLM systems, 3D modeling software or gaming engines nowadays comprise sophisticated methods and workflows for efficiently generating 3D content. Rather than relying on proprietary AR authoring tools tailored to specific use-cases or the chosen tracking approach, it seems more promising to build the AR authoring process upon these existing solutions and possibly extend their functionality. It should

be possible to author the content and to setup the tracking data in parallel and to seamlessly exchange one or the other, while having one target coordinate system as the main common interface.

These considerations make it necessary to decouple the authoring process of the virtual content from the generation of the tracking model and to provide means to accurately and reproducibly register the latter to a desired and predefined target coordinate system.

Virtual-model centric registration: Virtual-model centric approaches aim at establishing links to a predefined reference model and to transform and / or to directly reconstruct the captured model of the environment with regard to the given coordinate system. The extent to which the registration can be accomplished completely automatically depends on the amount of available information and assumptions that can be used for this purpose. For *indoor applications* often detailed and precise CAD models are available, but generic CAD-model-based tracking approaches still require end-user support for tracking initialization during runtime of the application. By contrast, in *outdoor applications* a variety of additional sensory inputs (e.g. GPS, compass), data sources (e.g. Google Street View), and application specific assumptions (e.g. Manhattan-world assumption in urban environments) are available and can be exploited to bootstrap and automatize the registration.

In *indoor applications*, CAD-model-based tracking can be one possibility for registering the SLAM model to a predefined coordinate system. There is however a discrepancy between what is needed for the visual tracking and recognition system and what is typically available as CAD data. One could e.g. use well-established feature-based ad-hoc registration techniques (using e.g. SIFT, SURF or any of the more recent descriptors), if a fully and realistically textured model of the scene was available (e.g. if obtained by 3D scans [RPS*18]). But in practice, readily available CAD models usually only contain geometric information. One way of registering purely geometric models to image data consists of matching lines segments or contours from model silhouettes or crease edges to strong intensity gradients (brightness jumps) in the image. In the classical RAPiD approach by Harris and Stennet [HS90, DC02, WVS05, WWZ*15], a 3D line model is projected with a pose prior into the image and correspondences between equidistant sample points on the lines and adjacent strong gradient point in the image are established. The initial pose is iteratively refined by minimizing the distance of each gradient point orthogonally to each line segment direction and by updating the correspondences in each iteration. The Chamfer matching approach [LTVC10, CC12, AAL*09] consists of pre-calculating a distance map for each pixel position to nearby high gradient points or extracted image line segments. Pose estimation or refinement involves the minimization of the normalized sum of all distance values over all pixel positions on which the projected line model falls onto. Both, Chamfer matching and RAPiD-based tracking approaches are not directly applicable to polygonal mesh-based CAD data. Rather, they require the availability of a line model containing the most informative inner and outer contours of an object. Wuest et al. [WWS07, WS07] present an approach which automatically achieves this conversion using standard graphics hardware. The model is rendered from the current view, whereby the depth and normal buffers are analyzed for depth discontinuities (silhouette edges) or abrupt surface orientation changes (crease edges), from which 3D line segments are constructed by connecting adjacent pixel positions. The line model is updated in every frame based on the most recent camera pose, which handles elegantly the problem that only a fraction of all model lines may be visible due to self-occlusion. As an alternative to edge-based tracking, Seo and Wuest [SW16] proposed a 'direct' approach for frame-to-frame tracking of Lambertian objects by modeling intensity variations as a function of model surface normals.

As described in the introduction (Chap. 1), the stability of edge or line based tracking approaches may deteriorate in cluttered environments or when the object itself is highly textured, so that the algorithm may be deceived by wrongly matched high-gradient points in the image. Moreover, model-based tracking approaches generally only have local convergence and require a good estimate of an initial pose. They allow for recursive tracking on a frame-by-frame basis. Ad-hoc registration (in a tracking-by-detection sense) using purely geometric models is

to our knowledge still an unsolved problem. In practice, this results in the user having to move to a predefined position and to superimpose a visualized line model onto the real object in the camera image for the purpose of registration. Thus, some manual effort always remains left and is imposed on the end-user at runtime.

Nonetheless, attempts to integrate model-based tracking with SLAM exist. Bleser et al. [BWS06] combine the RAPiD-based tracking approach [WVS05] with an EKF-based SLAM system. The SLAM starts as soon as the CAD model has been successfully recognized and tracked. The camera pose computed with the model-based tracker initially comprises the main source for triangulating landmarks on the object or in the surrounding scene. Thus, the SLAM model is reconstructed right from the beginning in the coordinate system of the CAD model. The approach of Tamaazousti et al. [TGBC*11] integrates CAD model data into a BA-based slam. Within the minimization of the camera trajectory and reconstructed scene points, features that presumably belong to the object are constrained to stay on the CAD-model's surface. Again, a good initial alignment of the camera at the beginning is important, otherwise feature points may be assigned to incorrect parts of the model and the algorithm may fail to converge to the correct minimum.

For *outdoor applications* there exist a variety of approaches which aim at registering a reconstructed real-world model, the camera path, or even only single images to a global geographical coordinate system. These approaches extensively make use of application-specific assumptions or additional data sources, which are only available outdoors.

Some authors consider registration to a pre-recorded and geo-referenced image database (appearance-based methods). Schindler et al. [SBS07] proposed an approach to geo-localize individual images to an own database of 20 kilometers of urban streetside images tagged with GPS and compass heading data. For each query image a set of most similar reference images is retrieved by means of global appearance descriptors (vocabulary tree [NS06]), from which a ranked list of 2D geo-location hypotheses (including heading) can be obtained. The reference database is collected by driving a vehicle with mounted cameras and sensors through a large city. Zamir and Shah [ZS10] avoid the need to create a large geo-referenced data set in advance by referencing to publicly available panoramic Google Street View (or Microsoft StreetSide) images. Instead of using global image descriptors, every SIFT feature votes for a location, thus creating a probability map of possible 2D locations in urban areas. Their final localization accuracy is in the range of tens to hundreds of meters. Vaca-Castano et al. [VCZS12] improve the accuracy to one to thirteen meters by considering video segments that follow a continuous motion. Their probabilistic approach includes a motion model (constant velocity) that allows to fuse the localization results of multiple images in the sequence within one probabilistic model. Since the accuracy of public image databases is not better than GPS estimates (which can be particularly low in urban canyons) Taneja et al. [TBP12] refine the accuracy of the associated pose of Google Street View images using precise cadastral 3D models and by jointly optimizing the pose and a semantic image segmentation. Strecha et al. [SPF10] proposes an approach to create geo-referenced urban image databases using handheld mobile devices or public internet photo collections. They divide the whole dataset into many smaller subsets, which are first reconstructed independently using SfM techniques. Then, all local reconstructions are registered to each other and to a geo-referenced 2D urban GIS maps (building outlines). Each local SfM reconstruction (cluster) is parameterized with a similarity transformation (7 DoF). All parameters of each of these transformations are optimized with a weighted, probabilistic cost function, in which many measurement cues are fused into, including GPS measurements and geo-tags of images, up-vector estimates (taken from the assumption that most feature points in urban photos reside on vertical building façades), GIS building outlines and elevation data.

Since appearance-based registration approaches are susceptible to varying lighting conditions or clutter such as movable objects (e.g. cars, pedestrians, varying vegetation), several works propose to perform the registration using purely geometric reference models. One idea consists of matching horizon outlines to digital elevation data in case that the horizon contour is distinguishable enough. Baatz et al. [BSKP12, SBK*16] propose and approach to geo-localize individual images of mountainous landscapes (Alps). They obtain a horizon outline via

image segmentation (sky versus ground) and match it to a large database of pre-generated synthetic mountain model contours rendered at equidistant positions. For the matching, the shape of the contours is encoded into a bag-of-words like descriptor and top-ranked matches are verified geometrically afterwards. In order to solve the same problem, Baboud et al. [BvES11] require the availability of the exact GPS location. The remaining three DoF of the rotation are retrieved by correlating the edge filtered input image with a synthetic panorama of mountain model silhouettes. For navigation in urban environments, Ramalingam et al. [RBSB10] explored a similar approach. The authors segment skyline outlines in images of an upward facing omni-directional camera and match them against rendered silhouettes of an untextured digital 3D city model for estimating the 2D position of the observer on the ground. Like the CAD-based 6-DoF tracking approaches (see above) the approach requires a good enough estimate of the current position for rendering and therefore it is only meant to work locally on a frame-by-frame basis.

For automatic ad-hoc registration (without the need of a pose prior) to geo-referenced 2D building outline maps or untextured 3D city models, many approaches rely on the availability of built-in sensors of mobile devices (GPS, accelerometers for gravity, and magnetic compass) to bootstrap registration of single images or the SLAM. Also typical is the (quasi) Manhattan-world assumption for urban environments, i.e. that building edges are either vertical or horizontal, so that the corresponding 2D edges in the image intersect at dominant vanishing points (VP). Chu et al. [CGC14] use the VP to determine the orientation of the camera w.r.t. gravity and for image rectification ([Gal05]). The vertical position of camera is assumed to be at a typical human's height (1.6m). The remaining 3 DoF (one rotation angle and two horizontal position coordinates) are determined by forming triplets of vertical image lines and by establishing putative matches with nearby building corners of a 2D city outline map. Among all pose hypotheses the one closest to the GPS position is selected. Arth et al. [APV*15] presented a conceptually similar approach comprising several improvements on robustness. In particular, they use semantic segmentation techniques (window detectors and façace classifiers) for suppressing clutter that deteriorates vertical building edge detection and for final pose verification. Using a convolutional neural network, Armagan et al. [AHRL17b] segment the image into four classes, namely horizontal and vertical building edges, façades and background (sky and ground) and iteratively refine the pose (again only three DoF) by maximizing the overlap between segmented image regions and corresponding segmentation masks rendered from 3D building models. Iterations are started from several randomly sampled positions around the initial GPS coordinates. In another publication, the same authors [AHRL17a] replaced the iterative procedure with a minimal solver using either three vertical building edge correspondences or two edges and one surface normal as input. To this end they trained another neural network that predicts the façade normals in the image. Finally, a score on the segmentation overlap is used for pose validation. Bansal and Daniilidis [BD14] present a purely geometric and combinatoric approach for image registration to urban elevation models based on putative correspondences at extremal building roof points. A central component is a formulation of the pose estimation problem that only requires a single putative correspondence as input. This is achieved by combining complementary assumptions and information of various sources: firstly, they assume that the observing camera is located at an average human's height from the ground floor; secondly, they require an estimate of the camera tilt (e.g. by means of an IMU's gravity vector); and thirdly, each detected image corner point is also associated to a direction vector of a nearby edge (possibly belonging to a horizontal 3D building edge). Their single-correspondence pose estimation algorithm allows to implement an efficient voting strategy to filter out incorrect matches from the huge set of all combinations between image points and potential extremal roof points of the digital city model. The pose is finally refined using the reduced set of correct matches and verified with a semantic image segmentation method. In this approach, the user must aim the camera upwards towards the roofs of the buildings.

5.3. Approach Overview and Contributions

Our approach for creating accurately registered visual appearance-based tracking models can be considered as a hybrid method sharing elements of both, user assisted SLAM-centric and automated virtual model-centric approaches.

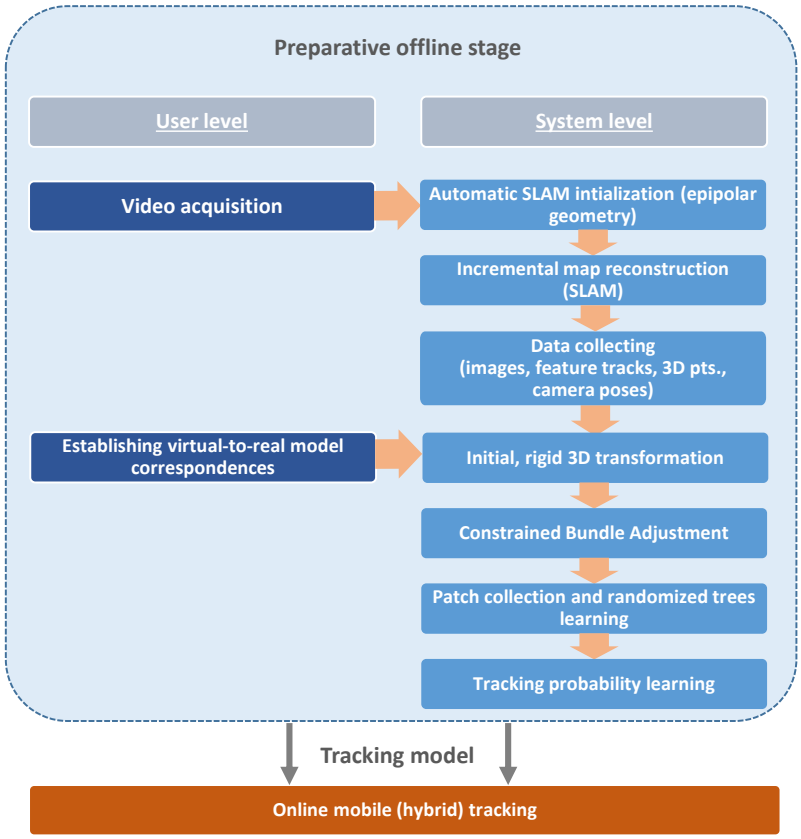


Figure 5.1.: Preparative feature map retrieval process. The result is a 3D tracking model, which is accurately registered to the coordinate system of an existing virtual model. It includes the geometry, the appearance and the view-dependent visibility of individual feature points.

Like model-centric methods, we present a technique to transform or register the reconstructed SLAM model of the real environment to a predefined or desired coordinate system. Authoring of AR content and capturing of the environmental model therefore remain decoupled from each other. These steps can be performed in parallel and their outcome can be exchanged seamlessly (e.g. for updating the tracking model due to changed environmental conditions). As discussed above, a fully automatic registration in previous works is achieved only, if it is possible to exploit use-case specific assumptions, complementary data sources, or additional sensor inputs, such as the (quasi) Manhattan-world assumption in urban environments, assumptions on the observing camera's height from ground to reduce the parameter space, applicability of segmentation algorithms (sky / ground / façade pixel

labeling in outdoor applications), or the availability of pre-registered, public image databases (e.g. Google Street View).

Our proposed approach does not depend on any of these prerequisites. Similar to SLAM-centric approaches, we rather let a user provide the necessary information for registration, so he maintains in control over the output and quality of the registration. This makes our approach applicable to a wide range of use-cases including indoor and outdoor scenarios. Furthermore, we exploit the provided input to the best possible extent, as we tightly integrate the registration information into the geometric scene reconstruction by means of a constrained BA (Chap. 3). This enables to correct low-frequency deformations of the reconstructed scene due to drift accumulation of SLAM. The correction of these systematic errors is crucial to ensure a highly accurate tracking at runtime. Thus, our system combines the complementary strengths of a human operator, who understands a scene and can make informed decisions about it, and numerical optimization techniques, which quantitatively make use of this information for highly precise registration.

Input to our system is a pre-recorded image sequence of the environment to be tracked. Since the capturing of this sequence is part of the preparative stage, we can assume that this is accomplished by a knowledgeable and trained person, who ensures that the camera motion contains sufficient translational movements for reconstruction. For registration, we require 3D correspondences between the SLAM model and the target coordinate system, for which we developed an easy-to-use editor for correspondence annotation. Beyond that no further user input is required. The system performs the constrained BA, feature learning for wide baseline tracking initialization, and feature visibility analysis automatically. This results in a generic and easy-to-use setup procedure for quick setup of an AR tracking. Fig. 5.1 provides an overview of all the processing steps inside the preparative phase and indicates at which stages the user is involved.

In the following, we will discuss all components of the system in more detail. In Sec. 5.4 we describe the incremental map reconstruction with a SLAM technique on a calibrated image sequences provided by the user with an particular emphasis on the automatic initialization. Sec. 5.5 is dedicated to the rigid transformation and optimal alignment (constrained bundle adjustment) of the initially reconstructed map using a small subset of known points. Subsequently (Sec. 5.6), we explain how both the reconstructed map and the image sequence can be used to train a randomized trees feature classifier for wide base-line tracking initialization. Sec. 5.7 focuses on retrieving tracking probabilities from the image sequences, such that a feature management system in a frame-to-frame tracking approach may dynamically allocate the expensive, image-based operations to only those features with a high tracking probability. Sec. 5.8 is devoted to sensor fusion based tracking system at runtime. After presenting evaluations (Sec. 5.9) and a report on the results of the major tracking competitions we participated at (Sec. 5.10), a conclusion is drawn (Sec. 5.11).

5.4. Structure Initialization and Incremental Map Building with SLAM

The initial map building for our tracking system is based on an image sequence, where the camera is moved around the target object or environment, observing similar positions and viewing directions than those that probably will be encountered during runtime of the augmented reality application. We make the assumption that the camera movement is smooth with overlapping features in succeeding frames and on average contains enough translation (no long purely rotational movements). In practical situations this is not a restriction, since the preparative procedure may be performed by a trained person.

For the incremental map reconstruction, we use the SLAM approach by Bleser et al. [BWS06], although in principle any other method might be used, as well. It uses affine, inverse-compositional Lukas-Kanade (KLT) algorithm for 2D tracking of feature points, which usually provides more accurate estimates of the feauture's

2D position than wide-baseline matching approaches based e.g. on SIFT. Newly initialized [ST94] and tracked features are triagulated, refined and recursively fed back for pose estimation by means of extended Kalman filter (EKF) techniques until the end of the image sequence is reached.

The absence of any prior information on the scene has the consequence that the SLAM must be initialized using the image data alone. In previous approaches, e.g. by Klein et al. [KM07, WKR07, KM09], an initial structure estimate is obtained by epipolar geometry algorithms using two key-frames which are well separated by sufficient translational movement. Additional user interaction is needed at the beginning of the SLAM, because the key frames need to be indicated manually by pressing twice a key. In order to further facilitate the preparation phase, we address the problem of structure initialization by means of an *automatic* translation detection mechanism. This is often useful when user interaction is not possible or when the camera movement cannot be controlled anymore, e.g. when map building is performed on a sequence which has been captured remotely.

Our reconstruction pipeline is automatically initialized by estimating the essential matrix E. By theory one can use the five-point algorithm by Nistér [Nis04]. We choose to use the traditional eight-point-algorithm [Har97a], where we can disambiguate between planar and general scene geometries based on an analysis of the Eigenvalues (in a similar way as this was done for the PnP problem by Lepetit [LMNF09]). In the planar case the homography H is calculated. After decomposing E or H into camera rotation and camera translation, the 3D positions of the tracked feature points can be reconstructed by triangulation.

In order to ensure the correctness of the SLAM initialization, a validation is required in order to detect, if both key-frames are separated by enough translational movement. Otherwise the reconstructed 3D points become unreliable. It should be clear that the absolute length of the decomposed translation vector itself cannot be used directly to make this decision, since it is (as well as the reconstructed geometry) only defined up to a scaling factor. It is therefore better to use the *apical angle*, ϕ_n, which is formed by the rays connecting the reconstructed 3D point with the two camera centers. Torii et al. [THPL08] present a method to approximate this angle by using the translation vector, $\mathbf{t}$, and the relative point depths, λ_n^p and λ_n^q, of the 3D point after triangulation via the formula $\phi_n \approx \|\mathbf{t}\|/\lambda_n^p$. This becomes problematic, if the motion only consists of a rotational component, since both, the translation vector and the triangulated point depths, are not well defined for this situation.

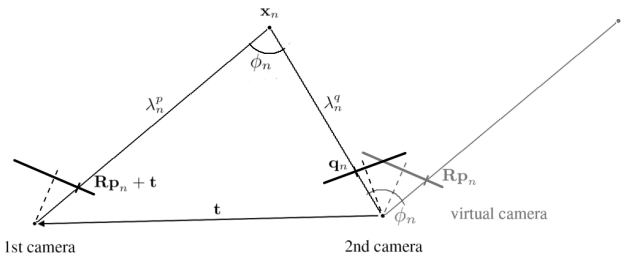


Figure 5.2.: Illustration of apical angle computation.

Here, we propose an alternative method for apical angle computation that is not an approximation. Moreover, it depends only on the estimated rotation, $\mathbf{R}$, which can also be determined for degenerate situations (planar scenes or purely rotational movements) using the Homography $\mathbf{H}$. We exploit the property that the apical angle at a point $\mathbf{x}_n$ is equivalent to the angle formed by the homogeneous point, $\mathbf{q}_n$, in the first image and the rotated point, $\mathbf{R}\mathbf{p}_n$, in the second one ($\phi_n = \measuredangle(\mathbf{q}_n, \mathbf{R}\mathbf{p}_n)$). To see this, recall that the homogeneous image points lie on the rays which pass through the 3D point and the corresponding camera centers. This leads to the following relation,

$$\lambda_n^q \mathbf{q}_n = \lambda_n^p \mathbf{R}\mathbf{p}_n + \mathbf{t}, \tag{5.1}$$

where $\lambda_n^p \mathbf{p}_n$ and $\lambda_n^q \mathbf{q}_n$ can be interpreted as the point's 3D coordinates in the first and second camera coordinate frame, respectively. A parallel shift of the ray on the left hand side by $(-\mathbf{t})$ (simply dropping the translation vector) will not change the angle of intersection, only its location (the rays will intersect at the origin of the second camera). This situation is illustrated in Fig. 5.2 by the 'virtual' camera placed at the second camera's origin. In short, the apical angle can be obtained by the following relation:

$$\cos(\phi_n) = \frac{\mathbf{q}_n^\mathsf{T} \mathbf{R} \mathbf{p}_n}{\|\mathbf{q}_n\| \|\mathbf{p}_n\|}. \tag{5.2}$$

Note that this relation is not restricted to planar point configurations, thus it may also serve for translation detection in epipolar-based reconstruction.

The initial reconstruction is accepted, if the average angle, $\bar{\phi} = \frac{1}{N} \sum \phi_n$, exceeds a threshold[1]. For higher accuracy the initial reconstruction is refined with an intermediate bundle adjustment, before the incremental SLAM algorithm starts.

5.5. Spatial Registration of Feature Maps

5.5.1. Incorporating User Knowlegde at Anchor Points

Once the incremental map building is finished, a 3D tracking model of the environment is available. However, as this monocular-based reconstruction is performed without any a priori information, the 3D coordinates of the reconstructed map are defined in an arbitrarily selected coordinate frame having seven degrees of freedom. We describe how an optimal rigid transformation of the 3D feature map into a desired coordinate frame can be realized, given the knowledge of the true coordinates of only a small subset of points in the map, such that the coordinate frame of the feature map and the coordinate frame, in which the augmentations are defined are coherent. Furthermore, the same points are used as constraints for the bundle adjustment variant of Chap. 3. This allows to correct non-rigid deformations caused by drift accumulation during reconstruction.

In general, our approach offers several possibilities of establishing 3D/3D correspondences between a small subset of reconstructed points and their true values (anchors) in the desired coordinate system.

With a graphical interface the user can browse through the image sequence used for reconstruction by using a slider. For each frame the reprojected points are displayed. A reconstructed 3D point can be selected by clicking on any of its reprojected 2D points. The true 3D value can be either set by entering the coordinates into a text field or by clicking on a point on the suface of a CAD model (if available for this part of the scene).

The interface also lets the user select a point in the image, which was not automatically reconstructed by the SLAM. In this case, the image sequence is re-processed, corresponding 2D points in the other images are matched or tracked automatically, and the point is triangulated in the SLAM coordinate system. The user can also delete single measurements, complete feature tracks, or single frames.

5.5.2. Rigid Transformation Based on Manually Selected Anchor Points

Given the 3D-3D correspondences at M anchor points obtained by one of the methods described in the previous Sec. 5.5.1, it is now possible to calculate an optimal rigid transformation including a global rotation $\mathbf{R}$, translation $\mathbf{t}$ and scale s for the reconstructed feature map. This can formally be formulated as a constrained least-squares problem as follows:

[1] in practice we set $\bar{\phi}_{th} = 5°$

$$\underset{\mathbf{R}\in SO(3),\mathbf{t}\in\mathbb{R}^3,s\in\mathbb{R}_+}{\arg\min} \sum_{j=1}^{J} \|(s\mathbf{R}\mathbf{x}_j + \mathbf{t} - \mathbf{y}_j)\|_2^2 \tag{5.3}$$

where $\mathbf{x}_j$ and $\mathbf{y}_j$ denote the reconstructed and true 3D coordinates at the anchor points, respectively.

This problem is known as the *absolute orientation problem*, which can be solved using corresponding closed-form algorithms [AHB87, Hor87, Ume91]. Apparently, this is also a special case of the family of problems which we considered in detail in Chap. 2. Here, we are dealing with the 3D point-to-point registration case and - as we have shown in Sec. 2.5.1.1 - our own algorithms can also be used with the same accuracy than the above-mentioned standard approaches. The objective function is very similar to Eq. 2.3 in Sec. 2.3.1, if one chooses the identity matrix $\mathbf{I}^{3\times3}$ for the orthogonal complement matrix $\mathbf{N}_j$. However, the values of the scaling and the translation vector will be different and need to re-interpreted (inverse scaling and scaled translation).

Based on an obtained solution for the similarity transform, it is possible to transform the remaining points of the feature map and all extrinsic camera parameters of the sequence globally and equally in a rigid way as described in Sec 3.4. However, this transformation of the map will not account for non-rigid deformations due to systematic errors during reconstruction. In the next section, we will show how the information contained in the user defined anchor points can be exploited within the bundle adjustment algorithm in order to compensate for these errors.

5.5.3. Bundle Adjustment with Anchor-Point Constraints for Refinement

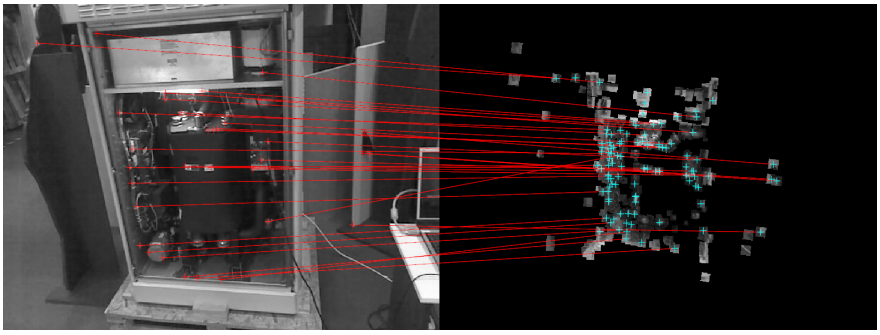
If the reconstruction is performed over long sequences, it is known that the accumulation of triangulation and pose estimation errors will result in a drift during reconstruction. As a consequence the feature map will obtain a deformed geometry. It may be particularly dominant at locations in the scene, where there are few features, so that the connectivity of the map is poor. These errors may even persist, when refining the reconstructed map with a global (unconstrained) bundle adjustment. When such a deformed map is used for tracking within an augmented reality application, this leads to systematic pose estimation errors and thus will lead to a constant misalignment of augmentations in certain parts of the scene.

Given the user defined true 3D-coordinates, it is possible to further make use of this information within the bundle adjustment from Chap. 3 in order to reduce these errors. In the case here, the anchor points represent completely known parameters. Therefore, this variant can be regarded as a special case of the general approach, which can also take into account partially defined correspondences. Since the variant here has a simpler structure, we would like to briefly comment on the algorithmic differences and explain the effect on the reconstruction result. Stated simply, we exclude the anchor points as optimization variables, but keep the true 3D coordinates as constraints that must be satisfied. As we will show later in our evaluations, even if only a small fraction of all points is known exactly, this already suffices to compensate for almost all of the drift, particularly when the points belong to distinct regions in the scene.

Compared with unconstrained bundle adjustment, the inclusion of anchor points can be achieved by the following differences on the augmented normal equations (cf. Eq. 3.3):

(a) The structure variables $\mathbf{b}_n$ corresponding to the anchor points $\mathbf{y}_j$ are excluded from the parameter vector $\mathbf{b}$, since it shall not be part of the optimization.

(b) Consequently, also the corresponding columns, $\partial f/\partial \mathbf{b}_n$ (partial derivatives with respect to structure), belonging to the anchor points need to be removed from the Jacobian $\mathbf{J}$. This is because the feasible set (or manifold) of completely known points is zero dimensional, so they are not allowed to move in any direction away from the given point.



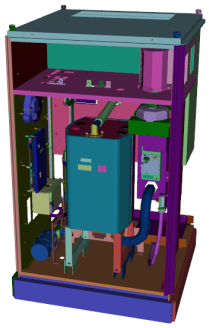


Figure 5.3.: Initialization of the camera pose with randomized trees classification. The camera image, the reconstructed feature map and the matches in between are illustrated. The 3D coordinates of key-points used for the classification are in the same coordinate system as the CAD model.

(c) The rows in $\mathbf{J}$, $\partial f / \partial \mathbf{a}_m$ (partial derivatives with respect to motion), are evaluated in each iteration using the coordinates of the anchor points, $\mathbf{y}_j$, instead of the reconstructed parameters, $\mathbf{b}_n$.

