New Editing Techniques for Video Post-Processing

Dissertation

zur Erlangung des Grades des
Doktors der Ingenieurwissenschaften (Dr.-Ing.)
der Naturwissenschaftlich-Technischen Fakultäten
der Universität des Saarlandes

vorgelegt von

Volker Scholz

Saarbrücken
2007

Abgabe der Dissertation: 5.04.2007
Datum des Kolloquiums: 21.05.2007

Dekan der Naturwissenschaftlich-Technischen Fakultät I:
Prof. Dr. Thomas Herfet

Mitglieder des Prüfungsausschusses:
Vorsitzender: Prof. Dr. Joachim Weickert
1. Gutachter: Prof. Dr. Hans-Peter Seidel
2. Gutachter: Prof. Dr. Marcus Magnor
Akademischer