(d) Similarily, also the reprojection errors, $\mathbf{p}_k - f_k(\mathbf{a}_m, \mathbf{y}_j)$, are evaluated by using the anchor points (and not the reconstructed points).

Intuitively, the effect of these modifications on the reconstructed geometry is indirect by nature. Replacing some of the reconstructed points with their corresponding anchors (d) will initially increase the reprojection error of the objective function. Difference (c) models, how the *camera parameters* need to be adapted in order to reduce the error. Finally, a change in the camera parameters will lead to a different (relative) reconstruction result of all points. Differences (b) and (a) will ensure that the anchor points remain unchanged during the whole optimization. We emphasize another intuition for this procedure: as the percentage of anchor points among all points increases, the behaviour of the optimization gradually shifts from structure-and-motion to pure motion estimation. In the limit (when all points are defined to be anchor points), the procedure will then be equivalent to pure (nonlinear, Levenberg-Marquardt based) pose estimation, with vanishing interrelations between the poses for each frame, so that each pose will be solved independently of the others.

5.6. Feature Learning

Frame-to-frame point feature tracking algorithms, where a rough estimate of the point position is always needed to estimate the position of a feature in the current frame, can only be used if the previous camera pose has been estimated successfully. If the tracking fails and no assumption of the camera pose and feature positions can be made any more, wide baseline matching techniques have to be used. A popular and widely used method is the randomized-tree-based classification that Lepetit et al. [LLF05] first applied to feature recognition.

A tracking-by-detection approach using randomized trees is presented by Özuysal et al. [OLFF06]. The authors' system consists of an offline training and an online tracking phase. During the learning step image patches at certain 3D coordinates of a model at different camera viewpoints are collected and used to train the trees. In the tracking phase, these trees are used for the key-point classification and the pose estimation. The system is limited to objects that can be approximated by ellipsoids. In a different approach by Williams et al. [WKR07] relocalization during online SLAM is addressed. An online-learning algorithm to train a simplification of Ran-

domized Trees (Randomized Lists) during the tracking phase is proposed, whereby online capability is partially traded against robustness of the wide-baseline feature recognition.

Since for usable AR applications a relocalization procedure must be available right from the beginning, we collect patches of a set of key-points during the offline stage, and use these patches to train the randomized trees. After a 3D feature map has been reconstructed and transformed into the coordinate system of the target model, the image sequence is processed again while image patches of the reconstructed points are collected. To improve the detection rate, for all points of the reconstructed feature map the responses of the corner detector are accumulated first, and only feature points with a high repeatability are regarded as key-points for the randomized-tree classification. All those key-points are then taken to train the randomized trees.

In Fig. 5.3 the wide baseline matching results of a classification step are visualized. At camera positions, which are similar to those of the training sequence, it is then possible to estimate a camera pose without any pre-knowledge and initialize the frame-to-frame tracking module.

5.7. Feature Management

Frame-by-frame tracking relies on an approximate prediction of the pose of the current frame. With this estimate the 3D feature points can be projected onto the image plane in order to search them locally, which in our case is achieved again with the KLT-algorithm. However, even when the projected point lies within the viewing frustum, it may not be visible or recognizable. For example, the feature may be located on geometric corners such as screws, buttons or plugs with a non-planar surface. In this case, the viewpoint dependent intensity variations of the feature are not modelled well by a homographic or projective warp of a planar texture. Other typical reasons include viewpoint dependent specular reflections on the surface and, most importantly, self-occlusions of the object. The general consequence is, that one specific feature can only be tracked successfully from a limited range of viewing directions.

Requiring that one should be able to track the object from many different views inevitably leads to larger feature maps. The reason is that there must exist a minimum number of features for each possible viewpoint for the pose to be computed reliably and accurately. The downside of a large map is, however, that at other viewpoints this also results in a high number of features that are not recognizable, e.g. for which the KLT algorithm will not converge for the reasons mentioned above. It is therefore necessary to restrict the expensive, image-based computations only to those features, which are expected to be successfully tracked at a predicted camera position.

Viewpoint dependence on feature level is also considered by others. Najafi et al. [NGN06] present a combined statistical analysis of the appearance and shape of features from possible viewpoints. In an offine training phase they coarsely sample the viewing space at discrete camera positions and create cluster groups of viewpoints for every model feature according to similar feature descriptors. Thereby a map is created which gives information about the detection repeatability, accuracy, and visibility from different viewpoints for every feature. However, their approach is mainly targeted towards tracking-by-detection and due to the high computational effort not applicable to real-time frame-by-frame tracking. In [AWK*09] the feature map of a wide area location is split into potentially visible sets, and only those sets are used for the feature localization, which are visible at the current camera position. The assignment of individual features to the sets, however, is only realized manually.

Our strategy for selecting a meaningful subset of feature points is based on a statistical analysis of the feature visibility during the offline stage using the image sequence provided by the user. The viewpoint dependence is therefore obtained without requiring explicit user interaction.

The algorithmic analysis is based on the approach by Wuest et al. [WPS07], therefore we only briefly outline the main elements and intuitions. To every feature in the reconstructed map a tracking probability $p(\mathbf{y}_\mathcal{V})$ is assigned, which describes the ability to track a feature depending on the camera position $\mathbf{y}_\mathcal{V}$ in the coordinate system $\mathcal{V}$ of the virtual model. This probability is estimated by observing the tracking successes and the tracking failures during the learning stage. These are approximated with a Gaussian mixture model (GMM) with a finite number of Gaussian distributions. Hierarchical merge operations are used to create a GMM out of all observations. More details can be found in the original publication [WPS07]. In our experiments we limited the number of Gaussians to 8.

In the online tracking phase, the tracking probability $p(\mathbf{y}_\mathcal{V})$ of all feature points is computed at the predicted current camera position $\mathbf{y}_\mathcal{V} = -\mathbf{R}_{\mathcal{V}\to\mathcal{C}}^\mathsf{T}\mathbf{t}_{\mathcal{V}\to\mathcal{C}}$. In the simplest case, one can use the pose of the previous frame as approximation. Incorporating the inertial measurements within an sensor fusion approach - as described in the next Sec. 5.8 - yields more robust and accurate predictions, which makes the combination of both techniques particularly reasonable. Next, a container of all potential features is sorted in descending order by their individual tracking probability. Only a certain amount of features with highest tracking probability is used for the further image-based tracking. Therefore unnecessary computations can be saved for features that are not from the current view. As a result, the computation time is reduced, whereas the robustness of the camera pose estimation remains mostly unaffected.

5.8. Sensor Fusion

To improve the robustness of an inside-out camera-based tracking system, inertial sensors and gyroscopes have often been used. Inertial sensors are capable of measuring the 3D orientation as well as 3D acceleration. The camera attitude, the angular velocity, and the acceleration as a second derivative of the position can be estimated. However, a purely inertial tracking unit is not able to determine the absolute position in indoor scenarios. Therefore several methods have been developed to combine inertial sensors with camera-based tracking approaches.

The InterSense VIS-Tracker [FN03] is an early commercially available system, which is based on artificial markers. Jiang et al. [JNY04] presented a system which fuses gyroscope measurements with a vision-based line tracking method. Reitmayr [RD06] also uses line features for the computer vision part of a hybrid tracking system, but instead of using a coarse line model, he uses a textured 3D model and extracts line features out of an image, which is rendered with a predicted camera pose. A similar analysis-by-synthesis technique is presented by Bleser et al. [BS09], where the camera-based tracking also relies on a textured model, but instead of line features, point-based features are used. However, creating a photo-realistic textured model still remains a difficult problem. Commercial tools exist, but do not always produce good results, especially in large-scale scenarios. If a scene does not consist of many planar, well-textured surfaces, it is almost impossible to create a 3D model from reference images, which can be rendered in a photo-realistic way from many different viewpoints.

Our approach for fusing inertial sensor data with a natural feature tracking method is mostly based on [BS09], but instead of using a textured model for the camera-based tracking, we rely on the sparse reconstructed 3D feature map, whose acquisition was described in the previous sections.

The camera-IMU system consists of a Point-Grey monochrome camera with a wide field-of-view lense and an XSense MT9-C inertial measurement unit. Both devices are synchronized in hardware, the IMU provides measurements at 100Hz and triggers the camera at 25Hz.

The camera pose is estimated with an extended Kalman filter (EKF), where both inertial measurements and 2D/3D-correspondences from the vision-based tracking are fused. The filter state consists of the camera position,

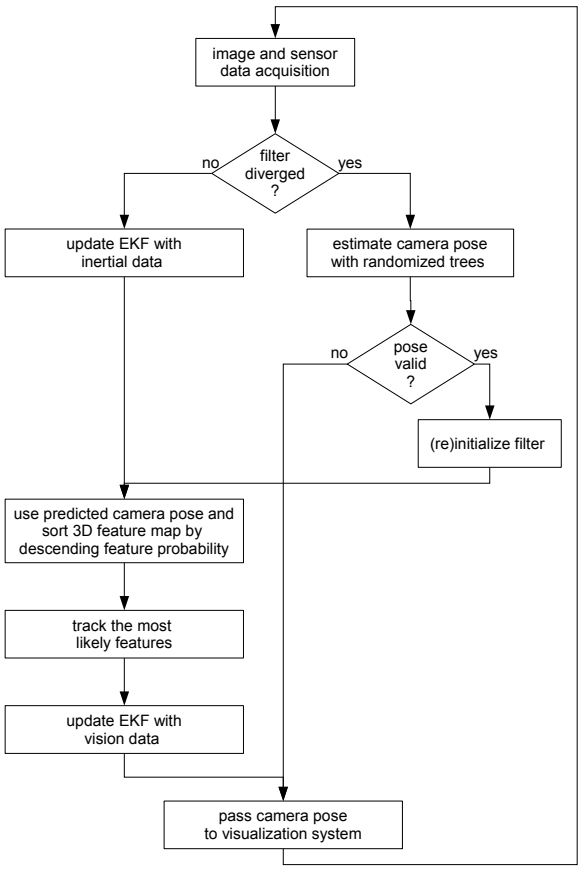


Figure 5.4.: System workflow of the hybrid tracking approach using camera-based tracking with a reconstructed feature map and an inertial sensor.

the velocity, the orientation, the angular velocity, the gyroscope biases, and the accelerometer biases. Details can be found in the original publication by Bleser et al. [BS09].

In Fig. 5.4 the system workflow of the hybrid tracking approach is outlined. In the acquisition step the current camera image and all IMU measurements that have not been processed are captured. If a divergence of the EKF is detected, the camera pose is initialized with randomized trees feature point classification. With a successful camera pose the EKF is initialized. If no divergence is detected, the EKF is updated with all available sensor measurements. After the filter update, a camera pose prediction is extracted out of the current filter state and the tracking probability of all points in the 3D feature map is calculated according to the camera pose prediction. A limited number of features with the highest tracking probability is projected into the current camera image and a local search for those feature points is performed. The resulting 2D/3D correspondences are used to update the EKF. The camera pose, which is now represented by the filter state, is then passed to the visualization system together with the current camera image.

To avoid false measurements, which arise from incorrectly tracked feature points, a simple χ^2-test is used, and only 2D/3D correspondences which pass the test are used to update the filter state. The divergence detection is based on a combination of monitoring the average re-projection error of the feature points and the covariance of the filter state vector.

It is important that a correct down vector in the coordinate system of the feature map is given, in order to correctly compensate the gravitation measured by the accelerometer. Furthermore, the relations between the scale of the feature map and the sensor coordinate system needs to be modelled, since the accelerometer delivers absolute acceleration measurements. In our system, the sensor coordinate system is modelled in meters.

5.9. Evaluation

5.9.1. Measuring Feature Map Accuracy with a Dedicated Setup

In order to illustrate the merits of the constrained bundle adjustment, we built a real model consisting of two highly textured plane faces perpendicular to each other. We reconstructed its feature map as described in Sec. 5.4. For creating a significant drift during reconstruction, we moved the camera at a very close distance along the model (approximately three centimeters), so that each image captured only a very small fraction of the scene, leading to a sparse connectivity among the features. The total map consisted of 1909 features, reconstructed in a sequence with 240 frames. The reconstructed map was transformed into a predefined coordinate system with a rigid transformation as described in Sec. 5.5.2. For this, we used a total number of eleven anchor points, five at the very left side of the model, another five at the right and one in the middle near the corner.

As can be seen in Fig. 5.5, the rigid transformation will already achieve a good alignment with the ground truth planes. However, it will not be able to account for the geometrical deformation caused by the accumulation of pose and triangulation errors during the reconstruction process (see 5.5b). A subsequent ordinary (i.e. unconstrained) bundle adjustment (using the library of [LA09] for comparison) will only partially compensate the deformation (c). In particular, if one were to take different (or even more) points for the rigid transformation, it would not lead to a better alignment, because a further improvement is impeded by the deformation of the map. This can be observed particularly well if only a subset of anchor points is taken to calculate the transformation. In Fig. 5.5d we only chose the five anchor points on the right and the one in the middle, which results in a better alignment of the right side, but also a stronger misalignment of the left part of map. Likewise, choosing points on the left leads to deviations on the right (e). Finally, 5.5f depicts the reconstructed feature map with the constrained bundle adjustment (as described in Sec. 5.5.3) using the same points that were chosen for rigid transformation. As can be seen, almost all of the drift is compensated, even though only a very small amount (eleven points) of the map is known exactly.

Table 5.1 summarizes a quantitative comparison corresponding to the situations depicted in Fig. 5.5. The errors in 3D represent the perpendicular distance of the features with respect to the planes they correspond to. Note that the average reprojection error for the constrained bundle adjustment increases slightly, since the reprojection error is not the only criterion for minimization, anymore. Note also, that the rigid transformation does not affect reprojection error (see table 5.1 (c-e)), as long as the extrinsic camera parameters are transformed equivalently.

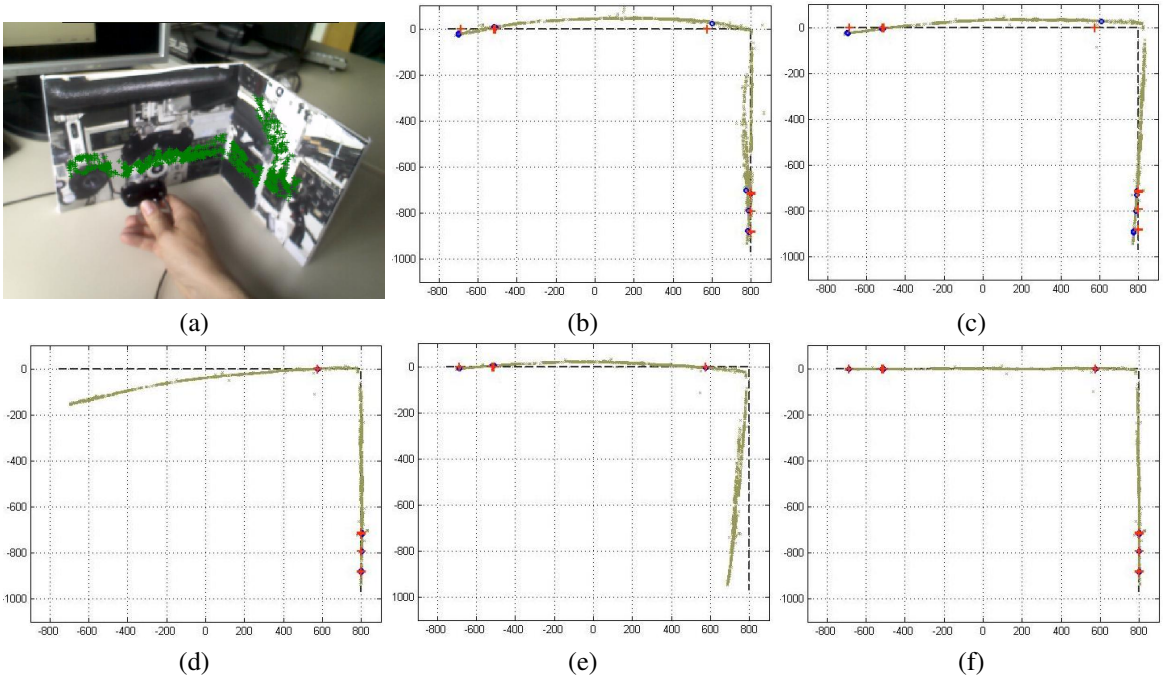


Figure 5.5.: Evaluation of geometrical transformation and drift compensation of the feature map. Experimental setup with augmented features (a). Rigidly transformed feature map before bundle adjustment (viewed from top)(b), after ordinary bundle adjustment using all anchor points for transformation (c) and using only anchor points on the right (d) or on the left side (e). The result after drift compensation with our constrained bundle adjustment method (Sec. 5.5.3)(f). Red crosses are anchor points and blue circles represent the corresponding reconstructed points. The dashed line indicates the ground truth planes.

Table 5.1.: Numerical results corresponding to Fig. 5.5. Reprojection errors are in pixels. Errors in 3D are measured perpendicular to the ground truth planes.

	avg. err. (in pixels)	**avg. 3D err. left plane**	**avg. 3D err. right plane**	**avg. 3D err. total**
(b)	0.6273	35.7384	15.2681	27.1662
(c)	0.2087	28.1170	12.9432	21.7627
(d)	0.2087	50.0717	6.0996	31.6578
(e)	0.2087	16.7571	72.2835	40.0095
(f)	0.2205	2.2425	2.7786	2.4670

5.9.2. Alignment Quality of Augmentations

We further evaluated qualitatively the effects of inaccurate (deformed) feature maps on the alignment of augmentations. To this end we reconstructed two sides of a large building with a 691 frames sequence. The KLT patches of the map consisting of 2562 points are depicted in 5.6a. Nineteen 3D-3D correspondences (anchor

points) equally distributed around the building were obtained using the CAD model of Fig. 5.6b. After performing a bundle adjustment with both, an unconstrained method (using the SBA library, again) and our constrained scheme of Sec. 5.5.3 [2], both maps were used for online tracking and augmentation.

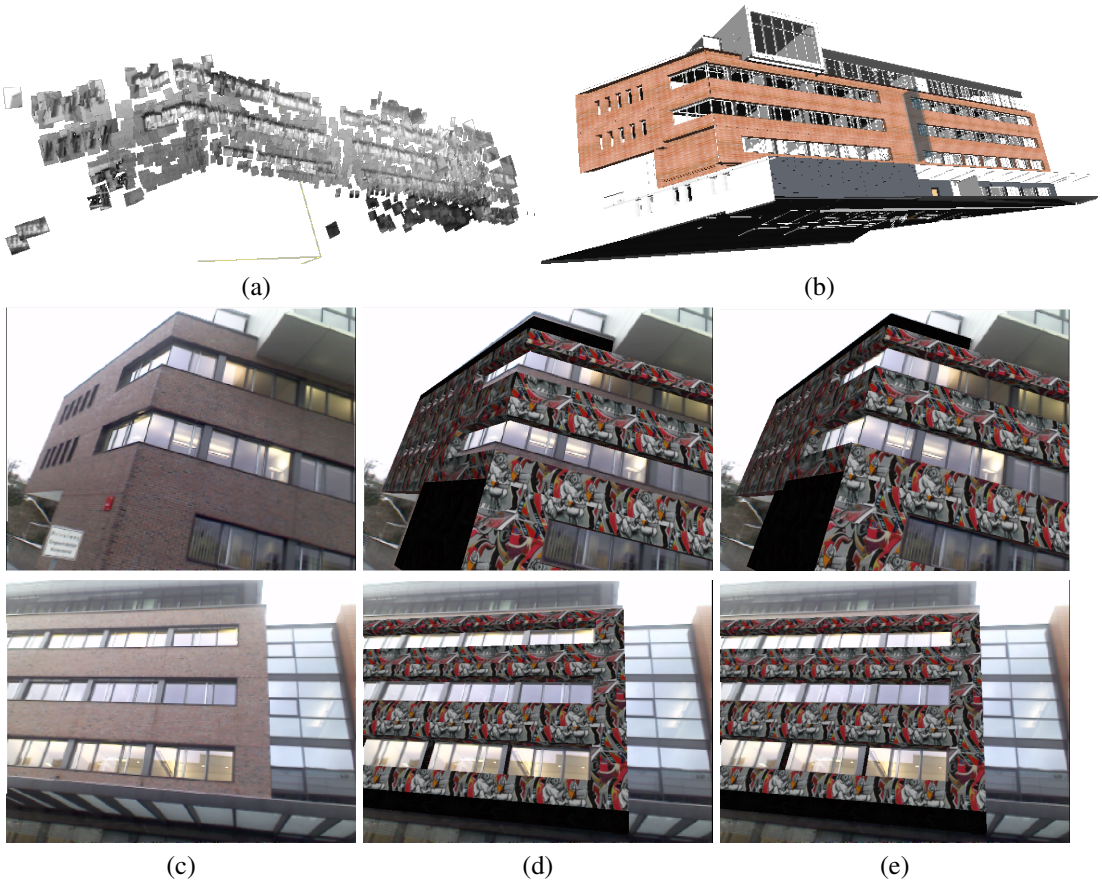


Figure 5.6.: Effect of deformed feature map on the alignment of augmentations during tracking. (a) Feature map after initial map construction. (b) CAD model of the tracked building. (c) Images of a sequence used for evaluation. Augmented images while tracking using the map obtained with unconstrained (d) and constrained bundle adjustment (e). Note the offset at the window frames and the edges of the building for the unconstrained case (d) especially in the upper picture.

When using the first map for tracking (Fig. 5.6d) the alignment of the augmentation and the real object was acceptable in certain parts of the scene (lower image) but at other locations we observed a constant (systematic) misalignment (upper image). By contrast, when using the map which was optimized with constraints, the augmentation stayed right in place during the complete sequence (Fig. 5.5e).

[2] In the first case the rigid transformation (absolute orientation, Sec. 5.5.2) was done after bundle adjustment of the map

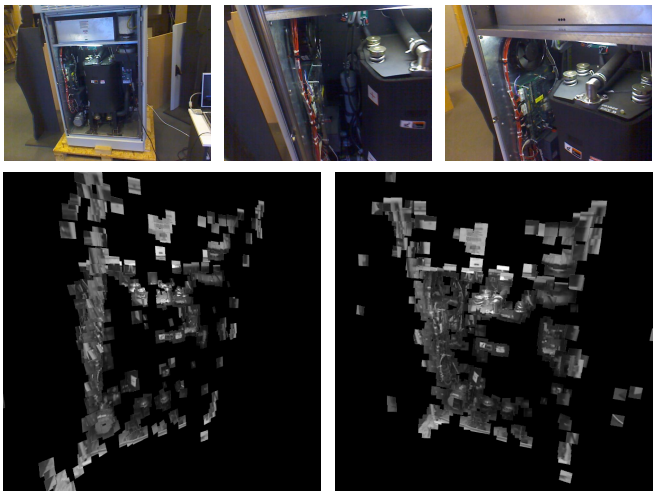
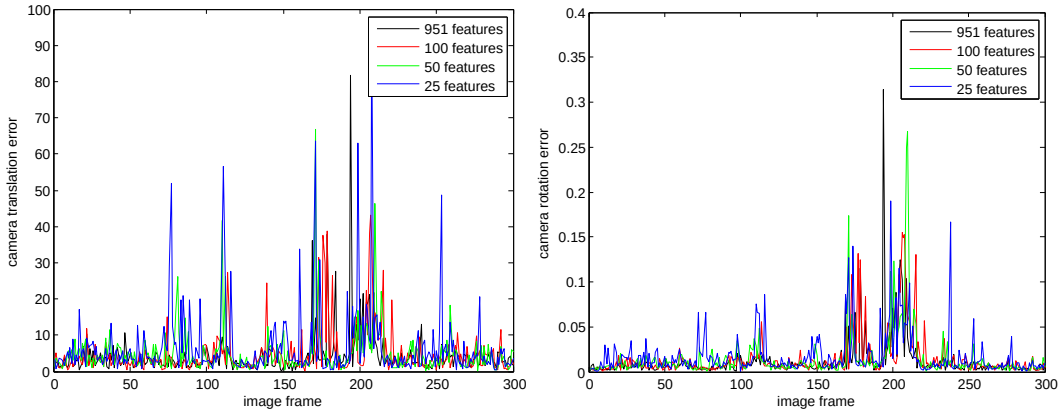


Figure 5.7.: Some frames of an image sequence and the resulting 3D feature map from different viewpoints are shown.



(a) (b)

Figure 5.8.: Error of the camera translation in millimeters (a) and the rotation in radians (b) with different numbers of active features.

Table 5.2.: Number of tracked features and the corresponding average error of the camera pose.

Number of features	average translation error [mm]	average rotation error [rad]	time [msec]
951	4.1882	0.0118	510.4
100	4.8370	0.0140	41.27
50	4.9458	0.0139	29.61
25	6.6953	0.0175	22.19

(a) (b)

Figure 5.9.: Tracking an industrial object with the sensor fusion approach. In (a) the tracked features are colored green, feature of the model not selected for the tracking step are orange. In (b) the same frame is visualized together with an augmentation.

5.9.3. Evaluating Feature Management: Tracking Accuracy versus Speed

For the experimental evaluation of the feature map acquisition and the tracking, an image sequence with a large industrial fuel cell is taken. The object is challenging for vision based tracking, as it potentially gives rise to self-occlusions and consists of many parts with specular surfaces. We demonstrate that feature management under such circumstances may increase tracking speed significantly without considerable loss in accuracy, since only the most relevant features are processed in each frame. In Fig. 5.7 some frames and the resulting 3D feature map are visualized after reconstruction and spatial registration with the model.

In absence of ground-truth data, we compared the online tracking including feature management to the pose estimates of the bundle-adjusted sequence by evaluating the absolute differences in translation (millimeters) and rotation (radians).[3] In each run, we varied the maximum number of features to be processed by the online tracking. In Fig. 5.8 these errors are plotted. It can be seen that the camera pose can be estimated successfully throughout the whole sequence, even if at every tracking step only the 25 most likely features are selected and tracked. Table 5.2 summarizes the corresponding average error, which is slightly increasing when the number of features is reduced, but it is still in a very acceptable range. Note, that a fair amount of computation time can be saved with only very slight losses of accuracy.

5.9.4. Hybrid Tracking

The sensor fusion approach described in Sec. 5.8 is also evaluated with the same industrial object. A sequence of image and sensor data is captured with the combined camera-IMU-system, a feature map is reconstructed with the images, and this feature map is then used together with the feature management system for the hybrid camera pose estimation. The feature management system only takes a certain amount of features into account. Since the sensor measurements already allow a fairly accurate prediction of the camera pose, image measurements are only needed to prevent the filter from drifting. Thus, even much less feature points are needed. In our experiments it

[3] For comparison: the height of the fuel cell depicted in Fig. 5.3 and Fig. 5.7 is 1550 millimeters

was possible to reduce the maximum number of tracked features to five. The first camera pose and the relocation after a divergence of the EKF is estimated with randomized trees.

The tracking system is capable of estimating the camera pose at 25Hz throughout the whole test sequence, even during very fast camera movements. In Fig. 5.9 a snapshot of the evaluated sequence can be seen. Compared to the pure vision-based tracking approach, the sensor fusion method is able to produce a accurate camera pose estimate, even with much less analyzed feature points. Fast camera movements, especially camera rotations, can be handled very stably while at the same time reducing the workload for image processing.

5.10. Evaluation on Public Benchmarks

With the approach presented in this chapter, we have participated at various *tracking competitions*. The aim of these events was to provide a common platform for comparison and to expose and document the technical progress of tracking solutions in the field of AR with regard to their limitations and their usability in real scenarios. Usually the contests addressed the global and absolute localization capability of the systems inside a static scene in terms of accuracy and robustness.

To this end, the typical task for the competitors was to identify certain elements in the scene (points or subparts of an object) by solely using an augmented reality visualization. Before the start of the contest run, the contestants received an individualized list of coordinates that belonged to some of these elements. The tracking system was supposed to localize itself in the scene. Then, for each point in the list, the system had to navigate the contestant to the corresponding object and to highlight it. For example, if the point was outside the viewing area, arrows could be used to indicate the general direction. For highlighting the object inside the view, 2D or 3D cross-hairs were suitable. Based on this visualization the contestant had to indicate the putative object or element to the organizers (point picking) or had to remove parts of the scene and place them at a different location (scene manipulation, e.g. ISMAR 2012 or the Lego example at ISMAR 2011). In some cases the competitors were asked to directly mark points on a blank surface or wall (without a reference to an identifiable object) by using a pen or by placing point markers, from which the deviations to the (unobservable) ground-truth points were measured in millimeters afterwards by the organizers (ISMAR 2009 and 2010 and Scenario 3 of the VW Tracking Challenge 2014). In the point-picking scenarios of earlier contests, one score point for each correctly identified object was given. From 2011 on, the evaluation also included negative score points for incorrect picks and a weighting factor per object, which reflected the difficulty of the individual task. For the AVILUS Tracking Contest in 2010 various practical scenarios were developed with partners from industry of the German funded AVILUS project (VW, Kuka, HDW, EADS). It was aimed at a qualitative comparison, which also took into account other criteria such as the tracking speed (frame rate) or the amount of effort during preparation. The results are summarized in a report [SK11].

Typically, before the actual contest run, some time for preparation was given. During this preparative phase the teams could take snapshots or videos of the environment, reconstruct the scene and test their approach in a dry-run. Furthermore, it was necessary to register the tracking system to the coordinate frame defined by the organizers. To this end, some reference marks were distributed across the environment, whose coordinates were provided. We used these reference points for spatial registration as described in this chapter (Sec. 5.5) including constrained BA. During the contest run, it was not allowed to use the reference points, neither for tracking nor for visualization (e.g. in order to assess the tracking accuracy by means of the offsets between projected and real points), so they were usually occluded after the preparative phase. The organizers have always taken great pains to accurately measure the target and reference points in the scene in advance, using special gauges such as total stations or optical coordinate measuring machines.

Table 5.3.: Major contest events at which our approach has been used and compared with the solutions of other competitors.

event name	year	location	description	score reached / score max	numb. of teams	place
ISMAR Tracking Competition	2009	Orlando	point picking, mark points on a transparent board	6/16	4	1st
AVLUS Tracking Contest	2010	Ottobrunn	various practical scenarios	qualitative comparison	3	report [SK11]
ISMAR Tracking Competition	2010	Seoul	point picking	no official announcement[a]	5	1st (tied)
ISMAR Tracking Contest	2011	Basel	point picking, scene manipulation, weighted score, negative points for false picks	8/26	5	2nd[b]
ISMAR Tracking Competition	2012	Atlanta	object picking and placing	18/18	3	1st
VW Tracking Challenge at ISMAR	2014	Munich			6	
Scenario 1			"Tracking of rotating vehicle", model-based, point picking	30/39		2nd
Scenario 2			"Tracking and learning on different vehicles", point picking	33/41		1st
Scenario 3			Tracking with high accuracy, point marker placement	placement error 2.16mm		1st (tied)[c]
Scenario 4			Tracking inside unknown area, point picking, SLAM tracking	0/12		(no winner)

[a] In 2010 (Seoul), inaccuracies in measuring the competition site by the organizers have led to one of the participants disputing the results, so that officially no winner was announced.

[b] At ISMAR 2011 (Basel) we participated with a remote tracking solution. A mobile phone was used to capture the images with the camera and to display the augmented image. The actual computations were performed by the server (laptop), notably the tracking and the augmentation. Transmission of the images back and forth was done via Wifi. This became problematic as soon as the room filled with many spectators, whose own cell phones were spamming the transmission link. As a result of the slow transmission rate we could not accomplish all tasks in time.

[c] In Scenario 3 in 2014 (Munich) our approach had the highest accuracy. However, an own carelessness forced us to redo parts of the preparation during the contest run. Since the time needed to fully accomplish the tasks was also a scoring criterion according to the official rules, the jury decided to split the first place among the two participants with the highest accuracy.

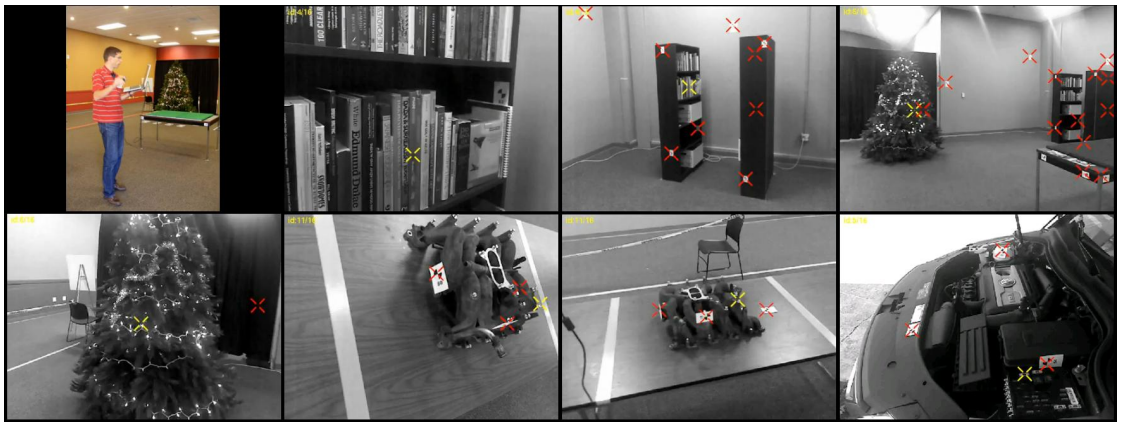


Figure 5.10.: ISMAR Tracking Competition 2009 at Orlando. The yellow crosses correctly indicate the location of some of the objects that needed to be identified with AR: individual books in a bookshelf, Christmas balls on a blinking tree, screws of an industrial object, parts inside an engine compartment of a car. Red crosses mark the position of reference points provided by the organizer for registration (not used during the competition run). Images captured during preparation phase are shown. The first image depicts the used mobile configuration (laptop and webcam) of our system.

Figure 5.11.: ISMAR Tracking Competition 2011 at Basel. From top to bottom: (i) our remote, server-based tracking with visualization on a mobile phone, some general impressions of the contest and the reconstructed and globally aligned feature map, (ii) augmented view for the scenarios: table with candies, mirror with transparent marbles, tunnel with grid of target points inside, transparent key board with grid of rings (iii) lego towers to be built at the correct position and height, target points inside and outside a vehicle.

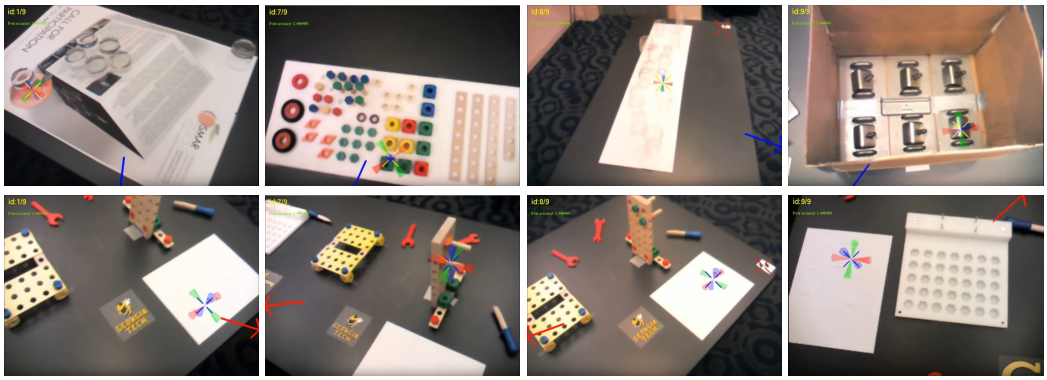


Figure 5.12.: ISMAR Tracking Competition 2012 at Atlanta. Several objects distributed across the competition site had to be picked (top row) and placed at designated spots on a table in the center of the room (bottom row). The red and blue arrows indicate the directions to the source and target locations, respectively, whenever they were outside the view. The type of objects included (from left to right): glasses with target locations on a paper sheet (possible destinations had been marked by the organizer with pencil outlines), toy screws and other parts that had to be mounted on a construction, glasses in a barely textured area, and industrial parts in a bin.

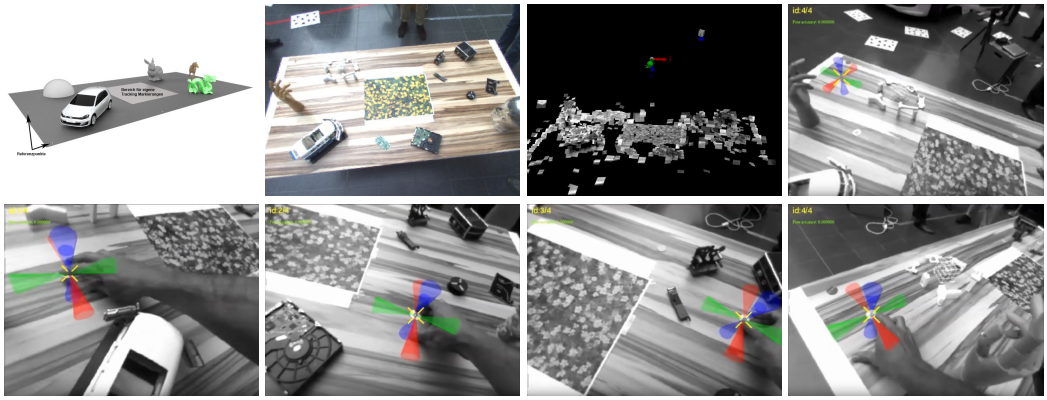


Figure 5.13.: Volkswagen Tracking Challenge 2014 at Munich, third of a total of four scenarios (accuracy contest). The task was to place small point markers at target locations as accurately as possible. After the contest run the organizers measured the deviations to the ground-truth positions with an optical coordinate measuring machine. In the center, within a demarcated area, it was allowed to place own markers or even 3D objects to support the tracking: we used a highly and irregularly textured planar pattern. Top row: general task description, the real setup of the task with our pattern, the reconstructed feature map, and a live view during the run with the highlighted target position. Bottom row: live recording of the run while placing the four markers.

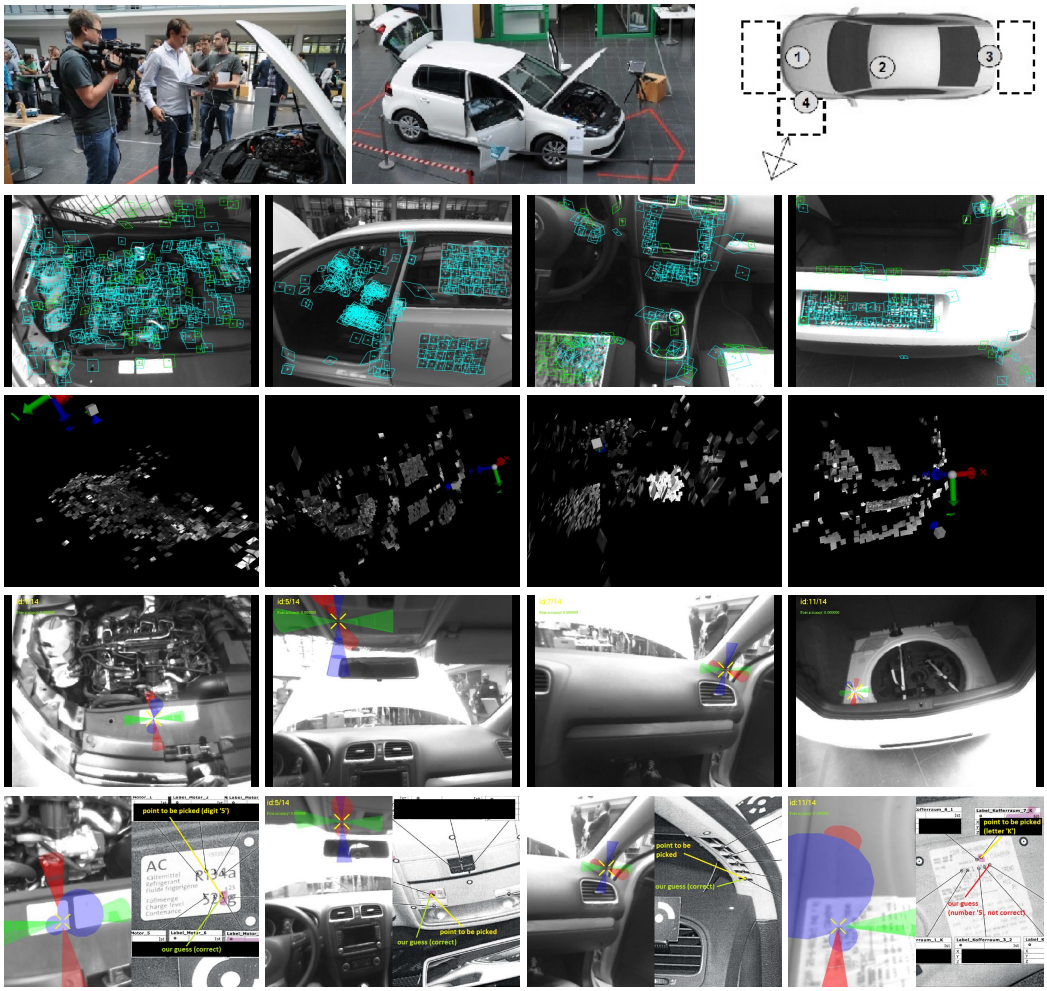


Figure 5.14.: Volkswagen Tracking Challenge 2014 at Munich, second of a total of four scenarios. From top to bottom: (i) general impressions and relevant tracking areas, (ii) images and feature tracks during preparative phase, (iii) reconstructed feature maps, (iv) live view of contest run, and (v) comparison of correct scene points and the points that have been selected during the run.

The difficulty across different tasks has been varied by the following means:

- **sparsely textured scenes or objects**, giving only a few features for tracking,
- **transparent objects or parts**, such as glasses (ISMAR 2012) or tiny and hardly visible elements on transparent boards (ISMAR 2011),
- **variations in lighting conditions**, such as using a projector to illuminate the scene with random patterns (VW Tracking Challenge 2014), or blinking fairy lights on a tree (ISMAR 2009),
- highly **reflective and specular objects**, such as a mirror with a grid of transparent marbles (ISMAR 2011),

- **substantial scene changes** between preparation and contest run, e.g. the removal of the cover to the spare wheel in the trunk of a car (VW Tracking Challenge 2014),
- scene parts with **repetitive patterns**,
- **small separation distance** between target objects or points, such as individual letters and digits on a receipt (VW Tracking Challenge 2014), or
- **large distance to the reference points** for spatial registration to the target coordinate system (VW Tracking Challenge 2014).

Table 5.3 provides an overview of the most important tracking contests in which the approach of this chapter has been used and summarizes the achieved results.[4][5] We have always scored one of the top two places in all competitions, in the majority we even emerged as the winner.

As it turned out, a sophisticated method for spatial registration of the tracking system to the target coordinate frame was a crucial element for accurate absolute localization. Its value was particularly evident whenever high accuracy was required, for example when one of several points had to be identified, which were only a few millimeters apart (such as individual digits on a receipt). The procedure for spatial registration was also the part in which our system differed most from that of the other competitors.

5.11. Conclusion

In this chapter, we have presented an approach for an easy and accurate set up of a tracking system for AR applications during a preparative phase.

A central aspect is the authoring or the spatial registration of the virtual content with regard to the reconstructed real word model of the SLAM. Existing methods can be divided into two categories, which we labeled *SLAM-centric* and *virtual-model centric* approaches. In user-centered SLAM-centric approaches, the virtual content is interactively inserted into the real environment using the SLAM coordinate system as reference by positioning, rotating, and scaling virtual objects until the results looks plausible. These approaches have drawbacks regarding the accuracy as well as the re-usability of the created content. By contrast, model-centric approaches attempt to directly register the SLAM model or the estimated tracking pose to the target coordinate system, which essentially allows to decouple the authoring process from the SLAM. However, an automated registration is currently only possible if additional sensor or data sources are available and thus they are bound to these extra conditions limiting its applicability. In particular, CAD-model-based tracking methods need user support at runtime for initialization, apart from requiring a model that matches reality well enough.

This motivated us to develop a system which combines the respective advantages of both approaches. With our system it is possible to register a reconstructed SLAM model to the virtual domain, but we let a user have control over this process via an editing interface. With a human operator in the loop, it is possible to better capture implicit knowledge or use-case specific expertise, and therefore to cover a large range of possible application areas including indoor and outdoor scenarios. At the same time, we exploit the given information to best extend possible to obtain highly accurate registration results. The provided correspondences between the real and virtual domain are not only used for rigid registration. They are also included into the constrained BA of Chap. 3, with which it is possible to further correct deformations of the reconstructed scene model in order to achieve a better overlay of augmentations during runtime.

[4] Not included in Table 5.3 are the VW Tracking Challenges in 2013 and 2015 in Wolfsburg and also the first ISMAR Tracking Competition in 2008 in Cambridge, because there we have used tracking approaches different to those reported in this work (In 2013 and 2015 we predominantly used CAD-model-based tracking).

[5] Some of the tracking competition results are still available online, see [TU 11] for 2011 and [Vol14] for 2014.

Based on the input sequence, the preparative phase is further used to capture additional information and to train the system for robust high-performance tracking in the online phase. In particular, we integrated the randomized-trees machine learning method for feature classification and fast tracking (re-)initialization. Moreover, we also train a feature management method which analyses the visibility of features as a function of the viewpoint, in order to allocate resource-expensive image processing tasks during runtime only to those which have a high probability of being successfully tracked. We argue that the proposed workflow and the algorithms together with their complementary benefits are well-suited for mobile devices with scarce computational resources, as the main workload is transferred into the preparative stage.

We have demonstrated the broad applicability, robustness, and high accuracy of our approach in our evaluations. Furthermore, we competed very successfully against other systems from academia and industry at various tracking competitions at ISMAR conferences and other events.

6. Geometric Optical See-Through Display Calibration

6.1. Introduction

In a larger context, this thesis addresses geometric registration for AR, i.e. the precise spatial embedding of virtual 3D data within the surrounding, real world of the user. Misplaced virtual objects may harm the acceptance of AR, as they would rather confuse and distract the user, so that the benefits of AR might be reduced or even reversed. An important aspect concerns the visualization component of an AR system. Video see-through (VST) displays, where the virtual content is overlaid onto a live camera image, represent the most widespread displaying technique for AR, because their technical realization can be accomplished very easily. However, they are ultimately undesirable due to ergonomic limitations or their lack of immersiveness. In contrast, optical see-through (OST) displays aim at presenting the virtual content directly in the user's field-of-view while allowing for an unobstructed perception of the surrounding reality.

Head-up displays (HUD), which are increasingly being installed in modern, high-end automobiles, represent one possible variant of OST-displays. Here, the windshield of the car is used as a transparent reflector, bringing important information directly into the driver's field-of-view. This increases safety and comfort as the driver does not have to redirect his attention away from the traffic situation ahead. Current devices are only limited to the presentation of textual or symbolic information, such as the current speed or navigation icons. The next logical step is to turn these devices to augmented reality displays so that the virtual 3D content appears spatially registered to the environment of the vehicle and the driver.

Augmented reality HUDs would enable a range of applications where 3D information is particularly useful [PDT*09, GTFC12]. One example is the displaying of navigation signs spatially registered to the environment (such as arrows which appear to lie directly on the street). Compared to symbolic or aural information presentation, this would further help to disambiguate multiple turning possibilities at complex road crossings. Other applications include lane guidance, indication of potential dangers [TSLB05, TK06] and the visualization of the minimal breaking distances [TLK07]. Originally, the idea of using HUDs as augmented reality devices was introduced and explored for avionic applications [FALS92, BH93, BB02, HJEW02]. Here, pilots are assisted in navigating the plane safely in the air or on the ground by visualizing flight corridors or by highlighting taxiways when the visibility is low.

HUDs (or OST displays in general) pose additional challenges to the perspectively correct rendering of virtual 3D content, as the virtual objects need to be presented from the user's perspective rather than the camera's perspective (as in the VST-displays). The viewing position of the user with respect to the display is subject to variations and not known in advance. This requires a parameterizable model for the interplay between the user's perspective and the display unit. Although some geometric information about the device may exist as specifications (such as CAD models, simulation data for the virtual plane, and the position of an average observer), it does not necessarily reflect the true situation after assembly and final use. For example, a small tilt of the projector unit may lead to gross displacements of the virtual image and consequently also to a misalignment of the displayed objects. In general, these errors may not be negligible for users expecting immersive AR.

In this chapter, we propose a camera-based calibration process to retrieve such a parametric display model without requiring any prior information on the device. '*Camera-based*' means that we essentially replace the observer with a camera in order to accurately and quantitatively determine the relationship between displayed virtual elements and the real world. These correspondences are captured from a variety of viewpoints, thus ensuring the parameterability of the model, which in turn allows the adaptation of the rendering to the user's perspective at runtime (head or eye-coupled display). Our proposed model is not limited to a perspective approximation only, but also includes view-dependent distortions caused by the optics of the device or the windshield of the vehicle.

To realize such a camera-based calibration method, it is necessary to register the camera path to the reference coordinate system (vehicle or HMD coordinate system). For this purpose, the techniques described in the previous chapters are used. As all information is taken from the images, including an accurate pose in reference coordinates, no external tracking equipment is needed.

The proposed display model and the calibration procedure were originally developed for an automotive HUD. Therefore the following exposition primarily addresses this particular case. But we also show how our method can be transferred to the calibration of HMDs. Compared to previous works on HMD calibration, our method allows an easier adaptation to different users, particularly with regard to the location of the eye centers relative to the glasses, including the individual eye separation distance.

6.2. Problem Statement and Related Work

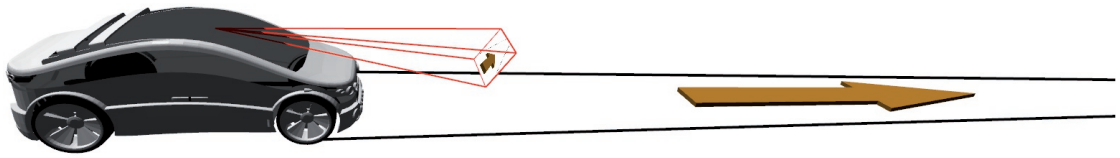


Figure 6.1.: Geometric model of the HUD. The magnifying optics of the HUD and the windshield of the car will together create a virtual image that appears at a certain distance from the driver.

As a use-case for our calibration method we use a real, monoscopic HUD installed in a prototype vehicle. The reference design consists of a planar displaying device[1] below the windshield. Light rays coming from this intermediate plane first pass through an optical magnifying unit and are then reflected by the windshield to finally reach the driver's eyes. The viewer will perceive the generated image overlaid on the real scene in front.

The underlying HUD model that we assume can be explained as follows (see Fig. 6.1). A monoscopic HUD creates one single image for both eyes. In order to mitigate disturbing effects due to the binocular disparity of the human vision, the optics are designed so that the virtual image appears at a relatively far distance from the driver, where these effects slowly begin to vanish. This virtual image plane does not necessarily have to be orthogonal to the viewing direction, rather it may also be oriented in an oblique angle. What needs to be realized is that in a first approximation this situation can be represented by a perspective pinhole model with the head of the viewer (or the midpoint between both eyes) being the center of projection.

[1] In our case a projector is used which produces an image on an intermediate image plane. Other devices such as OLED or TFT screens could also be used.

In addition, the generated image is subject to distortions. This is because for every possible viewing position the light rays will be reflected at different locations on the windshield. Due to its irregular surface geometry this will result in additional view-dependent, non-projective distortions of the image. Given this model we are concerned with three questions in this section:

- How to estimate the *projective properties* of the HUD, which we describe by the spatial geometry of the virtual plane, i.e. its position, rotation and scaling,
- how to model and estimate *non-projective distortions* induced by the optics and the windshield, and
- how to *render perspectively correct* on the oblique virtual plane given its spatial geometry, the estimated distortion model and the head position of the viewer?

To our knowledge, no calibration method for HUDs has been proposed in academic literature so far. However, optical see-through head-mounted displays (OSTHMD) share many similarities with HUDs, as they also use a semi-transparent reflector as combiner to create a virtual image in front of the eye. There exist many approaches for the estimation of the *projective properties* [TN00, GTKN01, MGT*01, KBB*12, OZTX04, GHA03, GFG08]. Most of these methods rely on user interaction, where several virtual points on the display have to be aligned with real markers in the scene [TN00, GTKN01, MGT*01, KBB*12]. Gao et al. [GHA03] and Gilson et al. [GFG08] create a setup that essentially replaces the human eye with a camera, so they obtain better means for the quantitative evaluation of the calibration accuracy. Common to all these approaches is that the projective characteristics of the display setup are represented by a single projection matrix (extrinsic and intrinsic parameters). The disadvantage is that the calibration remains valid only as long as the eye (center of projection for HMDs) does not change its location with respect to the display[2]. A recalibration is always necessary for each user or when the HMD is accidentally moved. Our approach to the projective modeling of the HUD is most similar to Owen et al. [OZTX04]. Instead of a single projection matrix we estimate the spatial geometry of the virtual plane, i.e. its position, orientation and scaling in vehicle coordinates. This allows us to synthesize any projection matrix from any head position of the observer for dynamic viewpoint-dependent rendering on the HUD.

The traditional approaches to calibrating *non-projective distortions* of projector systems or cameras [Tsa87, HS97b, Zha00, DF01] are based on the classical model by Brown [Bro71], where the distortions are modeled as polynomial functions of normalized image coordinates. Other parameterizations exist, such as logarithmic models [BL95] (see Sturm et al. [SRT*11] for more alternatives). Unfortunately these distortion models are not applicable to HUDs, as they make certain assumptions on the structure and the symmetry of the camera or projector optics. For example, in the classical model only radial and tangential distortions are represented (see Sec. 3.8 on page 63). Bhasker et al. [BM07, BJM07] propose a more generic modeling based on Bernstein polynomials. For a HUD using the windshield as combiner, the induced distortion is dependent on the actual surface geometry of the windshield. It may vary between different vehicle models and its design is not driven by optical requirements but rather follow aesthetical or aerodynamic considerations. Furthermore, the distortions are view dependent, so using only the normalized image points as input for the distortion model is not sufficient. Our proposed approach additionally encompasses the head position within a five-variate model in order to address this viewpoint dependence.

As for the perspective correct *rendering on oblique surfaces* our work is related to other approaches describing head-coupled VR displays. Deering [Dee92] and Ware [WAB93] (Fish-Tank VR) were one of the first to propose an immersive 3D planar display system by coupling the head motion to the perspective rendering of 3D objects. Shortly after that the first wide field-of-view immersive environment (CAVE) was created by assembling multiple such displays together [CNSD93]. Technically, either symmetric viewing frustum rendering with homography

[2]The camera-based calibrations by Gao et al. and Gilson et al. [GHA03, GFG08] assume that the camera is placed at exactly the same position relative to the display as the eye of the user. Gilson et al. claim that they can adapt the projection matrix to varying inter-pupillary distances between different users, but they fail to clearly show how this could be achieved.

based post-warping of the image or direct *off-axis rendering* with an asymmetrical frustum can be used as rendering mechanisms. In a more recent work by Harish et al. [HN13] these techniques were compared with respect to visual artifacts, rendering speed and effective image resolution and an 'outside-to-inside' multi-facet screen was realized.

Finally, our calibration method relies on an accurate registration of the image sequence to the vehicle coordinate system. Instead of using external tracking equipment we propose to use our reconstruction, registration and tracking algorithms from the previous chapters for this purpose. Thus, we show that these techniques not only serve for achieving spatially registered motion tracking in video see-through AR applications, but are also suitable for other measurement tasks that require high precision. We integrated the proposed HUD algorithms with the feature map reconstruction and registration of Chap. 5 and demonstrate the feasibility of such an approach.

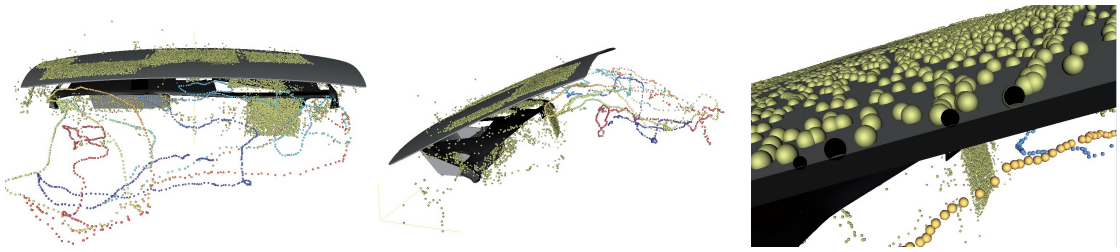
Shortly after our ISMAR publication [WWRF13] on HUD calibration, Itoh and Klinker proposed a calibration for HMDs that shares several commonalities to our proposed approach. In their INDICA method [IK14a, IK14b] they also proposed a view-dependent parametric model with the user-related parameters (position of the eye pupil center) and device-specific parameters (position and orientation of the virtual plane) being separated, in order to adaptively synthesize the correct projection matrix for rendering during runtime. Also - as in our case - the device parameters are retrieved with an offline procedure using a camera that observes known reference patterns through the OST-display. In addition, they mounted back-facing cameras onto the HMD in order to estimate the user's eye centers, eliminating the need of further interaction for user adaptation. In a succeeding work [IK15], they elaborated an alternative distortion model based on a 4D-4D light-ray mapping (light field), with which it becomes also possible to model distortions of the user's view onto the *real scene* (e.g. caused by thick or curved glasses). The parameters of this model are estimated with a machine learning method (Gaussian kernel regression). A general, recent survey on HMD calibration techniques was published by Grubert et al. [GIMS18].

In the final section of this chapter, we will briefly outline how our HUD calibration model and procedure can be transferred to HMD calibration (which was presented as a demonstrator at ISMAR 2014 [WEK*14]). The main difference with regard to Itoh's works is the adaptation to new users, where we propose a simple graphical slider or touch pad interface for quickly adjusting the horizontal and vertical position of each eye relative to the display until a virtual contour matches the outline of a real marker. Itoh's approach can be regarded as superior since it does not require any interaction at all. However, since the current trend in HMD design goes towards miniaturization, cost reduction and energy saving, it is conceivable that future HMDs might not have pre-installed back-facing cameras that allow for eye tracking. Therefore, a calibration method that does not rely on this prerequisite while still allowing a quick setup might still remain useful.

6.3. Accurate camera registration

In order to retrieve the calibration parameters of the HUD by using a camera, it is required that all images are accurately registered to the vehicle coordinate system. In general, this can be achieved by using mechanical measurement arms [FAR], where the camera can be mounted onto, or by employing commercial 'outside-in' trackers [Adv, Vic] with markers attached to the camera. Owen et al. [OZTX04] use a special mechanical construction to calibrate their head-mounted display. Both camera and display are attached to this rig, which allows to measure the translation of the camera relatively to the display in horizontal direction.

A general disadvantage of external trackers - in spite of being expensive - is that they require a two-stage calibration during preparation. First, the camera needs to be registered to the coordinate system of the tracker (hand-eye calibration), and second also the tracker itself must be calibrated to the vehicle coordinate system. This takes an additional amount of time and effort and may also be a source of error accumulation.



(a) (b) (c)

Figure 6.2.: Reconstructed reference feature map and CAD model of the interior of the vehicle after alignment. The golden spheres represent the reconstructed 3D coordinates of the feature points. The camera path is shown with spheres colored from blue (first frame) to red (last frame). (a) Bird's eye view. (b) View from the left hand side of the vehicle. (c) Close-up view that shows that the feature points are well aligned with the outer surface boundary of the windshield. The radius of the spheres is 2 mm.

As an alternative, we propose to directly exploit the localization information contained in the images by using a vision-based tracking method. A problem is, however, that due to the transparency of the windshield there would be large parts in the camera image without any trackable feature points. Therefore, to support the tracking we attached highly textured patterns on the outer side of the windshield, in particular around the viewing area, where the reflected image of the HUD will be observed. The area itself was covered with a black canvas to ensure the robust and accurate detection of the calibration pattern (see Fig. 6.3).

During a preparative stage, the reference feature map for tracking was reconstructed with structure from motion techniques. The map was then transformed to the vehicle coordinate system with a similarity transform by using 3D-3D correspondences of manually selected points on the surface of the CAD-model and reconstructed points of the map (see Chap 5). To this end additional marker points were placed at locations which were accurately identifiable in the CAD-model, such as corners or intersections. The map was finally refined with bundle adjustment constrained to the selected 3D-3D correspondences (Chap. 3 and Sec. 5.5). The reconstruction included a larger area than actually needed for the tracking in order to use widely separated reference points for increasing the accuracy fo the alignment.

Figure 6.2 shows the results of the reconstruction and alignment stage. As can be seen, the feature points are almost perfectly aligned with the CAD model within an error of less than two millimeters. This is the best that can be expected because manufacturing deviations in the automotive industry are also typically in that range. Having the feature map in vehicle coordinates we can compute the camera pose for every image during the actual HUD-calibration phase. Furthermore, the relatively large amount of features allows an estimation with a very low level of jitter.

We make some remarks regarding further enhancements and practical considerations. The preparative reconstruction and alignment phase is the most laborious and time-consuming part in the whole calibration procedure. Typically it takes 30-45 minutes until the setup is finished, which is not tolerable for automotive industries when the calibration is intended to be used within the production line. However, a practical approach could be to construct a special rig or frame with the textured pattern on it, which has the same size as the windshield so that it can be mounted with high reproducibility on every car. Furthermore, we note that we used a camera with a global

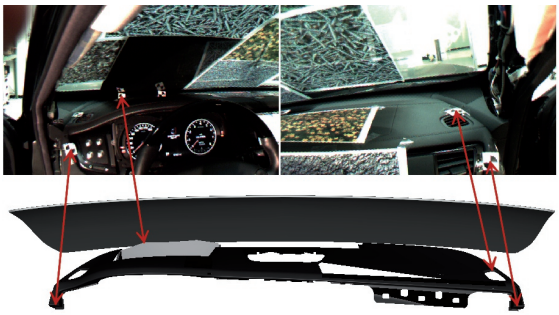


Figure 6.3.: Images from the video sequence during preparation. To increase the accuracy of the preparative SfM-based reconstruction and to reduce jittering of the camera pose tracking during HUD calibration, we attached highly textured patterns on the outer side of the windshield. At points that are accurately identifiable in the CAD model, additional marker points were placed (marked with arrows). These were used for scaled Euclidean transformation of the feature map to the vehicle coordinate system.

shutter[3] throughout all stages of the calibration since rolling shutter effects may deteriorate the reconstruction of the feature map during preparation [HFFR12] and also the estimation of the calibration parameters, afterwards.

6.4. Calibration

In the following sections, we will describe our approach to calibrating the HUD in detail, including target pattern detection, the estimation of the spatial HUD geometry, and the retrieval of the view-dependent distortions.

6.4.1. Reference Pattern Detection

In order to retrieve the calibration parameters, a known target pattern as shown in Fig. 6.4 is displayed on the HUD. The pattern consists of blurred colored blobs. Two center points have a different coloring in order to clearly identify the correct correspondences for the points, even if some point detections are missing due to trimming or because of varying (view-dependent) brightness levels of the image. To speed up the detection, the pattern searching is roughly restricted to the feasible area (dark region in Fig. 6.4).

After detection, the locations of the points are refined by fitting a symmetric Gaussian kernel to the surrounding intensities, i.e. we minimize

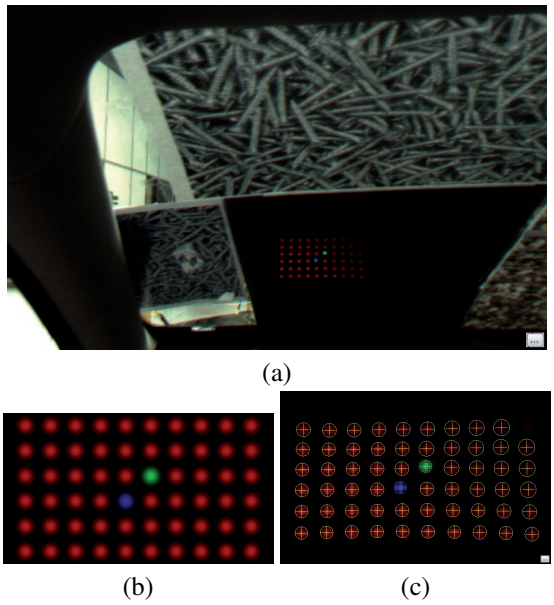
$$\arg\min_{\mathbf{y},c,\sigma^2}\{\sum_{W}(I(\mathbf{y})-c\cdot e^{\mathbf{y}^\mathsf{T}\mathbf{y}/\sigma^2})\}, \tag{6.1}$$

where $\mathbf{y}$ is the desired 2D image point and c and σ are slack variables that are discarded afterwards. The minimum is attained by means of Gauss-Newton iterations similar to the Lukas-Kanade feature tracker [BM04].

6.4.2. Estimation of Spatial Plane Geometry

We will now mathematically describe our method to retrieve the position, orientation and scaling of the virtual image plane, given the estimated camera path, the image measurements and their locations on the target pattern.

[3] We used an IDS UI-1220SE camera with 752-by-480 pixel resolution.



(a)

(b) (c)

Figure 6.4.: Target pattern used for calibration. (a) Pattern as seen from the camera. (b) Original pattern. (c) Close-up view of Fig. (a) with sub-pixel aligned point detections.

From the previous sections, we now have each rotation, $\mathbf{R}_{\mathcal{V}\to\mathcal{C}_n} \in \mathbb{R}^{3\times3}$, and translation, $\mathbf{t}_{\mathcal{V}\to\mathcal{C}_n} \in \mathbb{R}^3$, from vehicle ($\mathcal{V}$) to camera coordinates ($\mathcal{C}_n$) of the n-th image, $n \in \{1,...,N\}$. Further, we have a set of M homogenized reference points, $\mathbf{x}_{m(\mathcal{I})} \in \mathbb{R}^3$ ($m \in \{1,...,M\}$), in normalized coordinates on the display or reference pattern. We denote the local coordinate system of the reference image with the symbol $\mathcal{I}$. For each of the N cameras and M reference points, we may have a corresponding 2D measurement, $\mathbf{y}_{nm}$, i.e. the center points of the detected blobs in the image.

In the normalized[4] coordinate system of the reference image we name the outer four corner points as

$$\begin{aligned}\mathbf{x}_{TL(\mathcal{I})} &:= (-1,-1,1)^\mathsf{T},\\ \mathbf{x}_{TR(\mathcal{I})} &:= (1,-1,1)^\mathsf{T},\\ \mathbf{x}_{BR(\mathcal{I})} &:= (1,1,1)^\mathsf{T},\\ \mathbf{x}_{BL(\mathcal{I})} &:= (-1,1,1)^\mathsf{T}.\end{aligned} \tag{6.2}$$

Now our goal in this section is to estimate the coordinates of these four points in the vehicle coordinate frame, $\mathcal{V}$, i.e. $\mathbf{x}_{TL(\mathcal{V})}$, $\mathbf{x}_{TR(\mathcal{V})}$, $\mathbf{x}_{BR(\mathcal{V})}$, and $\mathbf{x}_{BL(\mathcal{V})}$, to have a geometrical mapping from the image to the virtual plane.[5]

Under the assumption that the virtual plane is rectangular, then there exists a scaled Euclidean transformation (with two scaling parameters) which connects any point on the reference image (in normalized image coordinates) to a point on the virtual plane (vehicle coordinates). We estimate this transformation using the reference points, $\mathbf{x}_{m(\mathcal{I})}$, and their associated measurements, $\mathbf{y}_{nm}$. Afterwards, we can apply this transformation to the four

[4] The coordinates are normalized to the width, w, and height, h, of the display device in pixels.

[5] In fact, three points would be enough, because the fourth point can always be obtained by the other three.

corner points to obtain them in vehicle coordinates. The transformation takes the form

$$\mathbf{x}_{m(\mathcal{V})} = \mathbf{R}_{\mathcal{I}\to\mathcal{V}} \cdot \mathbf{S}_{\mathcal{I}\to\mathcal{V}} \cdot \mathbf{x}_{m(\mathcal{I})} + \mathbf{t}_{\mathcal{I}\to\mathcal{V}}, \tag{6.3}$$

where again $\mathbf{R}_{\mathcal{I}\to\mathcal{V}}$ is a rotation matrix and $\mathbf{t}_{\mathcal{I}\to\mathcal{V}}$ a translation vector and $\mathbf{S}_{\mathcal{I}\to\mathcal{V}}$ is a diagonal scaling matrix,

$$\mathbf{S}_{\mathcal{I}\to\mathcal{V}} = \begin{bmatrix} s_h & & \\ & s_v & \\ & & 1 \end{bmatrix}. \tag{6.4}$$

The relation between the measurements in the camera image and points on the virtual plane in vehicle coordinates is given by a perspective projection [HZ04],

$$\hat{\mathbf{y}}_{nm} = \mathcal{P}(\mathbf{K}_{\mathcal{C}} \cdot (\mathbf{R}_{\mathcal{V}\to\mathcal{C}_n} \cdot \mathbf{x}_{m(\mathcal{V})} + \mathbf{t}_{\mathcal{V}\to\mathcal{C}_n})), \tag{6.5}$$

where $\mathbf{K}_{\mathcal{C}} \in \mathbb{R}^{3\times3}$ is composed of the intrinsic parameters of the camera. The 'hat' symbol refers to predicted values (as opposed to measured), i.e. $\hat{\mathbf{y}}_{nm} = \mathbf{y}_{nm}$ holds only for the noiseless case. $\mathcal{P}(\mathbf{x}) = (x_1/x_3, x_2/x_3)^{\mathsf{T}}$ denotes the perspective division.

Putting everything together, we formulate the task of estimating the plane geometry as finding the set of transformation parameters that minimizes the total re-projection error across all images. This nonlinear least-squares problem can be written as follows:

$$\underset{\mathbf{R}_{\mathcal{I}\to\mathcal{V}},\mathbf{t}_{\mathcal{I}\to\mathcal{V}},s_h,s_v}{\arg\min} \{\sum_{n=1}^{N}\sum_{m=1}^{M} \|\mathbf{y}_{nm} - \hat{\mathbf{y}}_{nm}\|^2\}, \tag{6.6}$$

subject to $R_{\mathcal{I}\to\mathcal{V}}$ being a rotation matrix, so the minimization involves solving for eight DoF in total. The solution is obtained with Levenberg-Marquardt based minimization. The required derivatives have been acquired analytically with a symbolic computation toolbox. In each iteration, the rotation is parametrized locally based on the Lie algebra for rotation groups [AMS07] (see also Sec. 3.6.1).

The initial parameters for minimization were taken from the reference construction model of the integrated HUD, in particular we knew the four corner points of the virtual image according to the reference CAD design. Having 3D-3D correspondences of the corner points in image ($\mathcal{I}$) and vehicle ($\mathcal{V}$) coordinates, one can compute initial values for the transformation parameters with a slightly modified variant[6] of Umeyama's algorithm [Ume91].

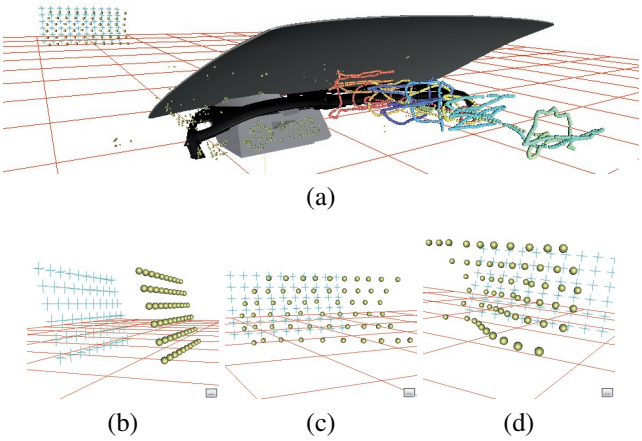
In case that no such information on the reference design of the HUD exists, we reconstruct each of the M reference points individually by triangulation [HS97a] with subsequent nonlinear refinement using all available measurements (at most N per point),

$$\tilde{\mathbf{x}}_{m(\mathcal{V})} = \underset{\mathbf{x}_{m(\mathcal{V})}}{\arg\min}\{\sum_{n=1}^{N} \|\mathbf{y}_{nm} - \mathcal{P}(\mathbf{K}_{\mathcal{C}} \cdot (\mathbf{R}_{\mathcal{V}\to\mathcal{C}_n} \cdot \mathbf{x}_{m(\mathcal{V})} + \mathbf{t}_{\mathcal{V}\to\mathcal{C}_n}))\|^2\}, \tag{6.7}$$

and use the 3D-3D correspondences of the reference points, $\mathbf{x}_{m(\mathcal{I})}$ and $\tilde{\mathbf{x}}_{m(\mathcal{V})}$ in order to obtain an initial estimate of $\mathbf{R}_{\mathcal{I}\to\mathcal{V}}$, $\mathbf{S}_{\mathcal{I}\to\mathcal{V}}$, and $\mathbf{t}_{\mathcal{I}\to\mathcal{V}}$ before further refinement with Eq. 6.6.

Triangulating the points individually also reveals some interesting insights into the actual shape of the virtual plane. As shown in Fig. 6.5 (c) and (d), the reconstructed points are rather located on a curved surface (similar to an excavator shovel), not really on a flat plane. This effect is caused by the distortions from the optics and

[6] Compared to Umeyama [Ume91], we need to solve for two scaling parameters instead of only one.



(a) (b) (c) (d)

Figure 6.5.: Reconstruction of the spatial geometry of the plane. Golden points are the reconstructed points according to different minimization models. Cyan crosses mark corresponding points from the CAD specifications. (a) Camera path for calibration registered to vehicle coordinates (2251 Frames). (b) Estimated plane according to Eq. 6.6. (c/d) Reconstructed points by triangulation Eq. 6.7.

the irregularly curved windshield. However, (as opposed to the HMD calibration of Owen et al. [OZTX04]) reversing this effect is in our case not possible, i.e. the knowledge of the optimal surface geometry of the virtual plane is not sufficient to compensate for the distortions. This conclusion can be drawn from the error analysis in Fig. 6.6. By increasing the number of DoF from 8 (Eq. 6.6) to $3M$ (Eq. 6.7), one would expect that the error is reduced significantly, but only a small improvement can be observed. In the subsequent sections, we will describe a better model for distortion handling.

6.4.3. From Camera Image to Virtual Plane Image Points

We briefly describe the perspective image generation model for rendering on the slanted virtual image plane. The designated intention of this section is twofold. First, we summarize the perspective projection model in terms of the retrieved calibration parameters so that it becomes apparent how these can be applied directly inside the rendering pipeline. Second, for distortion modeling we would like the measurements to be independent of the characteristics of the physical camera used for measuring. We will derive a homographic mapping, $\mathbf{H}_{\mathcal{C}_n \to \mathcal{I}}$, which transforms the measurements, $\mathbf{y}_{mn}$, in the camera image to normalized coordinates in the original reference image or in normalized texture coordinates of the virtual plane. With that we can define a distortion model that only needs the head motion and normalized image coordinates as input.

We denote the extrinsic and intrinsic parameters of the camera that achieves the projective rendering of any 3D scene in vehicle coordinates on the slanted virtual plane with $\mathbf{R}_{\mathcal{V} \to \mathcal{R}}$, $\mathbf{t}_{\mathcal{V} \to \mathcal{R}}$, and $\mathbf{K}_{\mathcal{R}}$. These can be derived analytically in terms of the retrieved calibration parameters, $\mathbf{R}_{\mathcal{I} \to \mathcal{V}}$, $\mathbf{t}_{\mathcal{I} \to \mathcal{V}}$, s_v, and s_h, the parameters of the measuring camera, $\mathbf{R}_{\mathcal{V} \to \mathcal{C}_n}$, $\mathbf{t}_{\mathcal{V} \to \mathcal{C}_n}$, and $\mathbf{K}_{\mathcal{C}}$, and the corner points, $\mathbf{X}_{TL(\mathcal{V})}$ and $\mathbf{X}_{BR(\mathcal{V})}$. For a concise deduction we will draw on a few key characteristics that hold for the projective pinhole camera model.

Regarding the rotation of the rendering camera, any vectors that define the horizontal or vertical direction of the virtual plane, such as $\mathbf{x}_{TR(\mathcal{V})} - \mathbf{x}_{TL(\mathcal{V})}$ and $\mathbf{x}_{BL(\mathcal{V})} - \mathbf{x}_{TL(\mathcal{V})}$, and the plane normal must be mapped onto the

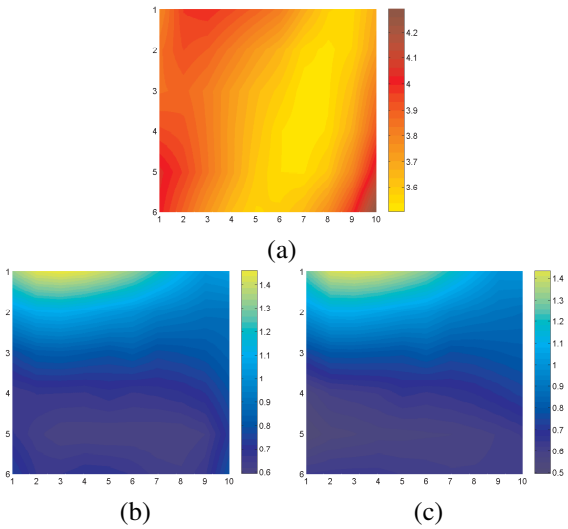


Figure 6.6.: Comparison of the aggregated errors of the different models of the virtual plane. The colors represent the RMSE in pixel units of the observing camera (resolution is 752-by-480 pixels). The rows and columns in the plot belong to each of the 60 reference points on the target pattern. Values in-between are interpolated for better visibility. (a) Plane according to design specifications. (b) Reconstructed plane according to Eq. 6.6. (c) Independent point triangulation according to Eq. 6.7.

unit vectors $\mathbf{e}_x$, $\mathbf{e}_y$ and $\mathbf{e}_z$. This implies that the rendering camera is not necessarily oriented toward the viewing direction but rather - when considering the situation depicted in Fig. 6.7 - towards the footwell area of the car. According to our definition of the four outer points of the virtual image in Eq. 6.2, the rotation is given by the inverse of the image-to-vehicle rotation,

$$\mathbf{R}_{\mathcal{V}\to\mathcal{R}} = \mathbf{R}^{\mathsf{T}}_{\mathcal{I}\to\mathcal{V}}. \tag{6.8}$$

The 3D scene must be rendered from the same viewpoint, no matter where the virtual plane is located or how it is oriented. Thus, the camera positions in the vehicle coordinate system, $-\mathbf{R}^{\mathsf{T}}_{\mathcal{V}\to\mathcal{R}}\cdot\mathbf{t}_{\mathcal{V}\to\mathcal{R}}$ and $-\mathbf{R}^{\mathsf{T}}_{\mathcal{V}\to\mathcal{C}_n}\cdot\mathbf{t}_{\mathcal{V}\to\mathcal{C}_n}$,

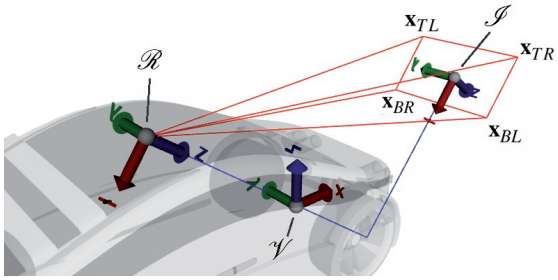


Figure 6.7.: Spatial geometry of the HUD and involved coordinate systems.

must be identical which yields,

$$\mathbf{t}_{\mathcal{V}\to\mathcal{R}} = \mathbf{R}_{\mathcal{V}\to\mathcal{R}} \cdot \mathbf{R}_{\mathcal{V}\to\mathcal{C}_n}^{\mathsf{T}} \cdot \mathbf{t}_{\mathcal{V}\to\mathcal{C}_n}. \tag{6.9}$$

Finally, the rendering must compensate the fact that the viewing direction towards the center of the virtual plane is, in general, not aligned with the normal of the plane. I.e. the intrinsic parameters, $\mathbf{K}_{\mathcal{R}}$, must ensure that the correct region of the image plane is mapped to the unit square $\{[-1,1]\times[-1,1]\}$ by representing an asymmetrical viewing frustum [Dee92]. With

$$\begin{aligned}(t,l,d)^{\mathsf{T}} &:= \mathbf{x}_{TL,(\mathcal{R})} = \mathbf{R}_{\mathcal{V}\to\mathcal{R}} \cdot \mathbf{x}_{TL,(\mathcal{V})} + \mathbf{t}_{\mathcal{V}\to\mathcal{R}}, \text{ and} \\ (b,r,d)^{\mathsf{T}} &:= \mathbf{x}_{BR,(\mathcal{R})} = \mathbf{R}_{\mathcal{V}\to\mathcal{R}} \cdot \mathbf{x}_{BR,(\mathcal{V})} + \mathbf{t}_{\mathcal{V}\to\mathcal{R}}\end{aligned} \tag{6.10}$$

it can be shown that the intrinsics are given by

$$\mathbf{K}_{\mathcal{R}} = \begin{bmatrix} d/s_h & & (t+b)/2s_h \\ & d/s_v & (l+r)/2s_v \\ & & 1 \end{bmatrix}. \tag{6.11}$$

As for the mapping of the measurements, $\mathbf{y}_{nm}$, any 2D point in one of the camera images is related to a point on the virtual plane in normalized texture units by a homography, $\mathbf{H}_{\mathcal{C}_n\to\mathcal{I}}$. Since the transformation is only induced by a rotation around a fixed viewpoint, it is given by an *infinite homography* [HZ04] of the form:

$$\mathbf{H}_{\mathcal{C}_n\to\mathcal{I}} = \mathbf{K}_{\mathcal{R}} \cdot \mathbf{R}_{\mathcal{V}\to\mathcal{R}} \cdot \mathbf{R}_{\mathcal{C}_n\to\mathcal{V}} \cdot \mathbf{K}_{\mathcal{C}}^{-1}. \tag{6.12}$$

The mapping from homogenized predicted image points, $(\hat{\mathbf{y}}_{mn}^{\mathsf{T}}, 1)^{\mathsf{T}}$, to the normalized texture points, $\mathbf{x}_{m(\mathcal{I})}$ is thus given by

$$\mathbf{x}_{m(\mathcal{I})} \sim \mathbf{H}_{\mathcal{C}_n\to\mathcal{I}} \cdot \begin{bmatrix} \hat{\mathbf{y}}_{mn} \\ 1 \end{bmatrix}. \tag{6.13}$$

We can now transform the real measurements, $\mathbf{y}_{mn}$, equivalently which gives us the necessary groundwork to model the distortions as a function of the normalized image plane coordinates in the following section.

6.4.4. Viewpoint-Sensitive Polynomial Model for Distortion Correction

Given the estimated location, orientation, and scaling of the virtual plane, it becomes possible to render the virtual content correctly from any viewpoint of the observer, provided that this setup represents a perspective pinhole camera model sufficiently well. However, the HUD optics and the windshield introduce additional image aberrations that must be compensated.

In this section, we will introduce our distortion model which describes the viewpoint-dependent deviations from the ideal perspective projection. The model does not require any precise knowledge on the windshield shape or the concrete realization of the optics. Rather, it is a purely data driven approach for which we use a generic polynomial and estimate its coefficients based on the observed differences between predicted and measured reference points.

Polynomials are also commonly used for ordinary cameras and projectors [Bro71, Tsa87, DF01, BM07]. One difference of traditional cameras or projectors compared to HUDs is that they usually consist of symmetrical lens optics and thus they also produce a mostly symmetrical distortion with dominant radial components according to a common distortion center. For polynomial distortion models, this leads to a reduced set of polynomial terms and coefficients. Despite being simplified models, they still remain valid for a large variety of camera

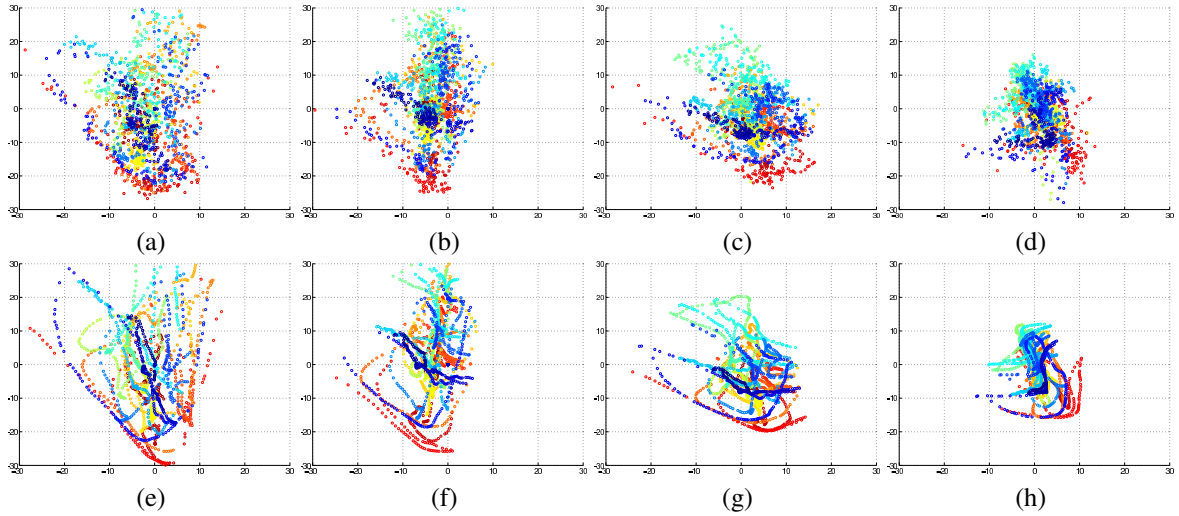


Figure 6.8.: Measured errors from the ideal perspective model (top row) and the estimated distortion according to our polynomial model (bottom row) with a polynomial model of third degree. Errors are in millimeters on the virtual plane, which is located approximately 7.5 meters from the driver. (a/e) Top-left reference point, (b/e) Top-right, (c/g) point in the middle, (d/h) point in the bottom middle of the reference pattern. The coloring is analogous to the camera path depicted in Fig. 6.5(a).

and projector types. However, for a HUD which uses the windshield as the combiner these distortion models are not suitable anymore because the same assumptions on the symmetry of the optics do not hold. Rather, as windshields usually are designed independently of any optical requirements, it is unlikely that any simplified and general model for HUD distortion exists.

An even more important difference is that in standard models the distortion is formulated as a function of the image coordinates only. This is possible as for these devices the center of projection remains always fixed with respect to the image plane. For HUDs the center of projection is, in fact, the eye or the head of the observer, so it varies continuously as he moves relatively to the projection plane. Light rays coming from the same pixel on the displaying unit will travel along different paths through the optics when seen from different viewpoints. As a consequence, the distortion produced by the HUD optics and the windshield will, in general, also be dependent on the current viewpoint (head position) of the observer, $\mathbf{h}_{(\mathcal{V})} =: (x, y, z)^{\mathsf{T}}$. Mathematically this can be expressed as

$$\begin{bmatrix} \tilde{u} \\ \tilde{v} \end{bmatrix} = \begin{bmatrix} u \\ v \end{bmatrix} + \begin{bmatrix} f_u(u, v, x, y, z) \\ f_v(u, v, x, y, z) \end{bmatrix}, \tag{6.14}$$

where for convenience and readability we dropped the coordinate system symbols and denote the 2D normalized image coordinates on the virtual plane as $(u, v)^{\mathsf{T}}$, i.e. $\mathbf{x}_{(\mathcal{I})} =: (u, v, 1)^{\mathsf{T}}$. The observed measurement $\mathbf{y}_{mn}$, transformed with homography from equation 6.12 into the n-th camera frame, is denoted by $(\tilde{u}, \tilde{v})^{\mathsf{T}}$.

Our approach consists of modeling the functions $f_u(.)$ and $f_v(.)$ as generic polynomials of five variables, where each monomial has at most the degree D,

$$f_u(u,v,x,y,z) \approx \sum_{\substack{d_1,d_2,d_3,d_4,d_5 \\ d_i \in \mathbb{N}_0 \\ \Sigma d_i \leq D}} a_{k(d_1,\ldots,d_5)} u^{d_1} v^{d_2} x^{d_3} y^{d_4} z^{d_5}, \tag{6.15}$$

and equivalently,

$$f_v(u,v,x,y,z) \approx \sum_{\substack{d_1,d_2,d_3,d_4,d_5 \\ d_i \in \mathbb{N}_0 \\ \Sigma d_i \leq D}} b_{k(d_1,\ldots,d_5)} u^{d_1} v^{d_2} x^{d_3} y^{d_4} z^{d_5}. \tag{6.16}$$

Depending on the maximal degree of the polynomial, D, the total number, K, of monomials and associated coefficients, $a_{k(.)}$ and $b_{k(.)}$, is given by

$$K = \sum_{d=0}^{D} \frac{(L+d-1)!}{(L-1)!d!}, \tag{6.17}$$

where L is the number of variables (in our case five). As an example, the number of monomials is 56 for $D = 3$ and 126 for $D = 4$.

Given the measurements $(\tilde{u}_{mn}, \tilde{v}_{mn})^\mathsf{T}$, our goal is to determine the optimal coefficients, a_k and b_k, which minimze the following functional,

$$\underset{a_1,\ldots,a_K,b_1,\ldots,b_K}{\arg\min} \{ \sum_{n=1}^{N} \sum_{m=1}^{M} \| \begin{bmatrix} \tilde{u}_{mn} \\ \tilde{v}_{mn} \end{bmatrix} - \begin{bmatrix} u_m \\ v_m \end{bmatrix} - \begin{bmatrix} f_u(u_m, v_m, x_n, y_n, z_n) \\ f_v(u_m, v_m, x_n, y_n, z_n) \end{bmatrix} \|^2 \}. \tag{6.18}$$

This is a linear least-squares problem which can be easily solved by setting up a system of equations of the form,

$$\mathbf{Q} \cdot \begin{bmatrix} \mathbf{a} & \mathbf{b} \end{bmatrix} = \begin{bmatrix} \tilde{\mathbf{u}} - \mathbf{u} & \tilde{\mathbf{v}} - \mathbf{v} \end{bmatrix} \tag{6.19}$$

by stacking the coefficients a_k and b_k and the image points $\tilde{u}_{mn}$,$\tilde{v}_{mn}$,u_m, and u_m into vectors $\mathbf{a}, \mathbf{b} \in \mathbb{R}^K$, and $\tilde{\mathbf{u}}, \mathbf{u}, \tilde{\mathbf{v}}, \mathbf{v} \in \mathbb{R}^{MN}$. The matrix $\mathbf{Q} \in \mathbb{R}^{MN \times K}$ is composed of all monomial terms, i.e.

$$\mathbf{Q} = \begin{bmatrix} 1 & u_1 & u_1^2 & \cdots & u_1^D & u_1 v_1 & u_1^2 v_1 & \cdots & z_1^D \\ & \vdots & & & & \vdots & & & \vdots \\ 1 & u_m & u_m^2 & \cdots & u_m^D & u_m v_m & u_m^2 v_m & \cdots & z_n^D \\ & \vdots & & & & \vdots & & & \vdots \\ 1 & u_M & u_M^2 & \cdots & u_M^D & u_M v_M & u_M^2 v_M & \cdots & z_N^D \end{bmatrix} \tag{6.20}$$

The solution is obtained via the pseudo-inverse $(\mathbf{Q}^\mathsf{T}\mathbf{Q})^{-1}\mathbf{Q}^\mathsf{T}$.

Fig. 6.8 illustrates the viewpoint dependency of the distortion for different reference points of the calibration pattern. The upper row depicts the error vectors between the measured point locations and their ideal, view-dependent perspective predictions using the estimated spatial geometry of the virtual plane according to Eq. 6.6. Each data point corresponds to a single frame in the sequence. Similar colors represent similar viewpoints. From that a clear correlation among the viewpoints and the distortions can be observed. The bottom row in Fig. 6.8 shows the extent to which the five-variate model reproduces this view dependency along the recorded camera path. We chose a polynomial of 3rd degree as an example.

6.5. Two-Pass Rendering

The rendering consists of two passes following Sections 6.4.3 and 6.4.4. First, we render the scene as seen on the virtual image plane and then apply a warping such that the projection on the windshield does not show any distortions.

Projective rendering: To compute the camera transformation (model-view matrix) we evaluate equations (6.8) and (6.9) with respect to the current head position. The camera projection is given by equation (6.11), which is equivalent to constructing a perspective frustum from the frustum corners (6.10) (like in glFrustum). The resulting image is stored in an off-screen texture.

Distortion: The polynomials (6.15) and (6.16) model the windshield distortion. To display the image generated in the previous step correctly, we apply an inverse distortion on the image. This is performed by non-uniform sampling the texture according to the error polynomials. To this end we evaluate the polynomials at equidistant sample points $\mathbf{v}_i = (u, v)$ on the normalized image plane. The sample points $\mathbf{v}_i$ form a uniform polygon, while the sampled points $\tilde{\mathbf{v}}_i = (\tilde{u}, \tilde{v})$ are the corresponding texture coordinates.

6.6. Evaluation

6.6.1. Accuracy versus Model Complexity

We evaluated the accuracy of our distortion model by cross-validation. To this end, we independently captured two different sequences of varying length (1193 and 2251 frames) with the target pattern displayed on the HUD. For each sequence we estimated both the spatial geometry and distortion models of varying polynomial degree. We evaluated the errors in one sequence by using the camera position $-\mathbf{R}_{\mathcal{C}\to\mathcal{V}}^{\mathsf{T}} \cdot \mathbf{t}_{\mathcal{C}\to\mathcal{V}}$ for simulating the head position of a driver. Then, we used the model of the other sequence to make predictions on the actual locations of the target pattern points.

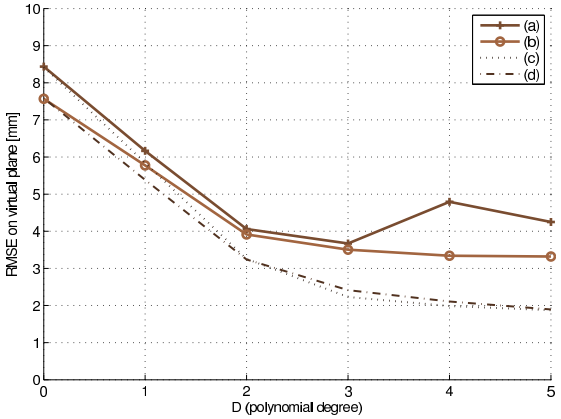


Figure 6.9.: Accumulated average error in millimeters on the virtual plane as a function of the complexity of the distortion model. The model was estimated for two independently captured sequences and evaluated by cross-validation. (a) The errors of sequence 1 (2251 frames) with the estimated model of sequence 2 (1193 frames). (b) Errors of sequence 2 using the first model. (c/d) The errors of the first and second sequences using their own calibration models.

Fig. 6.9 summarizes the results. The dashed and dotted lines (c/d) represent the errors in the sequences when their own model was used for prediction, i.e. these represent the residuals after minimization of the calibration model. They approach asymptotically towards the accuracy of the blob detection method[7] of Sec. 6.4.1 and naturally they should also not go below that. As for the cross validation (solid lines), it can be observed that the error remains on a significantly higher level in both situations. It even increases for a polynomial degree of four and five when applying the calibration model of the shorter sequence to the longer sequence (Fig. 6.9 (a)), which might be seen as a disappointing result at first glance. However, as shown in a further analysis (see Fig. 6.10) the error increase occurs mostly at locations that are underrepresented by the sequence used for calibration. These are also locations where a driver usually would not move his head to (e.g. directly in front of steering wheel or behind his own seat). At locations in between the calibration camera path the model performs almost equally well.

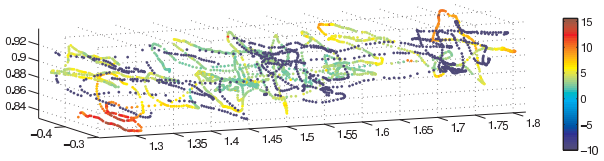


Figure 6.10.: Camera paths of both image sequences. The camera path whose calibration model was used (1193 images) is colored in solid dark blue. The coloring of the other path (2251 images) represents the average measurement-to-model error in millimeters.

6.6.2. Qualitative Validation

Furthermore, we evaluated the view-dependent dynamic rendering of 3D objects on the real HUD display qualitatively. This was realized by using the camera tracking again as a simulation unit for a head tracker. The camera position was sent as 'head-position' to the rendering system of the HUD (by means of a wireless connection). The rendering of the HUD then adapted the perspective rendering and the pre-warping of the image to the updated parameters.

At the same time, the camera image offered a means for qualitative inspection. Having the camera pose, we also rendered the same 3D object in the camera image, so that in the ideal case both virtual objects should overlap. Although there is a permanent lag of one frame (the same camera image is used for tracking as well as for inspection), the calibration model and the rendering can be validated qualitatively when moving the camera very slowly. Fig. 6.11 shows the results for a wireframe model of a tunnel of 60 meters length. As can be seen, both models are well aligned.

6.7. Extension for an User-Friendly HMD Calibration

We have extended our HUD calibration for optical see-through HMDs and presented it as a demonstrator at the ISMAR conference [WEK*14]. The purpose of this demonstrator was twofold. First, we wanted to show that OST-HMD and HUD calibration are in principle very similar and can be realized with common

[7] An error of two millimeters on the virtual plane approximately corresponds to an error of 0.4 pixel units in the image of the measuring camera.

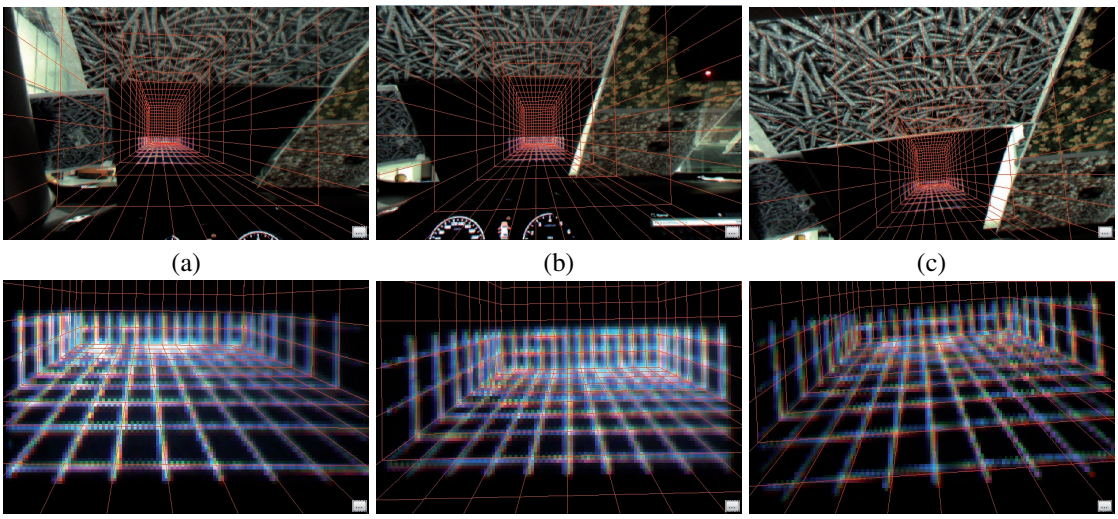


Figure 6.11.: Qualitative evaluation using the camera position as simulated head position of a driver, which is sent to the rendering system on the vehicle. A 3D grid tunnel is visualized both in the camera image (red) and on the HUD (white glow).

techniques. Second, we demonstrated that the idea of separating user and hardware related calibration parameters can be exploited in order to realize a very simple user adaptation as compared to the classical approaches [TN00, GTKN01, GTN02]. At the exhibit, visitors could quickly adjust the calibration to their own conditions and then immediately convince themselves of the compelling registration accuracy by means of small AR demo.

For the calibration, we propose to divide the calibration into two separate stages. In an offline stage, all user-independent parameters are retrieved, in particular the exact orientation and scaling of the two virtual display planes for each eye with respect to the coordinate frame of the OST-HMD built-in camera. In an online stage, the calibration is fine-tuned to user-specific parameters, such as the individual eye-separation distance and height. This leads to a simple and user-friendly procedure, where the user can interactively adjust the remaining parameters until virtual objects (such as marker contours) are well-aligned with their physical counterparts.

Later, this main idea was more deeply explored and further simplified by one of our students [BEK17]. His results are not part of this work.

Preparative offline stage The main objective of the preparative offline stage is to retrieve as many calibration parameters of the HMD as possible, so that the user-specific fine-tuning in the online stage can be simplified. Thus, our HMD calibration is in spirit similar to our approach to head-up display calibration. Again, We use a moving camera that observes a known reference pattern displayed on the HMD from various positions (see Fig. 6.12 and Fig. 6.13). This camera also observes surrounding parts in the scene, where a well-textured known calibration rig is placed. It is used as tracking target to determine the camera position for each frame. The same calibration rig is also observed by the HMD built-in camera. With this setup we can estimate all calibration parameters in the coordinate frame of the built-in camera.

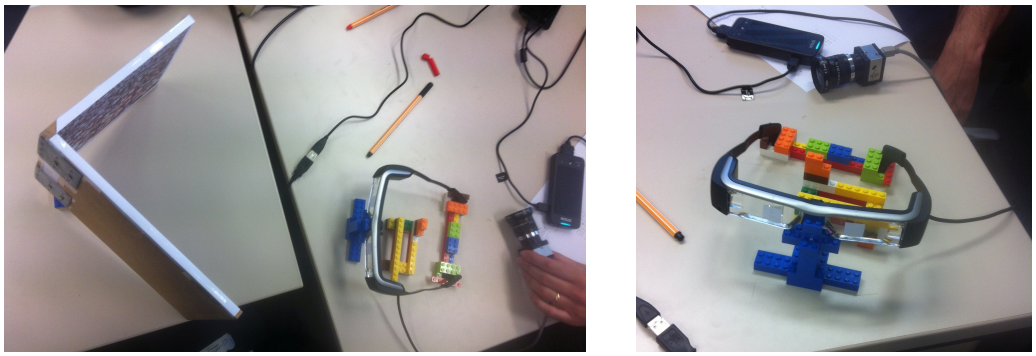
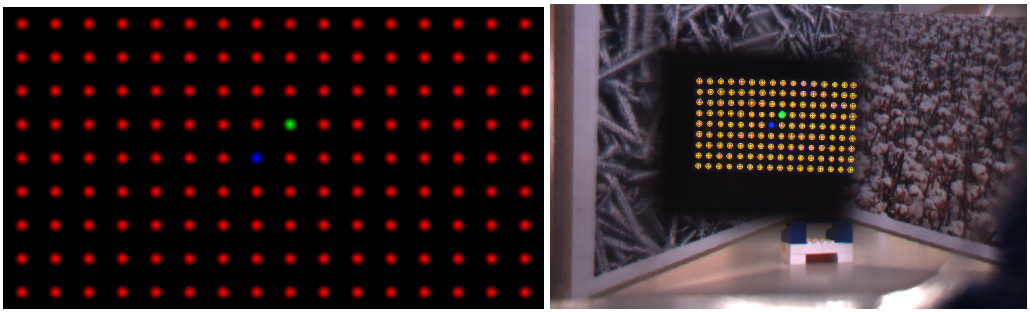


Figure 6.12.: Setup of the offline stage of the OST-HMD calibration.

Having the whole path of the moving camera in the coordinate system of the built-in camera, we can estimate the exact position, orientation and scaling of virtual display planes of each eye. However, for the specific HMD that we used (Epson Moverio 200), it turned out that the virtual planes are located at a very far distance, so that for rendering they can safely assumed to be at infinity. This further simplifies the calibration because we have less parameters that need to be estimated.

Online stage Once the parameters from the offline stage are retrieved, it only remains to determine the position of the user's eyes relative to the built-in camera. We propose a very simple procedure, where the user adjusts interactively the position the eyes by means of a slider-based interface. The user is asked to perform the adjustment until the virtual contours of a marker are well-aligned with the borders of the real marker seen by the user and tracked with the built-in camera. During calibration of one eye the user should close the other one. Once the displayed virtual and real objects match for one particular view they will also do for all other viewpoints. However, calibration becomes more accurate, if the marker is held at a close distance. As the position of the eyes along the z-axis (viewing direction) is less important it can be fixed to a default value, so the user only needs to retrieve four parameters, which makes the calibration simple and fast.



(a) (b)

Figure 6.13.: Target pattern used for HMD calibration. (a) Original pattern. (b) Target pattern as seen from the moving camera with yellow-colored point detections, and the highly textured calibration rig in the background. The relevant viewing area is covered with a black square to simplify the target pattern detection.

6.8. Conclusion

In this chapter, we considered the visualization side of geometric registration for AR. We presented a parametric model and a calibration method for a head-up display (HUD) as one instance of an optical see-through device.

Our model for the HUD consists of the spatial geometry of the virtual image plane, the (potentially moving) observer's location, and a view-dependent, five-variate polynomial distortion mapping for pre-correcting aberrations of the displayed image caused by the windshield and the optics. The required information for parameter estimation is gathered in a preparative phase with a camera that simultaneously observes parts of the real and the displayed virtual domains. The calibration process does not rely on expensive measurement hardware, rather the entire information is drawn from the camera images alone. For registering the camera path to the reference coordinate system of the vehicle, the techniques of the previous chapters of this thesis are used.

Once the calibration is completed, it is possible to visualize a distortion-free image for any head position of the driver and to augment the driver's view with 3D content, which can be modeled in the coordinate system of the vehicle. A major benefit is that the polynomial image distortion model is not only valid for a static viewing position, but rather covers a large volume of possible head positions inside the driver's cabinet. Together with a head tracker, our calibration model can be used as a head-coupled display, so that the virtual image plane acts like a window into the virtual world. It accurately adapts the rendering of the displayed contents to the dynamically changing perspective of the driver.

We have implemented and evaluated our calibration for a single HUD prototype. As our approach does not rely on any specific assumptions on the display configuration, it is reasonable to assume that this method applies to other HUD realizations as well. Moreover, as head-up displays and head-mounted displays share many similarities, we also presented an extension of this method to HMD calibration. In this case, our proposed separation between hardware and user related parameters offers an easy way of adapting the calibration model to different users by means of a user-friendly fine-tuning interface.

7. Conclusions

In this thesis, we investigated several aspects in the field of geometric registration for augmented reality (AR). Geometric registration as a generic term encompasses a multitude of methods with the goal that the displayed virtual content appears in the right position and perspectively correct from the user's view onto reality.

Geometric registration for AR is a dynamic motion tracking problem in the first place, requiring continuous estimations of the camera (or user) movements in real time. Therefore, a major focus of past research has been laid on the exploration of various SLAM techniques that allow to capture simultaneously the environment and the relative motion path of the camera. Tremendous progress has been made, and as a result, highly performant and robust solutions emerged.

The fact that SLAM systems are only targeted towards *relative* motion tracking means that they have no geometric connection to an absolute reference. Pure SLAM-based methods work in a self-selected coordinate system to represent the estimated camera path and the reconstructed real-world model. However, in order to visualize pre-existing virtual content in a coherent manner with regard to reality, the spatial relation between the reconstructed real-world model and this content must also be known or defined. An obvious possibility is to integrate the virtual models interactively into the SLAM model by manual editing until the overlay looks plausible (*SLAM-centric authoring*, see Sec. 5.2). However, due to the tight coupling of both models, reusing the created content in a different setup or exchanging the tracking model (or method) then becomes difficult. Rather than defining the virtual content relative to a randomly chosen and non-controllable coordinate system, it seems more appropriate to spatially register the SLAM model itself to the virtual or predefined target coordinate system.

Another problem is that SLAM alone cannot provide a consistent or even metric reconstruction that meets the accuracy requirements in many industrial applications, even if the results are globally optimized via bundle adjustment (BA). This is because BA is a poorly-conditioned problem due to the large number of parameters to be estimated, in which small errors in the measurements can lead to large changes in the estimated values (see Sec. 3.1). As a result, reconstructions of the environment and camera paths calculated by the BA are often deformed. In practical AR applications, this manifests itself in a perceivable, systematic offset of augmentations, when the user changes his location and observes the scene from a different viewpoint. It is likely that SLAM systems will continue to be further improved in future, but it also seems reasonable to assume that the necessary effort to achieve higher accuracy could be very large. Even a highly integrated system equipped with a variety of cameras, depth sensors, and inertial units, such as today's Hololens, cannot sufficiently solve this problem. Therefore, instead of relying only on the SLAM as the authoritative source of truth for geometric registration, available scene knowledge should be fed back and integrated into the computations in order to correct its inaccuracies.

Spatial registration can be solved via CAD-model-based tracking that automatically recognizes elements of the virtual scene in the image feed of a camera. However, this requires the availability of corresponding CAD models that match reality sufficiently well, and appropriate algorithms must also be applicable in this context (Sec. 1.2). Moreover, a fully automatic ad-hoc registration on untextured models is currently not possible, so that still a certain amount of aid from the end-user is required at runtime of the application. Using additional assumptions or sensors (GPS or Mahhattan-World assumption), some automatic registration methods for urban outdoor environments exist (Sec 5.2), but their general applicability is restricted, as they are bound by these extra

conditions. Automation and general applicability are often conflicting goals, asking to what extent a user can be supported with easy-to-use tools and algorithms that allow highly reproducible results in a wide application context.

In addition to the geometric alignment of SLAM, another registration problem relates to the way the virtual information is visually presented to the user. When using optical see-through (OST) devices, virtual objects are typically displayed via a semi-transparent reflector as combiner. In this setup, the user observes the surrounding reality from his own perspective and not from the viewpoint of the tracking camera (as in video see-through AR). While it is difficult, if not impossible to directly access the user's personal perception, it is nonetheless important to have an abstract model that emulates this situation as close as possible. It is necessary to know, how the rendering can be controlled accordingly so that the virtual content most likely appears seamlessly integrated into his view on the real world. This demands for calibration methods to retrieve the parameters of such a model in a quantitative manner.

In the present work, we have addressed the above-mentioned problems with the aim of fusing virtual worlds and reality in a geometrical sense. As stated at the beginning of this work (see Sec. 1.1), this requires equivalence relations that define what belongs together in each of the respective domains, and we posed the following questions:

- What is the source and how is this information provided (*data association*)?
- In which way can it be exploited algorithmically to the best extent possible for spatial registration of a SLAM system (*algorithmic exploitation*)?
- How do these equivalence relations actually appear in optical see-through devices and how can they be measured and used for display calibration (*modeling and measuring geometric perception in OST-devices*)?

The contributions of this thesis regarding these questions can be summarized as follows.

Data association: In this work, we proposed methods that allow a user to contribute with his context knowledge by providing correspondences between the real environment and the virtual world for spatial registration of SLAM systems. With a human operator in the loop who understands a scene and can make informed decisions about it, it becomes possible to cover situations where the automatic association is difficult and to retrieve implicit user knowledge that may be hard to formalize. We allow that the information might be provided in sparse and incomplete form, which serves to map the user knowledge in a more targeted manner and to simplify the process of information transfer from the user.

In a first variant, we have proposed a preparative process for setting up and registering a tracking system based on *natural features* (Chap. 5). Based on a pre-recorded and reconstructed image sequence, a simple user interface helps to establish links between the coordinate systems of the virtual scene and the SLAM. The user can browse through the sequence, select reconstructed feature points, and assign these to corresponding 3D points of a virtual model. In addition, this preparative phase is also used to retrieve additional information in an automated fashion. A feature classifier for tracking initialization and a feature management system for visibility handling are trained using the input sequence. The output is a registered and ready-to-use tracking model for a broad range of AR applications. The system was used in several industrial and academic AR projects, and its accuracy and general applicability was proved at various public benchmarks with high success.

As a second variant, we present a spatial registration approach for SLAM systems, in which the user specifies the connections to the virtual coordinate system by placing some *reference markers* in the scene (Chap. 4). The important part is that the position and orientation for each of the reference markers only needs to be defined partially, i.e. they can be positioned freely on edges or surfaces, which notably simplifies the setup. The registration information is encoded in the partial references to the virtual coordinate system of each marker,

and the geometric arrangement of these distributed connections over the entire marker setup. During runtime, spatial registration is accomplished automatically once enough reference markers have been reconstructed by the SLAM. The approach was implemented as a pure marker-based method, but it can be easily combined with a feature-based reconstruction. It was developed for an industrial quality inspection AR-application (Sec. 1.5). In our evaluations, we could show that the proposed system attained an absolute accuracy of 1mm or below.

Algorithmic exploitation: A central part of this thesis is the optimal exploitation of the sparse and distributed partial information for spatial registration provided by the user. This information is internalized into *closed-form* as well as *iterative*, nonlinear minimization formulations.

We presented a *closed-form algorithm* (Chap. 2) to determine the parameters of a Euclidean or similarity transformation based on different types of correspondences, namely point-to-point, point-to-line, and point-to-plane. It comprises several advantageous properties: it is applicable to minimal and over-constrained configurations, it has linear complexity, and it returns all minima at once in case of ambiguities. Our algorithm represents a generalization of several, most recent algorithms for absolute pose problems, including the well-known perspective-n-point (PnP) problem. Apart from being more general, it is also faster than existing algorithms. Due to its general applicability, the algorithm also optimally solves other problems in computer vision, such as the perspective-n-line (PnL) problem and generalized variants of PnP and PnL for multi-camera systems (GPnP and GPnL). In the context of this work, the algorithm was used for spatial registration between the virtual and the real word domain. In particular, it was indispensable for achieving spatial registration of the marker-based SLAM of Chap. 4 with reference markers positioned on surfaces (point-to-plane) or edges (point-to-line) that in this way only provided partially indeterminate links to the virtual model.

In Chap. 3 we proposed a variant of the *iterative bundle adjustment*, in which the links to the virtual target coordinate system are imposed as constraints. Instead of using the traditional Lagrangian method as a generic tool for constrained problems, we resort to optimization-on-manifold techniques, with the benefit of preserving the least squares character of the problem and reducing rather than increasing the total number of optimization variables. Our constrained BA minimization comprises three steps. After a transformation of the parameters into the coordinate system of the constraints (1), the parameters are projected onto their respective constraint manifolds (2). For the iterative minimization of the objective function (3), the constraint manifolds are modeled via appropriate derivative functions and parameter update rules along geodesic paths. For this last (third) step, we also analyzed to what extent existing sparse minimization frameworks can be used as a substitute for our own BA implementation. Our proposed constrained BA variant is capable of compensating the low-frequency deformations caused by the ill-conditionedness of this minimization problem to a large extent. This was demonstrated in each of the use-cases of Chap. 4 and Chap. 5.

Modeling and measuring geometric perception in OST-devices: For the calibration of optical see-through (OST) displays, we present a parametric display model, in which user-related and hardware-related parameters are decoupled from one another. This is particularly useful, as user related data, such as the head position in head-up displays (HUD) or the exact eye centers in head-mounted displays (HMD), is subject to frequent variations, whereas the hardware parameters generally remain constant throughout the lifetime of the device. Our calibration model includes a parametric perspective camera model and a generic multivariate polynomial distortion model for pre-compensation of view-dependent aberrations by the optics and the combiner.

In the calibration procedure presented in this thesis, a moving camera is used to emulate the users perception from various viewpoints and to see how virtual content displayed on the screen appears with regard to reality. As the parameters need to be determined directly in the reference coordinate system, it is necessary that the camera itself is also registered with this reference system. Thus, our display calibration can be considered as a further

use-case of the above-described methods for the spatial registration of the real model and the camera path to a desired target coordinate system.

A. Publications

The majority of the work described in this thesis has been peer-reviewed and presented at conferences and in journals. This is a list of the publications derived from this work:

1. WIENTAPPER F., SCHMITT M., FRAISSINET-TACHET M., KUIJPER A.: A universal, closed-form approach for absolute pose problems. *Computer Vision and Image Understanding (CVIU)* (2018) — [WSFTK18]
2. WIENTAPPER F., KUIJPER A.: Unifying algebraic solvers for scaled Euclidean registration from point, line and plane constraints. In *Proc. Asian Conf. on Computer Vision (ACCV)* (2017), pp. 52–66 — [WK17]
3. WIENTAPPER F., WUEST H., ROJTBERG P., FELLNER D.: A camera-based calibration for automotive augmented reality head-up-displays. In *IEEE Proc. Int'l Symp. on Mixed and Augmented Reality (ISMAR)* (Oct 2013), pp. 189–197 — [WWRF13]
4. WIENTAPPER F., ENGELKE T., KEIL J., WUEST H., MENSIK J.: [demo] user friedly calibration and tracking for optical stereo see-through augmented reality. In *IEEE Proc. Int'l Symp. on Mixed and Augmented Reality (ISMAR)* (Sept 2014), pp. 385–386 — [WEK*14]
5. WIENTAPPER F., WUEST H., KUIJPER A.: Composing the feature map retrieval process for robust and ready-to-use monocular tracking. *Computers & Graphics 35*, 4 (2011), 778 – 788 — [WWK11a]
6. WIENTAPPER F., WUEST H., KUIJPER A.: Reconstruction and accurate alignment of feature maps for augmented reality. In *IEEE Proc. of Int'l Conf. on 3D Imaging, Modeling, Processing, Visualization and Transmission (3DIMPVT)* (2011), pp. 140–147 — [WWK11b]

The following patent has been published based on the work on HUD-calibration:

7. GIEGERICH P., WIENTAPPER F., WUEST H.: Method and apparatus for controlling an image generating device of a head-up display. Patent. WO/2015/044280, 02 04, 2015 — [GWW15]

The following publications describe applications that have been realized using the presented methods and algorithms:

8. ENGELKE T., KEIL J., ROJTBERG P., WIENTAPPER F., SCHMITT M., BOCKHOLT U.: Content first: A concept for industrial augmented reality maintenance applications using mobile devices. In *ACM Proc. of the Multimedia Systems Conference (MMSys)* (New York, NY, USA, 2015), MMSys '15, ACM, pp. 105–111 — [EKR*15]
9. KEIL J., ZOELLNER M., ENGELKE T., WIENTAPPER F., SCHMITT M.: Controlling and filtering information density with spatial interaction techniques via handheld augmented reality. In *Virtual Augmented and Mixed Reality. Designing and Developing Augmented and Virtual Environments* (Berlin, Heidelberg, 2013), Shumaker R., (Ed.), Springer Berlin Heidelberg, pp. 49–57 — [KZE*13]
10. ENGELKE T., KEIL J., ROJTBERG P., WIENTAPPER F., WEBEL S., BOCKHOLT U.: Content first - a concept for industrial augmented reality maintenance applications using mobile devices. In *IEEE Proc. Int'l Symp. on Mixed and Augmented Reality (ISMAR)* (Oct 2013), pp. 251–252 — [EKR*13]

11. KEIL J., ZÖLLNER M., BECKER M., WIENTAPPER F., ENGELKE T., WUEST H.: The house of olbrich — an augmented reality tour through architectural history. In *IEEE Proc. Int'l Symp. on Mixed and Augmented Reality - Arts, Media, and Humanities (ISMAR-AMH)* (Oct 2011), pp. 15–18 — [KZB*11]

Publications in the related field of model-based tracking have also been co-authored. They are only succinctly discussed and not core to this work:

12. WUEST H., ENGEKLE T., WIENTAPPER F., SCHMITT F., KEIL J.: From CAD to 3D tracking — enhancing & scaling model-based tracking for industrial appliances. In *IEEE Proc. Int'l Symp. on Mixed and Augmented Reality (ISMAR-Adjunct)* (Sept 2016), pp. 346–347 — [WEW*16]
13. WUEST H., WIENTAPPER F., STRICKER D.: Adaptable model-based tracking using analysis-by-synthesis techniques. In *Proc. Int'l Conf. on Computer Analysis of Images and Patterns (CAIP)* (2007), vol. 4673, pp. 20–27 — [WWS07]
14. WUEST H., WIENTAPPER F., STRICKER D.: Acquisition of high quality planar patch features. In *Proc. Int'l Symp. on Advances in Visual Computing (ISVC)* (2008), pp. 530–539 — [WWS08]

Not related to the present thesis are the following publications:

15. WIENTAPPER F., AHRENS K., WUEST H., BOCKHOLT U.: Linear-projection-based classification of human postures in time-of-flight data. In *IEEE Proc. Int'l Conf. on Systems, Man and Cybernetics (SMC)* (Oct 2009), pp. 559–564 — [WAWB09]
16. AMORETTI M., COPELLI S., WIENTAPPER F., FURFARI F., LENZI S., CHESSA S.: Sensor data fusion for activity monitoring in the PERSONA ambient assisted living project. *Journal of Ambient Intelligence and Humanized Computing 4*, 1 (Feb 2013), 67–84 — [ACW*13]
17. AMORETTI M., WIENTAPPER F., FURFARI F., LENZI S., CHESSA S.: Sensor data fusion for activity monitoring in ambient assisted living environments. In *Sensor Systems and Software* (Berlin, Heidelberg, 2010), Hailes S., Sicari S., Roussos G., (Eds.), Springer Berlin Heidelberg, pp. 206–221 — [AWF*10]

B. Supervising Activities

The following list summarizes the student bachelor, diploma, and master thesis supervised by the author.

1. AHRENS K.: *Feature-basiertes Tracking mittels Bag of Visual Words*. Master's thesis, Hochschule Darmstadt — Universitry of Applied Sciences, Schöfferstraße 8b, 64295 Darmstadt, Germany, Oct 2011. Received the **Preis des Fachbereichs Informatik der Hochschule Darmstadt** award. — [Ahr11]
2. AHRENS K.: *Erkennung der menschlichen Körperhaltung mit Hilfe einer Tiefenbildkamera*. Bachelor's thesis, Hochschule Darmstadt — Universitry of Applied Sciences, Schöfferstraße 8b, 64295 Darmstadt, Germany, May 2009. — [Ahr09]

In the years from 2008 to 2017, the author also contributed to the lecture *Virtual and Augmented Reality* at the TU Darmstadt and recited there on the subject *Feature Extraction, Matching, and Pose Estimation*.

Bibliography

[AAL*09] ANDERSEN M., ANDERSEN R., LARSEN C., MOESLUND T. B., MADSEN O.: Interactive assembly guide using augmented reality. In *Proc. Int'l Symp. on Advances in Visual Computing (IJVC)*, Bebis G., Boyle R., Parvin B., Koracin D., Kuno Y., Wang J., Wang J.-X., Wang J., Pajarola R., Lindstrom P., Hinkenjann A., Encarnação M. L., Silva C. T., Coming D., (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 2009, pp. 999–1008. 4, 86

[ACW*13] AMORETTI M., COPELLI S., WIENTAPPER F., FURFARI F., LENZI S., CHESSA S.: Sensor data fusion for activity monitoring in the PERSONA ambient assisted living project. *Journal of Ambient Intelligence and Humanized Computing 4*, 1 (Feb 2013), 67–84. 134

[AD03] ANSAR A., DANIILIDIS K.: Linear pose estimation from points or lines. *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI) 25*, 5 (May 2003), 578–589. 16

[Adv] ADVANCED REALTIME TRACKING GMBH:. http://www.ar-tracking.com/technology/markers/. Accessed: 2013-04-13. 114

[AFS*11] AGARWAL S., FURUKAWA Y., SNAVELY N., SIMON I., CURLESS B., SEITZ S. M., SZELISKI R.: Building rome in a day. *Commun. ACM 54*, 10 (Oct. 2011), 105–112. 37, 39, 62, 82

[AHB87] ARUN K. S., HUANG T. S., BLOSTEIN S. D.: Least-squares fitting of two 3-D point sets. *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI) PAMI-9*, 5 (Sept 1987), 698–700. 13, 15, 30, 93

[Ahr09] AHRENS K.: *Erkennung der menschlichen Körperhaltung mit Hilfe einer Tiefenbildkamera*. Bachelor's thesis, Hochschule Darmstadt — Universitry of Applied Sciences, Schöfferstraße 8b, 64295 Darmstadt, Germany, May 2009. 135

[Ahr11] AHRENS K.: *Feature-basiertes Tracking mittels Bag of Visual Words*. Master's thesis, Hochschule Darmstadt — Universitry of Applied Sciences, Schöfferstraße 8b, 64295 Darmstadt, Germany, Oct 2011. Received the **Preis des Fachbereichs Informatik der Hochschule Darmstadt** award. 135

[AHRL17a] ARMAGAN A., HIRZER M., ROTH P., LEPETIT V.: Accurate camera registration in urban environments using high-level feature matching. In *Proc. of the British Machine Vision Conf. (BMVC)* (September 2017), BMVA Press. 88

[AHRL17b] ARMAGAN A., HIRZER M., ROTH P. M., LEPETIT V.: Learning to align semantic segmentation and 2.5D maps for geolocalization. In *IEEE Proc. Conf. on Computer Vision and Pattern Recognition (CVPR)* (July 2017), pp. 4590–4597. 88

[AKr12] ASK E., KUANG Y., ÅSTRÖM K.: Exploiting p-fold symmetries for faster polynomial equation solving. In *IEEE Proc. Int'l Conf. on Pattern Recognition (ICPR)* (Nov 2012), pp. 3232–3235. 26

[AMO] AGARWAL S., MIERLE K., OTHERS: Ceres solver. http://ceres-solver.org(accessed 18 March 2017). 39, 57, 61, 62

[AMS07] ABSIL P.-A., MAHONY R., SEPULCHRE R.: *Optimization Algorithms on Matrix Manifolds*. Princeton University Press, 2007. 46, 54, 118

[App17] APPLE INC.: ARKit. https://developer.apple.com/arkit/, 2017. Accessed: 2017-09-14. 2, 84, 85

[APV*15] ARTH C., PIRCHHEIM C., VENTURA J., SCHMALSTIEG D., LEPETIT V.: Instant outdoor localization and SLAM initialization from 2.5D maps. *IEEE Transactions on Visualization and Computer Graphics (TVCG) 21*, 11 (Nov 2015), 1309–1318. 88

[ARTC12] AGRAWAL A., RAMALINGAM S., TAGUCHI Y., CHARI V.: A theory of multi-layer flat refractive geometry. In *IEEE Proc. Conf. on Computer Vision and Pattern Recognition (CVPR)* (June 2012), pp. 3346–3353. 15

[ASS*09] AGARWAL S., SNAVELY N., SIMON I., SEITZ S. M., SZELISKI R.: Building rome in a day. In *IEEE Proc. Int'l Conf. on Computer Vision (ICCV)* (Sept 2009), pp. 72–79. 43

[ASSS10] AGARWAL S., SNAVELY N., SEITZ S. M., SZELISKI R.: Bundle adjustment in the large. In *Proc. European Conf. on Computer Vision (ECCV)* (Berlin, Heidelberg, 2010), Daniilidis K., Maragos P., Paragios N., (Eds.), Springer Berlin Heidelberg, pp. 29–42. 39, 43, 46, 62, 82

[ATR10] AGRAWAL A., TAGUCHI Y., RAMALINGAM S.: Analytical forward projection for axial non-central dioptric and catadioptric cameras. In *Proc. European Conf. on Computer Vision (ECCV)*, Daniilidis K., Maragos P., Paragios N., (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 2010, pp. 129–143. 15

[AWF*10] AMORETTI M., WIENTAPPER F., FURFARI F., LENZI S., CHESSA S.: Sensor data fusion for activity monitoring in ambient assisted living environments. In *Sensor Systems and Software* (Berlin, Heidelberg, 2010), Hailes S., Sicari S., Roussos G., (Eds.), Springer Berlin Heidelberg, pp. 206–221. 134

[AWK*09] ARTH C., WAGNER D., KLOPSCHITZ M., IRSCHARA A., SCHMALSTIEG D.: Wide area localization on mobile phones. In *IEEE Proc. Int'l Symp. on Mixed and Augmented Reality (ISMAR)* (2009), pp. 73–82. 95

[Azu97] AZUMA R. T.: A survey of augmented reality. *Presence: Teleoperators and Virtual Environments 6*, 4 (1997), 355–385. 1

[Azu16] AZUMA R. T.: The most important challenge facing augmented reality. *Presence: Teleoperators and Virtual Environments 25*, 3 (2016), 1–30. 3

[BÅ09] BYRÖD M., ÅSTRÖM K.: Bundle adjustment using conjugate gradients with multiscale preconditioning. In *Proc. of the British Machine Vision Conf. (BMVC)* (2009), BMVA Press, pp. 36.1–36.10. doi:10.5244/C.23.36. 43

[BÅ10] BYRÖD M., ÅSTRÖM K.: Conjugate gradient bundle adjustment. In *Proc. European Conf. on Computer Vision (ECCV)* (Berlin, Heidelberg, 2010), Daniilidis K., Maragos P., Paragios N., (Eds.), Springer Berlin Heidelberg, pp. 114–127. 43

[BAC*16] BERGAMASCO F., ALBARELLI A., COSMO L., RODOLÀ E., TORSELLO A.: An accurate and robust artificial marker based on cyclic codes. *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI) 38*, 12 (Dec 2016), 2359–2373. 68

[BB02] BURCH D., BRAASCH M.: Enhanced head-up display for general aviation aircraft. In *IEEE Proc. Digital Avionics Systems Conf.* (2002), vol. 2, pp. 11C6–1–11C6–11. 111

[BD14] BANSAL M., DANIILIDIS K.: Geometric urban geo-localization. In *IEEE Proc. Conf. on Computer Vision and Pattern Recognition (CVPR)* (June 2014), pp. 3978–3985. 88

[BEK17] BERNARD F., ENGELKE T., KUIJPER A.: User friendly calibration for tracking of optical stereo see-through head worn displays for augmented reality. In *IEEE Proc. Int'l Conf. on Cyberworlds (CW)* (Sept 2017), pp. 33–40. 126

[Bel03] BELL B.: CppAD: A package for differentiation of C++ algorithms. http://www.coin-or.org/CppAD, 2003. Accessed: 2017-06-23. 57

[Ber99] BERTSEKAS D. P.: *Nonlinear programming*, second ed. Athena scientific Belmont, 1999. 38, 44

[BH93] BURGESS M., HAYES R.: Synthetic vision-a view in the fog. *IEEE Aerospace and Electronic Systems Magazine 8*, 3 (1993), 6–13. 111

[BJM07] BHASKER E., JUANG R., MAJUMDER A.: Registration techniques for using imperfect and partially calibrated devices in planar multi-projector displays. *IEEE Trans. on Visualization and Computer Graphics (TVCG) 13*, 6 (2007), 1368–1375. 113

[BL95] BASU A., LICARDIE S.: Alternative models for fish-eye lenses. *Pattern Recognition Letters 16*, 4 (1995), 433–441. 113

[BLE03] BÖRLIN N., LINDSTRÖM P., ERIKSSON J.: A globally convergent gauss-newton algorithm for the bundle adjustment problem with functional constraints. In *Optical 3-D measurement techniques : applications in GIS, mapping, manifactoring, quality control, robotics, navigation, mobile mapping, medical imaging, VR generation and animation*, vol. 2 of *Optical 3-D Measurement Techniques*. Wichmann-Verlag, 2003, pp. 269–276. 45

[BM04] BAKER S., MATTHEWS I.: Lucas-kanade 20 years on: A unifying framework. *Int'l J. of Computer Vision (IJCV) 56*, 3 (2004), 221–255. 37, 116

[BM07] BHASKER E., MAJUMDER A.: Geometric modeling and calibration of planar multi-projector displays using rational bezier patches. In *IEEE Proc. Conf. on Computer Vision and Pattern Recognition (CVPR)* (2007), pp. 1–8. 113, 121

[BNR02] BARATOFF G., NEUBECK A., REGENBRECHT H.: Interactive multi-marker calibration for augmented reality applications. In *IEEE/ACM Proc. Int'l Symp. on Mixed and Augmented Reality (ISMAR)* (Washington, DC, USA, 2002), ISMAR '02, IEEE Computer Society, pp. 107–. 68, 69

[Bou] BOUGUET J.-Y.: Camera calibration toolbox for matlab. http://www.vision.caltech.edu/bouguetj/calib_doc/. Accessed: 2019-01-04. 65

[BPR15] BAYDIN A. G., PEARLMUTTER B. A., RADUL A. A.: Automatic differentiation in machine learning: a survey. *CoRR abs/1502.05767* (2015). 56

[Bro71] BROWN D. C.: Close-range camera calibration. *Photogrametric Engineering 37*, 8 (1971), 855–866. 64, 113, 121

[BS96] BENDTSEN C., STAUNING O.: *FADBAD, a flexible C++ package for automatic differentiation*. Tech. Rep. IMM-REP-1996-17, Department of Mathematical Modelling, Technical University of Denmark, 1996. 57

[BS09] BLESER G., STRICKER D.: Advanced tracking through efficient image processing and visual-inertial sensor fusion. *Computer & Graphics (C&G) 33*, 1 (2009), 59–72. 96, 97

[BSKP12] BAATZ G., SAURER O., KÖSER K., POLLEFEYS M.: Large scale visual geo-localization of images in mountainous terrain. In *Proc. European Conf. on Computer Vision (ECCV)* (Berlin, Heidelberg, 2012), Fitzgibbon A., Lazebnik S., Perona P., Sato Y., Schmid C., (Eds.), Springer Berlin Heidelberg, pp. 517–530. 87

[BSS06] BAZARAA M. S., SHERALI H. D., SHETTY C. M.: *Nonlinear programming: theory and algorithms*, 3rd ed. John Wiley & Sons, 2006. 38, 44

[BvES11] BABOUD L., ČADÍK M., EISEMANN E., SEIDEL H. P.: Automatic photo-to-terrain alignment for the annotation of mountain pictures. In *IEEE Proc. Conf. on Computer Vision and Pattern Recognition (CVPR)* (June 2011), pp. 41–48. 88

[BWS06] BLESER G., WUEST H., STRICKER D.: Online camera pose estimation in partially known and dynamic scenes. In *IEEE/ACM Proc. Int'l Symp. on Mixed and Augmented Reality (ISMAR)* (2006), pp. 56–65. 81, 87, 90

[CB14] COLLINS T., BARTOLI A.: Infinitesimal plane-based pose estimation. *Int'l J. of Computer Vision (IJCV) 109*, 3 (2014), 252–286. 14

[CC04] CHEN C.-S., CHANG W.-Y.: On pose recovery for generalized visual sensors. *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI) 26*, 7 (July 2004), 848–861. 16

[CC12] CHOI C., CHRISTENSEN H. I.: 3D textureless object detection and tracking: An edge-based approach. In *IEEE/RSJ Proc. Int'l Conf. on Intelligent Robots and Systems (IROS)* (Oct 2012), pp. 3877–3884. 4, 86

[CCC*16] CADENA C., CARLONE L., CARRILLO H., LATIF Y., SCARAMUZZA D., NEIRA J., REID I., LEONARD J. J.: Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age. *IEEE Trans. on Robotics 32*, 6 (Dec 2016), 1309–1332. 37, 83

[CDH08] CHEN Y., DAVIS T. A., HAGER W. W.: Algorithm 887: Cholmod, supernodal sparse cholesky factorization and update/downdate. *ACM Trans. on Mathematical Software (TOMS)* (2008), 1–14. 43, 61, 62, 63

[CGC14] CHU H., GALLAGHER A., CHEN T.: GPS refinement and camera orientation estimation from a single image and a 2D map. In *IEEE Proc. Conf. on Computer Vision and Pattern Recognition Workshops (CVPRW)* (June 2014), pp. 171–178. 88

[Che90] CHEN H. H.: Pose determination from line-to-plane correspondences: existence condition and closed-form solutions. In *IEEE Proc. Int'l Conf. on Computer Vision (ICCV)* (Dec 1990), pp. 374–378. 15, 16, 23

[CLO07] COX D. A., LITTLE J., O'SHEA D.: *Ideals, Varieties, and Algorithms: An Introduction to Computational Algebraic Geometry and Commutative Algebra, third edition (Undergraduate Texts in Mathematics)*. Springer-Verlag, Berlin, Heidelberg, 2007. 17

[CNSD93] CRUZ-NEIRA C., SANDIN D. J., DEFANTI T. A.: Surround-screen projection-based virtual reality: the design and implementation of the CAVE. In *ACM Proc. Conf. on Comp. Graphics and Interactive Techniques (SIGGRAPH)* (1993), pp. 135–142. 113

[CSP16] CAMPOSECO F., SATTLER T., POLLEFEYS M.: Minimal solvers for generalized pose and scale estimation from two rays and one point. In *Proc. European Conf. on Computer Vision (ECCV)*, Leibe B., Matas J., Sebe N., Welling M., (Eds.). Springer International Publishing, Cham, 2016, pp. 202–218. 15, 16

[Dav03] DAVISON A. J.: Real-time simultaneous localisation and mapping with a single camera. In *IEEE Proc. Int'l Conf. on Computer Vision (ICCV)* (Oct 2003), pp. 1403–1410 vol.2. 37, 82

[Dav06a] DAVIS T.: *Direct Methods for Sparse Linear Systems*. Society for Industrial and Applied Mathematics, 2006. 43

[Dav06b] DAVIS T.: SuiteSparse. http://faculty.cse.tamu.edu/davis/suitesparse.html, 2006. Accessed: 2017-11-08. 43

[DC00] DRUMMOND T., CIPOLLA R.: Application of lie algebras to visual servoing. *Int'l J. of Computer Vision (IJCV) 37*, 1 (2000), 21–41. 47

[DC02] DRUMMOND T., CIPOLLA R.: Real-time visual tracking of complex structures. *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI) 24*, 7 (Jul 2002), 932–946. 4, 47, 86

[DD95] DEMENTHON D. F., DAVIS L. S.: Model-based object pose in 25 lines of code. *Int'l J. of Computer Vision (IJCV) 15*, 1 (Jun 1995), 123–141. 17

[Dee92] DEERING M.: High resolution virtual reality. In *ACM Proc. Conf. on Comp. Graphics and Interactive Techniques (SIGGRAPH)* (1992), pp. 195–202. 113, 121

[Del12] DELLAERT F.: *Factor graphs and GTSAM: A hands-on introduction*. Tech. Rep. GT-RIM-CP&R-2012-002, Georgia Institute of Technology, 2012. 39, 61, 62

[Der94] DERMANIS A.: The photogrammetric inner constraints. *ISPRS Journal of Photogrammetry and Remote Sensing 49*, 1 (1994), 25 – 39. 50

[DF01] DEVERNAY F., FAUGERAS O.: Straight lines have to be straight. *Machine Vision and Applications 13*, 1 (2001), 14–24. 113, 121

[DGLN04] DAVIS T. A., GILBERT J. R., LARIMORE S. I., NG E. G.: A column approximate minimum degree ordering algorithm. *ACM Trans. Mathematical Software (TOMS) 30*, 3 (Sept. 2004), 353–376. 43

[DHD] DAVIS T., HAGER W., DUFF I.: Suitesparse. http://faculty.cse.tamu.edu/davis/suitesparse.html. Accessed: 2017-07-21. 63

[DK06] DELLAERT F., KAESS M.: Square root SAM: Simultaneous localization and mapping via square root information smoothing. *The International Journal of Robotics Research 25*, 12 (2006), 1181–1203. 37, 43, 69

[DRLR89] DHOME M., RICHETIN M., LAPRESTE J. T., RIVES G.: Determination of the attitude of 3D objects from a single perspective view. *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI) 11*, 12 (Dec 1989), 1265–1278. 16, 23

[Ead] EADE E.:. http://ethaneade.com/lie.pdf. Accessed: 2017-06-20. 56

[ED06] EADE E., DRUMMOND T.: Scalable monocular SLAM. In *IEEE Proc. Conf. on Computer Vision and Pattern Recognition (CVPR)* (June 2006), vol. 1, pp. 469–476. 37, 82

[EE11] ELQURSH A., ELGAMMAL A.: Line-based relative pose estimation. In *IEEE Proc. Conf. on Computer Vision and Pattern Recognition (CVPR)* (June 2011), pp. 3049–3056. 15

[EKR*13] ENGELKE T., KEIL J., ROJTBERG P., WIENTAPPER F., WEBEL S., BOCKHOLT U.: Content first - a concept for industrial augmented reality maintenance applications using mobile devices. In *IEEE Proc. Int'l Symp. on Mixed and Augmented Reality (ISMAR)* (Oct 2013), pp. 251–252. 9, 133

[EKR*15] ENGELKE T., KEIL J., ROJTBERG P., WIENTAPPER F., SCHMITT M., BOCKHOLT U.: Content first: A concept for industrial augmented reality maintenance applications using mobile devices. In *ACM Proc. of the Multimedia Systems Conference (MMSys)* (New York, NY, USA, 2015), MMSys '15, ACM, pp. 105–111. 9, 133

[ENvG07] ESS A., NEUBECK A., VAN GOOL L.: Generalised linear pose estimation. In *Proc. of the British Machine Vision Conf. (BMVC)* (2007), BMVA Press, pp. 22.1–22.10.

doi:10.5244/C.21.22. 14, 17

[ESC14] ENGEL J., SCHÖPS T., CREMERS D.: LSD-SLAM: Large-scale direct monocular SLAM. In *Proc. European Conf. on Computer Vision (ECCV)*, Fleet D., Pajdla T., Schiele B., Tuytelaars T., (Eds.). Springer International Publishing, Cham, 2014, pp. 834–849. 6, 47, 83

[ESN06] ENGELS C., STEWÉNIUS H., NISTÉR D.: Bundle adjustment rules. In *Photogrammetric Computer Vision (PCV)* (2006). 40, 46

[EvdH09] ERIKSSON A., VAN DEN HENGEL A.: Optimization on the manifold of multiple homographies. In *IEEE Proc. Int'l Conf. on Computer Vision Workshops (ICCVW)* (Sept 2009), pp. 242–249. 46

[FALS92] FOYLE D., AHUMADA A., LARIMER J., SWEET B.: Enhanced synthetic vision systems: Human factors research and implications for future systems. *SAE T.: Journal of Aerospace 101* (1992), 1734–1741. 111

[FAR] FARO TECHNOLOGIES INC.:. http://www.faro.com/products/metrology/faroarm/overview. Accessed: 2013-04-13. 114

[FB81] FISCHLER M. A., BOLLES R. C.: Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM 24*, 6 (June 1981), 381–395. 16

[Fia10] FIALA M.: Designing highly reliable fiducial markers. *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI) 32*, 7 (July 2010), 1317–1324. 68

[Fio01] FIORE P. D.: Efficient linear solution of exterior orientation. *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI) 23*, 2 (Feb 2001), 140–148. 16

[FLWW98] FYLSTRA D., LASDON L., WATSON J., WAREN A.: Design and use of the microsoft excel solver. *Interfaces 28*, 5 (1998), 29–55. 45

[FN03] FOXLIN E., NAIMARK L.: VIS-Tracker: A wearable vision-inertial self-tracker. In *IEEE Proc. Virtual Reality (VR)* (2003), pp. 199–206. 96

[Fua99] FUA P.: Using model-driven bundle-adjustment to model heads from raw video sequences. In *IEEE Proc. Int'l Conf. on Computer Vision (ICCV)* (1999), vol. 1, pp. 46–53 vol.1. 44, 46

[Gal05] GALLAGHER A. C.: Using vanishing points to correct camera rotation in images. In *IEEE Proc. Canadian Conf. on Computer and Robot Vision (CRV)* (May 2005), pp. 460–467. 88

[Gan84] GANAPATHY S.: Decomposition of transformation matrices for robot vision. *Pattern Recognition Letters 2*, 6 (1984), 401 – 412. 16, 19

[GFG08] GILSON S. J., FITZGIBBON A. W., GLENNERSTER A.: Spatial calibration of an optical see-through head-mounted display. *Journal of Neuroscience Methods 173*, 1 (2008), 140–146. 113

[GFK14] GABBARD J. L., FITCH G. M., KIM H.: Behind the glass: Driver challenges and opportunities for AR automotive applications. *Proc. of the IEEE 102*, 2 (Feb 2014), 124–136. 7

[GHA03] GAO C., HUA H., AHUJA N.: Easy calibration of a head-mounted projective display for augmented reality systems. In *IEEE Proc. Virtual Reality (VR)* (2003), pp. 53–60. 113

[GHT11] GAUGLITZ S., HÖLLERER T., TURK M.: Evaluation of interest point detectors and feature descriptors for visual tracking. *Int'l J. of Computer Vision (IJCV) 94*, 3 (2011), 335. 4, 37

[GHTC03] GAO X.-S., HOU X.-R., TANG J., CHENG H.-F.: Complete solution classification for the perspective-three-point problem. *IEEE Trans. on Pattern Analysis and Machine Intelligence*

(PAMI) 25, 8 (Aug 2003), 930–943. 16

[GIMS18] GRUBERT J., ITOH Y., MOSER K., SWAN J. E.: A survey of calibration methods for optical see-through head-mounted displays. *IEEE Trans. on Visualization and Computer Graphics (TVCG) 24*, 9 (Sept 2018), 2649–2662. 114

[GJMSMCMJ14] GARRIDO-JURADO S., MUÑOZ-SALINAS R., MADRID-CUEVAS F., MARÍN-JIMÉNEZ M.: Automatic generation and detection of highly reliable fiducial markers under occlusion. *Pattern Recognition 47*, 6 (2014), 2280 – 2292. 70

[GJU96] GRIEWANK A., JUEDES D., UTKE J.: Algorithm 755: ADOL-C: A package for the automatic differentiation of algorithms written in C/C++. *ACM Trans. on Mathematical Software 22*, 2 (June 1996), 131–167. 57

[GKS*10] GRISETTI G., KÜMMERLE R., STACHNISS C., FRESE U., HERTZBERG C.: Hierarchical optimization on manifolds for online 2D and 3D mapping. In *IEEE Proc. Int'l Conf. on Robotics and Automation (ICRA)* (May 2010), pp. 273–278. 39

[GLTH12] GAUGLITZ S., LEE C., TURK M., HÖLLERER T.: Integrating the physical environment into mobile remote collaboration. In *ACM Proc. Int'l Conf. on Human-computer Interaction with Mobile Devices and Services (MobileHCI)* (New York, NY, USA, 2012), MobileHCI, ACM, pp. 241–250. 85

[GNTH14] GAUGLITZ S., NUERNBERGER B., TURK M., HÖLLERER T.: World-stabilized annotations and virtual scene navigation for remote collaboration. In *ACM Proc. Symp. on User Interface Software and Technology (UIST)* (New York, NY, USA, 2014), UIST '14, ACM, pp. 449–459. 85

[Goo17] GOOGLE INC.: ARCore. https://developers.google.com/ar/, 2017. Accessed: 2017-09-14. 2, 85

[GSB*07] GEORGEL P., SCHROEDER P., BENHIMANE S., HINTERSTOISSER S., APPEL M., NAVAB N.: An industrial augmented reality solution for discrepancy check. In *IEEE/ACM Proc. Int'l Symp. on Mixed and Augmented Reality (ISMAR)* (Nov 2007), pp. 111–115. 3

[GSC*07] GOESELE M., SNAVELY N., CURLESS B., HOPPE H., SEITZ S. M.: Multi-view stereo for community photo collections. In *IEEE Proc. Int'l Conf. on Computer Vision (ICCV)* (Oct 2007), pp. 1–8. 37

[GSV*12] GAUGLITZ S., SWEENEY C., VENTURA J., TURK M., HÖLLERER T.: Live tracking and mapping from both general and rotation-only camera motion. In *IEEE Proc. Int'l Symp. on Mixed and Augmented Reality (ISMAR)* (Nov 2012), pp. 13–22. 85

[GTFC12] GEORGE P., THOUVENIN I., FREMONT V., CHERFAOUI V.: DAARIA: Driver assistance by augmented reality for intelligent automobile. In *IEEE Proc. Intelligent Vehicles Symp.* (2012), pp. 1043–1048. 111

[GTKN01] GENC Y., TUCERYAN M., KHAMENE A., NAVAB N.: Optical see-through calibration with vision-based trackers: propagation of projection matrices. In *IEEE/ACM Proc. Int'l Symp. on Augmented Reality (ISAR)* (2001), pp. 147–156. 113, 126

[GTN02] GENC Y., TUCERYAN M., NAVAB N.: Practical solutions for calibration of optical see-through devices. In *IEEE/ACM Proc. Int'l Symp. on Mixed and Augmented Reality (ISMAR)* (2002), pp. 169–175. 126

[Gue07] GUENTER B.: Efficient symbolic differentiation for graphics applications. *ACM Trans. on Graphics 26*, 3 (July 2007). 57

[GVL96] GOLUB G. H., VAN LOAN C. F.: *Matrix computations*. Johns Hopkins studies in the mathematical sciences. The Johns Hopkins University Press, Baltimore, London, 1996. 20

[GW08] GRIEWANK A., WALTHER A.: *Evaluating Derivatives*, second ed. Society for Industrial and Applied Mathematics, 2008. 56

[GWW15] GIEGERICH P., WIENTAPPER F., WUEST H.: Method and apparatus for controlling an image generating device of a head-up display. Patent. WO/2015/044280, 02 04, 2015. 12, 133

[Har97a] HARTLEY R.: In defense of the eight-point algorithm. *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI) 19*, 6 (1997), 580 –593. 40, 91

[Har97b] HARTLEY R. I.: Lines and points in three views and the trifocal tensor. *Int'l J. of Computer Vision (IJCV) 22*, 2 (Mar 1997), 125–140. 15

[HFFR12] HEDBORG J., FORSSEN P.-E., FELSBERG M., RINGABY E.: Rolling shutter bundle adjustment. In *IEEE Proc. Conf. on Computer Vision and Pattern Recognition (CVPR)* (2012), pp. 1434–1441. 116

[HHLM07] HELMKE U., HÜPER K., LEE P. Y., MOORE J.: Essential matrix estimation using gauss-newton iterations on a manifold. *Int'l J. of Computer Vision (IJCV) 74*, 2 (Aug 2007), 117–136. 46

[HHN88] HORN B. K. P., HILDEN H. M., NEGAHDARIPOUR S.: Closed-form solution of absolute orientation using orthonormal matrices. *J. Optical Society of America (JOSA) 5*, 7 (Jul 1988), 1127–1135. 13, 15, 30

[HJEW02] HARRAH S., JONES W., ERICKSON C., WHITE J.: The NASA approach to realize a sensor enhanced-synthetic vision system (SE-SVS) [aircraft displays]. In *IEEE Proc. Digital Avionics Systems Conf.* (2002), vol. 2, pp. 11A4–1–11 vol.2. 111

[HK10] HMAM H., KIM J.: Optimal non-iterative pose estimation via convex relaxation. *Image and Vision Computing 28*, 11 (2010), 1515 – 1523. 17

[HLLPF16] HEE LEE G., LI B., POLLEFEYS M., FRAUNDORFER F.: Minimal solutions for pose estimation of a multi-camera system. In *Proc. Int'l Symp. on Robotics Research (ISRR)*, Inaba M., Corke P., (Eds.). Springer International Publishing, Cham, 2016, pp. 521–538. 16

[HLON91] HARALICK R. M., LEE D., OTTENBURG K., NOLLE M.: Analysis and solutions of the three point perspective pose estimation problem. In *IEEE Proc. Conf. on Computer Vision and Pattern Recognition (CVPR)* (Jun 1991), pp. 592–598. 16

[HN13] HARISH P., NARAYANAN P.: Designing perspectively correct multiplanar displays. *IEEE Trans. on Visualization and Computer Graphics (TVCG) 19*, 3 (2013), 407–419. 114

[Hog14] HOGAN R. J.: Fast reverse-mode automatic differentiation using expression templates in C++. *ACM Trans. on Mathematical Software (TOMS) 40*, 4 (July 2014), 26:1–26:16. 57

[Hor87] HORN B. K. P.: Closed-form solution of absolute orientation using unit quaternions. *J. of the Optical Society of America (JOSA) 4*, 4 (1987), 629–642. 13, 15, 30, 93

[HR11] HESCH J. A., ROUMELIOTIS S. I.: A direct least-squares (DLS) method for pnp. In *IEEE Proc. Int'l Conf. on Computer Vision (ICCV)* (Los Alamitos, CA, USA, 2011), vol. 0, IEEE Computer Society, pp. 383–390. 13, 14, 15, 17, 20, 21, 23, 24, 25, 28, 32, 75, 76

[HS90] HARRIS C., STENNETT C.: RAPID - a video rate object tracker. In *BMVA Proc. of the British Machine Vision Conference (BMVC)* (1990), BMVA Press, pp. 15.1–15.6. doi:10.5244/C.4.15. 4, 86

[HS97a] HARTLEY R. I., STURM P.: Triangulation. *Computer Vision and Image Understanding (CVIU) 68*, 2 (1997), 146–157. 37, 118

[HS97b] HEIKKILA J., SILVEN O.: A four-step camera calibration procedure with implicit image correction. In *IEEE Proc. Conf. on Computer Vision and Pattern Recognition (CVPR)* (1997), pp. 1106–1112. 113

[HZ04] HARTLEY R. I., ZISSERMAN A.: *Multiple View Geometry in Computer Vision*, second ed. Cambridge University Press, ISBN: 0521540518, 2004. 16, 19, 40, 41, 118, 121

[IK14a] ITOH Y., KLINKER G.: Interaction-free calibration for optical see-through head-mounted displays based on 3D eye localization. In *IEEE Proc. Symp. on 3D User Interfaces (3DUI)* (March 2014), pp. 75–82. 114

[IK14b] ITOH Y., KLINKER G.: Performance and sensitivity analysis of INDICA: Interaction-free display calibration for optical see-through head-mounted displays. In *IEEE Proc. Int'l Symp. on Mixed and Augmented Reality (ISMAR)* (Sept 2014), pp. 171–176. 114

[IK15] ITOH Y., KLINKER G.: Light-field correction for spatial calibration of optical see-through head-mounted displays. *IEEE Trans. on Visualization and Computer Graphics (TVCG) 21*, 4 (April 2015), 471–480. 114

[Int17] INTER IKEA SYSTEMS B.V.: IKEA Place App. https://itunes.apple.com/de/app/ikea-place/id1279244498?mt=8, 2017. Accessed: 2018-03-16. 84

[IPSS17] ILA V., POLOK L., SOLONY M., SVOBODA P.: SLAM++ - a highly efficient and temporally scalable incremental SLAM framework. *The Int'l J. of Robotics Research (IJRR) 36*, 2 (2017), 210–230. 43, 47, 83

[JNS*12] JEONG Y., NISTÉR D., STEEDLY D., SZELISKI R., KWEON I. S.: Pushing the envelope of modern methods for bundle adjustment. *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI) 34*, 8 (Aug 2012), 1605–1617. 82

[JNY04] JIANG B., NEUMANN U., YOU S.: A robust hybrid tracking system for outdoor augmented reality. In *IEEE Proc. Virtual Reality (VR)* (2004), p. 3. 96

[JRMF16] JAVORNIK A., ROGERS Y., MOUTINHO A. M., FREEMAN R.: Revealing the shopper experience of using a "magic mirror" augmented reality make-up application. In *ACM Proc. Conf. on Designing Interactive Systems (DIS)* (New York, NY, USA, 2016), DIS '16, ACM, pp. 871–882. 7

[KB99] KATO H., BILLINGHURST M.: Marker tracking and HMD calibration for a video-based augmented reality conferencing system. In *IEEE/ACM Proc. Int'l Workshop on Augmented Reality (IWAR)* (1999), pp. 85–94. 68

[KBB*12] KELLNER F., BOLTE B., BRUDER G., RAUTENBERG U., STEINICKE F., LAPPE M., KOCH R.: Geometric calibration of head-mounted displays and its effects on distance estimation. *IEEE Trans. on Visualization and Computer Graphics (TVCG) 18*, 4 (2012), 589–596. 113

[KBP08] KUKELOVA Z., BUJNAK M., PAJDLA T.: Automatic generator of minimal problem solvers. In *Proc. European Conf. on Computer Vision (ECCV)* (Berlin, Heidelberg, 2008), Forsyth D., Torr P., Zisserman A., (Eds.), Springer Berlin Heidelberg, pp. 302–315. 26

[KF14] KNEIP L., FURGALE P.: OpenGV: A unified and generalized approach to real-time calibrated geometric vision. In *IEEE Proc. Int'l Conf. on Robotics and Automation (ICRA)* (May 2014), pp. 1–8. 26

[KFS13] KNEIP L., FURGALE P., SIEGWART R.: Using multi-camera systems in robotics: Efficient solutions to the NPnP problem. In *IEEE Proc. Int'l Conf. on Robotics and Automation (ICRA)* (May 2013), pp. 3770–3776. 16, 17

[KGS*11] KÜMMERLE R., GRISETTI G., STRASDAT H., KONOLIGE K., BURGARD W.: G2o: A general framework for graph optimization. In *IEEE Proc. Int'l Conf. on Robotics and Automation (ICRA)* (May 2011), pp. 3607–3613. 39, 61, 62

[KH07] KAHL F., HENRION D.: Globally optimal estimates for geometric reconstruction problems. *Int'l J. of Computer Vision (IJCV) 74*, 1 (Aug 2007), 3–15. 17

[KHF16] KUKELOVA Z., HELLER J., FITZGIBBON A.: Efficient intersection of three quadrics and applications in computer vision. In *IEEE Proc. Conf. on Computer Vision and Pattern Recognition (CVPR)* (June 2016), pp. 1799–1808. 16

[KJR*11] KAESS M., JOHANNSSON H., ROBERTS R., ILA V., LEONARD J., DELLAERT F.: iSAM2: Incremental smoothing and mapping with fluid relinearization and incremental variable reordering. In *IEEE Proc. Int'l Conf. on Robotics and Automation (ICRA)* (May 2011), pp. 3281–3288. 39, 43

[KL14] KNEIP L., LI H.: Efficient computation of relative pose for multi-camera systems. In *IEEE Proc. Conf. on Computer Vision and Pattern Recognition (CVPR)* (June 2014), pp. 446–453. 15

[KLS14] KNEIP L., LI H., SEO Y.: UPnP: An optimal O(n) solution to the absolute pose problem with universal applicability. In *Proc. European Conf. on Computer Vision (ECCV)* (Cham, 2014), Fleet D., Pajdla T., Schiele B., Tuytelaars T., (Eds.), Springer International Publishing, pp. 127–142. 13, 14, 15, 17, 20, 21, 22, 23, 24, 26, 28, 32, 33, 34, 75, 76

[KM07] KLEIN G., MURRAY D.: Parallel tracking and mapping for small AR workspaces. In *IEEE/ACM Proc. Int'l Symp. on Mixed and Augmented Reality (ISMAR)* (2007), pp. 1–10. 6, 37, 47, 71, 82, 84, 91

[KM09] KLEIN G., MURRAY D.: Parallel tracking and mapping on a camera phone. In *IEEE Proc. Int'l Symp. on Mixed and Augmented Reality (ISMAR)* (2009), pp. 83–86. 83, 91

[KOE*12] KAHN S., OLBRICH M., ENGELKE T., KEIL J., RIESS P., WEBEL S., GRAF H., BOCKHOLT U., PICINBONO G.: Beyond 3D "as-built" information using mobile ar enhancing the building lifecycle management. In *IEEE Proc. Int'l Conf. on Cyberworlds* (Sept 2012), pp. 29–36. 9

[Kon10] KONOLIGE K.: Sparse sparse bundle adjustment. In *Proceedings of the British Machine Vision Conference* (2010), BMVA Press, pp. 102.1–102.11. doi:10.5244/C.24.102. 39, 43

[KRD08] KAESS M., RANGANATHAN A., DELLAERT F.: iSAM: Incremental smoothing and mapping. *IEEE Trans. on Robotics 24*, 6 (Dec 2008), 1365–1378. 39, 43

[KRW13] KIM H., REITMAYR G., WOO W.: IMAF: in situ indoor modeling and annotation framework on mobile phones. *Personal and Ubiquitous Computing 17*, 3 (Mar 2013), 571–582. 84

[KS07] KLOPSCHITZ M., SCHMALSTIEG D.: Automatic reconstruction of wide-area fiducial marker models. In *IEEE/ACM Proc. Int'l Symp. on Mixed and Augmented Reality (ISMAR)* (Nov 2007), pp. 71–74. 68, 69

[KSR98] KLINKER G., STRICKER D., REINERS D.: The use of reality models in augmented reality applications. In *Proc. of the European Workshop on 3D Structure from Multiple Images of Large-Scale Environments (SMILE)*, Koch R., Van Gool L., (Eds.). Springer Berlin Heidel-

berg, Berlin, Heidelberg, 1998, pp. 275–289. 82

[KSS11] KNEIP L., SCARAMUZZA D., SIEGWART R.: A novel parametrization of the perspective-three-point problem for a direct computation of absolute camera position and orientation. In *IEEE Proc. Conf. on Computer Vision and Pattern Recognition (CVPR)* (June 2011), pp. 2969–2976. 16

[KUY04] KOTAKE D., UCHIYAMA S., YAMAMOTO H.: A marker calibration method utilizing a priori knowledge on marker arrangement. In *IEEE/ACM Proc. Int'l Symp. on Mixed and Augmented Reality (ISMAR)* (Nov 2004), pp. 89–98. 68, 69

[KZB*11] KEIL J., ZÖLLNER M., BECKER M., WIENTAPPER F., ENGELKE T., WUEST H.: The house of olbrich — an augmented reality tour through architectural history. In *IEEE Proc. Int'l Symp. on Mixed and Augmented Reality - Arts, Media, and Humanities (ISMAR-AMH)* (Oct 2011), pp. 15–18. 3, 7, 134

[KZE*13] KEIL J., ZOELLNER M., ENGELKE T., WIENTAPPER F., SCHMITT M.: Controlling and filtering information density with spatial interaction techniques via handheld augmented reality. In *Virtual Augmented and Mixed Reality. Designing and Developing Augmented and Virtual Environments* (Berlin, Heidelberg, 2013), Shumaker R., (Ed.), Springer Berlin Heidelberg, pp. 49–57. 133

[LA05] LOURAKIS M. L. A., ARGYROS A. A.: Is levenberg-marquardt the most efficient optimization algorithm for implementing bundle adjustment? In *IEEE Proc. Int'l Conf. on Computer Vision (ICCV)* (Oct 2005), vol. 2, pp. 1526–1531 Vol. 2. 43

[LA09] LOURAKIS M. A., ARGYROS A.: SBA: A software package for generic sparse bundle adjustment. *ACM Trans. on Mathematical Software (TOMS) 36*, 1 (2009), 1–30. 39, 40, 43, 46, 54, 59, 61, 63, 69, 98

[LBGBD12] LARNAOUT D., BOURGEOIS S., GAY-BELLILE V., DHOME M.: Towards bundle adjustment with GIS constraints for online geo-localization of a vehicle in urban center. In *IEEE Proc. Int'l Conference on 3D Imaging, Modeling, Processing, Visualization Transmission (3DIM-PVT)* (Oct 2012), pp. 348–355. 46

[Lev44] LEVENBERG K.: A method for the solution of certain non-linear problems in least squares. *Quarterly of Applied Mathematics 2*, 2 (1944), 164–168. 39, 41

[LF05] LEPETIT V., FUA P.: Monocular model-based 3D tracking of rigid objects: A survey. *Foundations and Trends in Computer Graphics and Vision 1*, 1 (2005), 1–89. 3

[LGBBD16] LARNAOUT D., GAY-BELLILE V., BOURGEOIS S., DHOME M.: Fast and automatic city-scale environment modelling using hard and/or weak constrained bundle adjustments. *Machine Vision and Applications 27*, 6 (Aug 2016), 943–962. 44, 46

[LHD88] LINNAINMAA S., HARWOOD D., DAVIS L. S.: Pose determination of a three-dimensional object using triangle pairs. *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI) 10*, 5 (Sept 1988), 634–647. 16

[LHhK08] LI H., HARTLEY R., HAK KIM J.: A linear approach to motion estimation using generalized camera models. In *IEEE Proc. Conf. on Computer Vision and Pattern Recognition (CVPR)* (June 2008), pp. 1–8. 15

[LHM00] LU C. P., HAGER G. D., MJOLSNESS E.: Fast and globally convergent pose estimation from video images. *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI) 22*, 6 (Jun 2000), 610–622. 14, 17, 31

[Lhu11] LHUILLIER M.: Fusion of GPS and structure-from-motion using constrained bundle adjustments. In *IEEE Proc. Conf. on Computer Vision and Pattern Recognition (CVPR)* (June 2011), pp. 3025–3032. 44

[Lhu12] LHUILLIER M.: Incremental fusion of structure-from-motion and GPS using constrained bundle adjustments. *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI) 34*, 12 (Dec 2012), 2489–2495. 44

[LL09] LIM H., LEE Y. S.: Real-time single camera SLAM using fiducial markers. In *2009 ICCAS-SICE* (Aug 2009), pp. 177–182. 68, 69

[LLF05] LEPETIT V., LAGGER P., FUA P.: Randomized trees for real-time keypoint recognition. In *IEEE Proc. Conf. on Computer Vision and Pattern Recognition (CVPR)* (2005), pp. 775–781. 82, 94

[LMNF09] LEPETIT V., MORENO-NOGUER F., FUA P.: EPnP: An accurate O(N) solution to the PnP problem. *Int'l J. of Computer Vision (IJCV) 81*, 2 (Feb. 2009), 155–166. 16, 24, 31, 91

[Lou16] LOURAKIS M.: An efficient solution to absolute orientation. In *IEEE Proc. Int'l Conf. on Pattern Recognition (ICPR)* (Dec 2016), pp. 3816–3819. 15

[Low04] LOWE D. G.: Distinctive image features from scale-invariant keypoints. *Int'l J. of Computer Vision (IJCV) 60*, 2 (Nov 2004), 91–110. 83

[LPT13] LIM J. J., PIRSIAVASH H., TORRALBA A.: Parsing IKEA objects: Fine pose estimation. In *IEEE Proc. Int'l Conf. on Computer Vision (ICCV)* (Dec 2013), pp. 2992–2999. 4

[LTVC10] LIU M. Y., TUZEL O., VEERARAGHAVAN A., CHELLAPPA R.: Fast directional chamfer matching. In *IEEE Proc. Conf. on Computer Vision and Pattern Recognition (CVPR)* (June 2010), pp. 1696–1703. 4, 86

[LWJR78] LASDON L. S., WAREN A. D., JAIN A., RATNER M.: Design and testing of a generalized reduced gradient code for nonlinear programming. *ACM Trans. Mathematical Software (TOMS) 4*, 1 (Mar. 1978), 34–50. 45

[LXX12] LI S., XU C., XIE M.: A robust O(n) solution to the perspective-n-point problem. *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI) 34*, 7 (July 2012), 1444–1450. 17

[MA15] MIRALDO P., ARAUJO H.: Direct solution to the minimal generalized pose. *IEEE Trans. on Cybernetics 45*, 3 (March 2015), 404–415. 16

[MAMT15] MUR-ARTAL R., MONTIEL J. M. M., TARÓS J. D.: ORB-SLAM: A versatile and accurate monocular SLAM system. *IEEE Trans. on Robotics 31*, 5 (Oct 2015), 1147–1163. 6, 37, 47, 83

[Mar63] MARQUARDT D. W.: An algorithm for least-squares estimation of nonlinear parameters. *J. of the Society for Industrial and Applied Mathematics 11*, 2 (1963), 431–441. 39, 41

[MGT*01] MCGARRITY E., GENC Y., TUCERYAN M., OWEN C., NAVAB N.: A new system for online quantitative evaluation of optical see-through augmentation. In *IEEE/ACM Proc. Int'l Symp. on Augmented Reality (ISAR)* (2001), pp. 157–166. 113

[Mic16] MICROSOFT CORPORATION: Microsoft Hololens. https://www.microsoft.com/en-us/hololens, 2016. Accessed: 2017-09-14. 2, 85

[MKS01] MA Y., KOŠECKÁ J., SASTRY S.: Optimization criteria and geometric algorithms for motion and structure estimation. *Int'l J. of Computer Vision (IJCV) 44*, 3 (Sep 2001), 219–249. 39

[MLD*06] MOURAGNON E., LHUILLIER M., DHOME M., DEKEYSER F., SAYD P.: Real time localization and 3D reconstruction. In *IEEE Proc. Conf. on Computer Vision and Pattern Recognition (CVPR)* (2006), vol. 1, pp. 363–370. 82

[MMHM14] MARTIN P., MARCHAND E., HOULIER P., MARCHAL I.: Mapping and re-localization for mobile augmented reality. In *IEEE Proc. Int'l Conf. on Image Processing (ICIP)* (Oct 2014), pp. 3352–3356. 84

[MMYC16] MUÑOZ-SALINAS R., MARÍN-JIMÉNEZ M. J., YEGUAS-BOLIVAR E., CARNICER R. M.: Mapping and localization from planar markers. *CoRR abs/1606.00151* (2016). 68, 69, 70

[MP13] MCNAMEE J. M., PAN V.: *Numerical Methods for Roots of Polynomials - Part II*, vol. 16 of *Studies in Computational Mathematics*. Elsevier, 2013. 16

[MR11] MIRZAEI F. M., ROUMELIOTIS S. I.: Globally optimal pose estimation from line correspondences. In *IEEE Proc. Int'l Conf. on Robotics and Automation (ICRA)* (May 2011), pp. 5581–5588. 13, 14, 17, 23

[MSUK14] MIDDELBERG S., SATTLER T., UNTZELMANN O., KOBBELT L.: Scalable 6-DOF localization on mobile devices. In *Proc. European Conf. on Computer Vision (ECCV)* (Cham, 2014), Fleet D., Pajdla T., Schiele B., Tuytelaars T., (Eds.), Springer International Publishing, pp. 268–283. 83

[MUS16] MARCHAND E., UCHIYAMA H., SPINDLER F.: Pose estimation for augmented reality: A hands-on survey. *IEEE Trans. on Visualization and Computer Graphics (TVCG) 22*, 12 (Dec 2016), 2633–2651. 1

[Nak15] NAKANO G.: Globally optimal DLS method for PnP problem with cayley parameterization. In *Proc. of the British Machine Vision Conf. (BMVC)* (September 2015), Xie X., Jones M. W., Tam G. K. L., (Eds.), BMVA Press, pp. 78.1–78.11. 24, 33

[Nau08] NAUMANN U.: Optimal jacobian accumulation is NP-complete. *Mathematical Programming 112*, 2 (2008), 427–441. 57

[Nav04] NAVAB N.: Developing killer apps for industrial augmented reality. *IEEE Computer Graphics and Applications (CGA) 24*, 3 (May 2004), 16–20. 7

[NBB16] NEUNERT M., BLOESCH M., BUCHLI J.: An open source, fiducial based, visual-inertial motion capture system. In *IEEE Proc. Int'l Conf. on Information Fusion (FUSION)* (July 2016), pp. 1523–1530. 68, 69

[NF93] NAVAB N., FAUGERAS O.: Monocular pose determination from lines: critical sets and maximum number of solutions. In *IEEE Proc. Conf. on Computer Vision and Pattern Recognition (CVPR)* (Jun 1993), pp. 254–260. 16

[NGN06] NAJAFI H., GENC Y., NAVAB N.: Fusion of 3D and appearance models for fast object detection and pose estimation. In *Proc. Asian Conf. on Computer Vision (ACCV)* (2006). 95

[Nis04] NISTÉR D.: An efficient solution to the five-point relative pose problem. *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI) 26*, 6 (June 2004), 756–770. 15, 40, 91

[NK06] NÖLLE S., KLINKER G.: Augmented reality as a comparison tool in automotive industry. In *IEEE/ACM Proc. Int'l Symp. on Mixed and Augmented Reality (ISMAR)* (Oct 2006), pp. 249–250. 3

[Nöl06] NÖLLE S.: *Augmented reality als Vergleichswerkzeug am Beispiel der Automobilindustrie*. PhD thesis, Technische Universität München, 2006. 3, 5

[NS06] NISTÉR D., STEWÉNIUS H.: Scalable recognition with a vocabulary tree. In *IEEE Proc. Conf. on Computer Vision and Pattern Recognition (CVPR)* (2006), vol. 2, pp. 2161–2168. 87

[NS07] NISTÉR D., STEWÉNIUS H.: A minimal solution to the generalised 3-point pose problem. *Journal of Mathematical Imaging and Vision 27*, 1 (Jan 2007), 67–79. 16

[NW06] NOCEDAL J., WRIGHT S. J.: *Numerical Optimization*, second ed. Springer, New York, NY, USA, 2006. 38, 44

[OGK*13] OLBRICH M., GRAF H., KAHN S., ENGELKE T., KEIL J., RIESS P., WEBEL S., BOCKHOLT U., PICINBONO G.: Augmented reality supporting user-centric building information management. *The Visual Computer 29*, 10 (Oct 2013), 1093–1105. 9

[OKO06] OLSSON C., KAHL F., OSKARSSON M.: The registration problem revisited: Optimal solutions from points, lines and planes. In *IEEE Proc. Conf. on Computer Vision and Pattern Recognition (CVPR)* (June 2006), vol. 1, pp. 1206–1213. 17

[OKO09] OLSSON C., KAHL F., OSKARSSON M.: Branch-and-bound methods for Euclidean registration problems. *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI) 31*, 5 (May 2009), 783–794. 15, 17

[OLFF06] ÖZUYSAL M., LEPETIT V., FLEURET F., FUA P.: Feature harvesting for tracking-by-detection. In *Proc. European Conf. on Computer Vision (ECCV)* (2006), vol. 3953, pp. 592–605. 82, 94

[Ols11] OLSON E.: Apriltag: A robust and flexible visual fiducial system. In *IEEE Proc. Int'l Conf. on Robotics and Automation (ICRA)* (May 2011), pp. 3400–3407. 68

[OVBS17] ÖZYEŞIL O., VORONINSKI V., BASRI R., SINGER A.: A survey of structure from motion. *Acta Numerica 26* (2017), 305–364. 83

[OZTX04] OWEN C., ZHOU J., TANG A., XIAO F.: Display-relative calibration for optical see-through head-mounted displays. In *IEEE/ACM Proc. Int'l Symp. on Mixed and Augmented Reality (ISMAR)* (2004), pp. 70–78. 7, 113, 114, 119

[PBDM07] PENTENRIEDER K., BADE C., DOIL F., MEIER P.: Augmented reality-based factory planning - an application tailored to industrial needs. In *IEEE/ACM Proc. Int'l Symp. on Mixed and Augmented Reality (ISMAR)* (Nov 2007), pp. 31–42. 3

[PDT*09] PLAVŠIĆ M., DUSCHL M., TÖNNIS M., BUBB H., KLINKER G.: Ergonomic Design and Evaluation of Augmented Reality Based Cautionary Warnings for Driving Assistance in Urban Environments. In *Proceedings of the International Ergonomics Association (IEA)* (2009). 111

[PFC*17] PIRE T., FISCHER T., CASTRO G., DE CRISTÓFORIS P., CIVERA J., BERLLES J.: S-PTAM: Stereo parallel tracking and mapping. *Robotics and Autonomous Systems* (2017), –. 15

[PIS13a] POLOK L., ILA V., SMRZ P.: Cache efficient implementation for block matrix operations. In *ACM Proc. of the High Performance Computing Symposium (HPC)* (San Diego, CA, USA, 2013), HPC '13, Society for Computer Simulation International, pp. 4:1–4:8. 43

[PIS*13b] POLOK L., ILA V., SOLONY M., SMRZ P., ZEMCIK P.: Incremental block cholesky factorization for nonlinear least squares in robotics. In *Proceedings of Robotics: Science and Systems* (Berlin, Germany, June 2013). 43

[Ple03] PLESS R.: Using many cameras as one. In *IEEE Proc. Conf. on Computer Vision and Pattern Recognition (CVPR)* (June 2003), vol. 2, pp. II–587–93 vol.2. 15

[Pon09] PONCE J.: What is a camera? In *IEEE Proc. Conf. on Computer Vision and Pattern Recognition (CVPR)* (June 2009), pp. 1526–1533. 15

[PPS13] PETERSEN N., PAGANI A., STRICKER D.: Real-time modeling and tracking manual workflows from first-person vision. In *IEEE Int'l Symp. on Mixed and Augmented Reality (ISMAR)* (Oct 2013), pp. 117–124. 84

[PS12] PETERSEN N., STRICKER D.: Learning task structure from video examples for workflow tracking and authoring. In *IEEE Proc. Int'l Symp. on Mixed and Augmented Reality (ISMAR)* (Nov 2012), pp. 237–246. 84

[PS15] PETERSEN N., STRICKER D.: Cognitive augmented reality. *Computers & Graphics 53* (2015), 82 – 91. 40 years of Computer Graphics in Darmstadt. 84

[PSI*13] POLOK L., SOLONY M., ILA V., SMRZ P., ZEMCIK P.: Efficient implementation for block matrix operations for nonlinear least squares problems in robotic applications. In *IEEE Proc. Int'l Conf. on Robotics and Automation (ICRA)* (May 2013), pp. 2263–2269. 43

[PSR13] PIRCHHEIM C., SCHMALSTIEG D., REITMAYR G.: Handling pure camera rotation in keyframe-based SLAM. In *IEEE Proc. Int'l Symp. on Mixed and Augmented Reality (ISMAR)* (Oct 2013), pp. 229–238. 85

[PZv17] PŘIBYL B., ZEMČIK P., ČADIK M.: Absolute pose estimation from line correspondences using direct linear transformation. *Computer Vision and Image Understanding (CVIU) 161*, Supplement C (2017), 130 – 144. 16

[QL99] QUAN L., LAN Z.: Linear n-point camera pose determination. *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI) 21*, 8 (Aug 1999), 774–780. 16

[RBS11] RAMALINGAM S., BOUAZIZ S., STURM P.: Pose estimation using both points and lines for geo-localization. In *IEEE Proc. Int'l Conf. on Robotics and Automation (ICRA)* (May 2011), pp. 4716–4723. 16

[RBSB10] RAMALINGAM S., BOUAZIZ S., STURM P., BRAND M.: SKYLINE2GPS: Localization in urban canyons using omni-skylines. In *IEEE/RSJ Proc. Int'l Conf. on Intelligent Robots and Systems (IROS)* (Oct 2010), pp. 3816–3823. 88

[RD06] REITMAYR G., DRUMMOND T. W.: Going out: Robust tracking for outdoor augmented reality. In *IEEE/ACM Proc. Int'l Symp. on Mixed and Augmented Reality (ISMAR)* (2006), pp. 109–118. 96

[RED07] REITMAYR G., EADE E., DRUMMOND T. W.: Semi-automatic annotations in unknown environments. In *IEEE/ACM Proc. Int'l Symp. on Mixed and Augmented Reality (ISMAR)* (Nov 2007), pp. 67–70. 84

[RKL14] ROSEN D. M., KAESS M., LEONARD J. J.: RISE: An incremental trust-region method for robust online sparse least-squares estimation. *IEEE Trans. on Robotics 30*, 5 (Oct 2014), 1091–1108. 43

[RLW*10] REITMAYR G., LANGLOTZ T., WAGNER D., MULLONI A., SCHALL G., SCHMALSTIEG D., PAN Q.: Simultaneous localization and mapping for augmented reality. In *IEEE Proc. Int'l Symp. on Ubiquitous Virtual Reality* (July 2010), pp. 5–8. 84

[RPS*18] RAMBACH J., PAGANI A., SCHNEIDER M., ARTEMENKO O., STRICKER D.: 6DoF object tracking based on 3D scans for augmented reality remote live support. *Computers 7*, 1 (2018). 86

[RTG*14] RAO Q., TROPPER T., GRÜNLER C., HAMMORI M., CHAKRABORTY S.: AR-IVI —implementation of in-vehicle augmented reality. In *IEEE Proc. Int'l Symp. on Mixed and Augmented Reality (ISMAR)* (Sept 2014), pp. 3–8. 6

[RW12] RING W., WIRTH B.: Optimization methods on riemannian manifolds and their application to shape space. *SIAM J. on Optimization 22*, 2 (2012), 596–627. 46

[SBK*16] SAURER O., BAATZ G., KÖSER K., LADICKÝ L., POLLEFEYS M.: Image based geo-localization in the alps. *Int'l J. of Computer Vision (IJCV) 116*, 3 (Feb 2016), 213–225. 87

[SBS07] SCHINDLER G., BROWN M., SZELISKI R.: City-scale location recognition. In *IEEE Proc. Conf. on Computer Vision and Pattern Recognition (CVPR)* (June 2007), pp. 1–7. 87

[SD12] SARKIS M., DIEPOLD K.: Camera-pose estimation via projective newton optimization on the manifold. *IEEE Trans. on Image Processing 21*, 4 (April 2012), 1729–1741. 39, 47

[SFHT14] SWEENEY C., FRAGOSO V., HÖLLERER T., TURK M.: gDLS: A scalable solution to the generalized pose and scale problem. In *Proc. European Conf. on Computer Vision (ECCV)* (Cham, 2014), Fleet D., Pajdla T., Schiele B., Tuytelaars T., (Eds.), Springer International Publishing, pp. 16–31. 13, 14, 15, 17, 20, 21, 22, 23, 26, 28, 32, 34, 75, 76

[SFHT16] SWEENEY C., FRAGOSO V., HÖLLERER T., TURK M.: Large scale SfM with the distributed camera model. In *IEEE Proc. Int'l Conf. on 3D Vision (3DV)* (Oct 2016), pp. 230–238. 17, 20, 21

[SFS*17] SHEA R., FU D., SUN A., CAI C., MA X., FAN X., GONG W., LIU J.: Location-based augmented reality with pervasive smartphone sensors: Inside and beyond pokemon go! *IEEE Access 5* (2017), 9619–9631. 7

[SGBBC16] SALEHI A., GAY-BELLILE V., BOURGEOIS S., CHAUSSE F.: Improving constrained bundle adjustment through semantic scene labeling. In *Proc. European Conf. on Computer Vision (ECCV)*, Hua G., Jégou H., (Eds.). Springer International Publishing, Cham, 2016, pp. 133–142. 44, 46

[SH16] SCHMALSTIEG D., HÖLLERER T.: *Augmented reality: principles and practice*. Addison-Wesley Professional, 2016. 2, 7

[SHT15] SWEENEY C., HÖLLERER T., TURK M.: Theia: A fast and scalable structure-from-motion library. In *ACM Proc. Int'l Conf. on Multimedia (MM)* (New York, NY, USA, 2015), MM '15, ACM, pp. 693–696. 26

[SIY04] SATO T., IKEDA S., YOKOYA N.: Extrinsic camera parameter recovery from multiple image sequences captured by an omni-directional multi-camera system. In *Proc. European Conf. on Computer Vision (ECCV)*, Pajdla T., Matas J., (Eds.), vol. 3022 of *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg, 2004, pp. 326–340. 44

[SK11] SCHWERDTFEGER B., KLINKER G.: *Abschlussbericht: Avilus Tracking Contest 2010, 16.-27. August 2010*. Tech. rep., TU München, 2011. 103, 104

[SKYT02] SATO T., KANBARA M., YOKOYA N., TAKEMURA H.: Dense 3-D reconstruction of an outdoor scene by hundreds-baseline stereo using a hand-held video camera. *Int'l J. of Computer Vision (IJCV) 47* (2002), 119–129. 44

[SL92] SMITH S., LASDON L.: Solving large sparse nonlinear programs using GRG. *ORSA Journal on Computing 4*, 1 (1992), 2–15. 45

[SL04] SKRYPNYK I., LOWE D. G.: Scene modelling, recognition and tracking with invariant image features. In *IEEE/ACM Proc. Int'l Symp. on Mixed and Augmented Reality (ISMAR)* (Nov

2004), pp. 110–119. 84

[SLZ01] SHAN Y., LIU Z., ZHANG Z.: Model-based bundle adjustment with application to face modeling. In *IEEE Proc. Int'l Conf. on Computer Vision (ICCV)* (2001), vol. 2, pp. 644–651 vol.2. 46

[SMD10a] STRASDAT H., MONTIEL J. M. M., DAVISON A. J.: Real-time monocular SLAM: Why filter? In *IEEE Proc. Int'l Conf. on Robotics and Automation (ICRA)* (May 2010), pp. 2657–2664. 6, 37, 69, 83

[SMD10b] STRASDAT H., MONTIEL J. M. M., DAVISON A. J.: Scale drift-aware large scale monocular SLAM. In *Robotics: Science and Systems* (2010), Matsuoka Y., Durrant-Whyte H. F., Neira J., (Eds.), The MIT Press. 39, 54

[SP06] SCHWEIGHOFER G., PINZ A.: Robust pose estimation from a planar target. *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI) 28*, 12 (Dec 2006), 2024–2030. 14

[SP08] SCHWEIGHOFER G., PINZ A.: Globally optimal O(n) solution to the PnP problem for general camera models. In *Proc. of the British Machine Vision Conf.(BMVC)* (2008), BMVA Press, pp. 55.1–55.10. doi:10.5244/C.22.55. 17

[SPF10] STRECHA C., PYLVÄNÄINEN T., FUA P.: Dynamic and scalable large scale image reconstruction. In *IEEE Proc. Conf. on Computer Vision and Pattern Recognition (CVPR)* (June 2010), pp. 406–413. 87

[SRT*11] STURM P., RAMALINGAM S., TARDIF J.-P., GASPARINI S., BARRETO J. A.: Camera models and fundamental concepts used in geometric computer vision. *Foundations and Trends in Computer Graphics and Vision 6*, 1–2 (Jan. 2011), 1–183. 15, 113

[SSS08] SNAVELY N., SEITZ S. M., SZELISKI R.: Modeling the world from internet photo collections. *Int'l J. of Computer Vision (IJCV) 80*, 2 (Nov 2008), 189–210. 82

[SSSZ17] SHI Z., SUN J., SHANG Y., ZHANG X.: Hard constrained sparse bundle adjustment of multi-camera with block matrix. In *Proc. SPIE* (2017), vol. 10458, pp. 10458 – 10458 – 9. 45

[ST94] SHI J., TOMASI C.: Good features to track. In *IEEE Proc. Conf. on Computer Vision and Pattern Recognition (CVPR)* (1994), pp. 593 –600. 91

[ST98] SZELISKI R., TORR P. H. S.: Geometrically constrained structure from motion: Points on planes. In *Proc. European Workshop on 3D Structure from Multiple Images of Large-Scale Environments (SMILE)*, Koch R., Van Gool L., (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 1998, pp. 171–186. 44, 46

[Sut74] SUTHERLAND I. E.: Three-dimensional data input by tablet. *Proceedings of the IEEE 62*, 4 (April 1974), 453–461. 16, 19

[SW16] SEO B.-K., WUEST H.: A direct method for robust model-based 3d object tracking from a monocular rgb image. In *Proc. European Conf. on Computer Vision Workshops (ECCVW)* (Cham, 2016), Hua G., Jégou H., (Eds.), Springer International Publishing, pp. 551–562. 86

[TB15] TAIT M., BILLINGHURST M.: The effect of view independence in a collaborative ar system. *Computer Supported Cooperative Work (CSCW) 24*, 6 (2015), 563–589. 85

[TBP12] TANEJA A., BALLAN L., POLLEFEYS M.: Registration of spherical panoramic images with cadastral 3D models. In *IEEE Proc. Int'l Conf. on 3D Imaging, Modeling, Processing, Visualization Transmission (3DIMPVT)* (Oct 2012), pp. 479–486. 87

[TGBC*11] TAMAAZOUSTI M., GAY-BELLILE V., COLLETTE S., BOURGEOIS S., DHOME M.: Nonlinear refinement of structure from motion reconstruction by taking advantage of a partial

knowledge of the environment. In *IEEE Proc. Conf. on Computer Vision and Pattern Recognition (CVPR)* (2011), pp. 3073–3080. 46, 81, 87

[THPL08] TORII A., HAVLENA M., PAJDLA T., LEIBE B.: Measuring camera translation by the dominant apical angle. In *IEEE Proc. Conf. on Computer Vision and Pattern Recognition (CVPR)* (2008), pp. 1 –7. 91

[TK94] TAYLOR C. J., KRIEGMAN D. J.: *Minimization on the Lie Group SO(3) and Related Manifolds*. Tech. Rep. 9405, Center for Systems Science, Dept. of Electrical Engineering, Yale University, New Haven, CT, April 1994. 46, 54

[TK06] TÖNNIS M., KLINKER G.: Effective control of a car driver's attention for visual and acoustic guidance towards the direction of imminent dangers. In *IEEE/ACM Proc. Int'l Symp. on Mixed and Augmented Reality (ISMAR)* (2006), pp. 13–22. 111

[TLK07] TÖNNIS M., LANGE C., KLINKER G.: Visual longitudinal and lateral driving assistance in the head-up display of cars. In *IEEE/ACM Proc. Int'l Symp. on Mixed and Augmented Reality (ISMAR)* (2007), pp. 91–94. 111

[TMHF00] TRIGGS B., MCLAUCHLAN P. F., HARTLEY R. I., FITZGIBBON A. W.: Bundle adjustment — a modern synthesis. In *Vision Algorithms: Theory and Practice: International Workshop on Vision Algorithms Corfu, Greece, September 21–22, 1999 Proceedings*, Triggs B., Zisserman A., Szeliski R., (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 2000, pp. 298–372. 11, 13, 15, 38, 40, 45, 50, 53, 67, 82

[TN00] TUCERYAN M., NAVAB N.: Single point active alignment method (spaam) for optical see-through HMD calibration for ar. In *IEEE/ACM Proc. Int'l Symp. on Augmented Reality (ISAR)* (2000), pp. 149–158. 7, 113, 126

[Tsa87] TSAI R.: A versatile camera calibration technique for high-accuracy 3D machine vision metrology using off-the-shelf TV cameras and lenses. *IEEE J. of Robotics and Automation 3*, 4 (1987), 323–344. 113, 121

[TSLB05] TÖNNIS M., SANDOR C., LANGE C., BUBB H.: Experimental evaluation of an augmented reality visualization for directing a car driver's attention. In *IEEE/ACM Proc. Int'l Symp. on Mixed and Augmented Reality (ISMAR)* (Washington, DC, USA, 2005), ISMAR '05, IEEE Computer Society, pp. 56–59. 111

[TU 11] TU MÜNCHEN: Ismar tracking-contest 2011. `http://far.in.tum.de/TrackingCompetition/WebHome?sortcol=1&table=3&up=1`, 2011. Accessed: 2017-12-22. 108

[TUI17] TAKETOMI T., UCHIYAMA H., IKEDA S.: Visual SLAM algorithms: a survey from 2010 to 2016. *IPSJ Transactions on Computer Vision and Applications 9*, 1 (Jun 2017), 16. 83

[Ume91] UMEYAMA S.: Least-squares estimation of transformation parameters between two point patterns. *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI) 13*, 4 (1991), 376 –380. 13, 15, 28, 30, 75, 93, 118

[US11] UCHIYAMA H., SAITO H.: Random dot markers. In *IEEE Proc. Virtual Reality Conf. (VR)* (March 2011), pp. 35–38. 68

[UTIdML15] UCHIYAMA H., TAKETOMI T., IKEDA S., D. M. LIMA J. P. S.: [poster] abecedary tracking and mapping: A toolkit for tracking competitions. In *IEEE Proc. Int'l Symp. on Mixed and Augmented Reality (ISMAR)* (Sept 2015), pp. 198–199. 59

[VARS14a] VENTURA J., ARTH C., REITMAYR G., SCHMALSTIEG D.: Global localization from monocular SLAM on a mobile phone. *IEEE Trans. on Visualization and Computer Graphics (TVCG) 20*, 4 (April 2014), 531–539. 83

[VARS14b] VENTURA J., ARTH C., REITMAYR G., SCHMALSTIEG D.: A minimal solution to the generalized pose-and-scale problem. In *IEEE Proc. Conf. on Computer Vision and Pattern Recognition (CVPR)* (Los Alamitos, CA, USA, 2014), vol. 0, IEEE Computer Society, pp. 422–429. 16, 22

[VCZS12] VACA-CASTANO G., ZAMIR A. R., SHAH M.: City scale geo-spatial trajectory estimation of a moving camera. In *IEEE Proc. Conf. on Computer Vision and Pattern Recognition (CVPR)* (June 2012), pp. 1186–1193. 87

[Vel95] VELDHUIZEN T.: Expression templates. *C++ Report 7* (1995), 26–31. 57

[VFMN16] VAKHITOV A., FUNKE J., MORENO-NOGUER F.: Accurate and linear time pose estimation from points and lines. In *Proc. European Conf. on Computer Vision (ECCV)* (Cham, 2016), Leibe B., Matas J., Sebe N., Welling M., (Eds.), Springer International Publishing, pp. 583–599. 24

[VH12] VENTURA J., HÖLLERER T.: Wide-area scene mapping for mobile visual tracking. In *IEEE Proc. Int'l Symp. on Mixed and Augmented Reality (ISMAR)* (Nov 2012), pp. 3–12. 83

[Vic] VICON MOTION SYSTEMS:. http://www.vicon.com/index.html. Accessed: 2013-04-13. 114

[VIK*02] VLAHAKIS V., IOANNIDIS M., KARIGIANNIS J., TSOTROS M., GOUNARIS M., STRICKER D., GLEUE T., DAEHNE P., ALMEIDA L.: Archeoguide: an augmented reality guide for archaeological sites. *IEEE Computer Graphics and Applications 22*, 5 (Sep 2002), 52–60. 3

[vKP10] VAN KREVELEN D., POELMAN R.: A survey of augmented reality technologies, applications and limitations. *International Journal of Virtual Reality (IJVR) 9*, 2 (Nov. 2010), 1–20. 2, 7

[Vol14] VOLKSWAGEN AG: Vw tracking challenge 2014. http://www.tracking-challenge.de/content/tc/content/de/history/challenge_2014/results.html, 2014. Accessed: 2017-12-22. 108

[WAB93] WARE C., ARTHUR K., BOOTH K. S.: Fish tank virtual reality. In *Proc. of the INTERACT and CHI Conf. on Human Factors in Computing Systems* (1993), pp. 37–42. 113

[WACS11] WU C., AGARWAL S., CURLESS B., SEITZ S. M.: Multicore bundle adjustment. In *IEEE Proc. Conf. on Computer Vision and Pattern Recognition (CVPR)* (June 2011), pp. 3057–3064. 39, 43, 61, 62

[WAWB09] WIENTAPPER F., AHRENS K., WUEST H., BOCKHOLT U.: Linear-projection-based classification of human postures in time-of-flight data. In *IEEE Proc. Int'l Conf. on Systems, Man and Cybernetics (SMC)* (Oct 2009), pp. 559–564. 134

[WBF11] WAGNER R., BIRBACH O., FRESE U.: Rapid development of manifold-based graph optimization systems for multi-sensor calibration and SLAM. In *IEEE/RSJ Proc. Int'l Conf. on Intelligent Robots and Systems (IROS)* (Sept 2011), pp. 3305–3312. 39

[WEK*14] WIENTAPPER F., ENGELKE T., KEIL J., WUEST H., MENSIK J.: [demo] user friedly calibration and tracking for optical stereo see-through augmented reality. In *IEEE Proc. Int'l Symp. on Mixed and Augmented Reality (ISMAR)* (Sept 2014), pp. 385–386. 12, 114, 125, 133

[WEW*16] WUEST H., ENGEKLE T., WIENTAPPER F., SCHMITT F., KEIL J.: From CAD to 3D tracking — enhancing & scaling model-based tracking for industrial appliances. In *IEEE Proc. Int'l Symp. on Mixed and Augmented Reality (ISMAR-Adjunct)* (Sept 2016), pp. 346–347. 134

[WF02] WELCH G., FOXLIN E.: Motion tracking: no silver bullet, but a respectable arsenal. *IEEE Computer Graphics and Applications (CGA) 22*, 6 (Nov 2002), 24–38. 2

[WK17] WIENTAPPER F., KUIJPER A.: Unifying algebraic solvers for scaled Euclidean registration from point, line and plane constraints. In *Proc. Asian Conf. on Computer Vision (ACCV)* (2017), pp. 52–66. 11, 15, 27, 29, 133

[WKR07] WILLIAMS B., KLEIN G., REID I.: Real-time SLAM relocalisation. In *IEEE Proc. Int'l Conf. on Computer Vision (ICCV)* (2007), pp. 1–8. 91, 94

[WLS08] WAGNER D., LANGLOTZ T., SCHMALSTIEG D.: Robust and unobtrusive marker tracking on mobile phones. In *IEEE/ACM Proc. Int'l Symp. on Mixed and Augmented Reality (ISMAR)* (Washington, DC, USA, 2008), ISMAR '08, IEEE Computer Society, pp. 121–124. 68

[WPS07] WUEST H., PAGANI A., STRICKER D.: Feature management for efficient camera tracking. In *Proc. Asian Conf. on Computer Vision (ACCV)* (2007), pp. 769–778. 82, 96

[WS07] WUEST H., STRICKER D.: Tracking of industrial objects by using CAD models. *J. of Virtual Reality and Broadcasting (JVRB) 4(2007)*, 1 (2007). 86

[WSFTK18] WIENTAPPER F., SCHMITT M., FRAISSINET-TACHET M., KUIJPER A.: A universal, closed-form approach for absolute pose problems. *Computer Vision and Image Understanding (CVIU)* (2018). 11, 12, 133

[WSHN10] WANG L., SPRINGER M., HEIBEL H., NAVAB N.: Floyd-warshall all-pair shortest path for accurate multi-marker calibration. In *IEEE Proc. Int'l Symp. on Mixed and Augmented Reality (ISMAR)* (Oct 2010), pp. 277–278. 68, 69

[WVS05] WUEST H., VIAL F., STRICKER D.: Adaptive line tracking with multiple hypotheses for augmented reality. In *IEEE/ACM Int'l Symp. on Mixed and Augmented Reality (ISMAR)* (Oct 2005), pp. 62–69. 86, 87

[WWK11a] WIENTAPPER F., WUEST H., KUIJPER A.: Composing the feature map retrieval process for robust and ready-to-use monocular tracking. *Computers & Graphics 35*, 4 (2011), 778 – 788. 12, 39, 61, 83, 133

[WWK11b] WIENTAPPER F., WUEST H., KUIJPER A.: Reconstruction and accurate alignment of feature maps for augmented reality. In *IEEE Proc. of Int'l Conf. on 3D Imaging, Modeling, Processing, Visualization and Transmission (3DIMPVT)* (2011), pp. 140–147. 12, 39, 61, 83, 133

[WWRF13] WIENTAPPER F., WUEST H., ROJTBERG P., FELLNER D.: A camera-based calibration for automotive augmented reality head-up-displays. In *IEEE Proc. Int'l Symp. on Mixed and Augmented Reality (ISMAR)* (Oct 2013), pp. 189–197. 12, 114, 133

[WWS07] WUEST H., WIENTAPPER F., STRICKER D.: Adaptable model-based tracking using analysis-by-synthesis techniques. In *Proc. Int'l Conf. on Computer Analysis of Images and Patterns (CAIP)* (2007), vol. 4673, pp. 20–27. 4, 81, 86, 134

[WWS08] WUEST H., WIENTAPPER F., STRICKER D.: Acquisition of high quality planar patch features. In *Proc. Int'l Symp. on Advances in Visual Computing (ISVC)* (2008), pp. 530–539. 134

[WWZ*15] WANG G., WANG B., ZHONG F., QIN X., CHEN B.: Global optimal searching for textureless 3D object tracking. *The Visual Computer 31*, 6 (Jun 2015), 979–988. 86

[XZCK17] XU C., ZHANG L., CHENG L., KOCH R.: Pose estimation from line correspondences: A complete analysis and a series of solutions. *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI) 39*, 6 (June 2017), 1209–1222. 17, 23

[Yan81] YANNAKAKIS M.: Computing the minimum fill-in is NP-complete. *SIAM Journal on Algebraic Discrete Methods 2*, 1 (1981), 77–79. 43

[YASZ17] YOUNES G., ASMAR D., SHAMMAS E., ZELEK J.: Keyframe-based monocular SLAM: design, survey, and future directions. *Robotics and Autonomous Systems 98*, Supplement C (2017), 67 – 88. 83

[YYBM09] YAMADA T., YAIRI T., BENER S. H., MACHIDA K.: A study on SLAM for indoor blimp with visual markers. In *2009 ICCAS-SICE* (Aug 2009), pp. 647–652. 68, 69

[Zac14] ZACH C.: Robust bundle adjustment revisited. In *Proc. European Conf. on Computer Vision (ECCV)* (Cham, 2014), Fleet D., Pajdla T., Schiele B., Tuytelaars T., (Eds.), Springer International Publishing, pp. 772–787. 41

[ZBO14] ZACUR E., BOSSA M., OLMOS S.: Left-invariant riemannian geodesics on spatial transformation groups. *SIAM J. on Imaging Sciences 7*, 3 (2014), 1503–1557. 56

[Zha97] ZHANG Z.: Parameter estimation techniques: a tutorial with application to conic fitting. *Image and Vision Computing 15*, 1 (1997), 59 – 76. 40

[Zha00] ZHANG Z.: A flexible new technique for camera calibration. *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI) 22*, 11 (2000), 1330–1334. 113

[ZKS*13] ZHENG Y., KUANG Y., SUGIMOTO S., ÅSTRÖM K., OKUTOMI M.: Revisiting the PnP problem: A fast, general and optimal solution. In *IEEE Proc. Int'l Conf. on Computer Vision (ICCV)* (Dec 2013), pp. 2344–2351. 13, 23, 24, 27, 31

[ZS10] ZAMIR A. R., SHAH M.: Accurate image localization based on google maps street view. In *Proc. European Conf. on Computer Vision (ECCV)* (Berlin, Heidelberg, 2010), Daniilidis K., Maragos P., Paragios N., (Eds.), Springer Berlin Heidelberg, pp. 255–268. 87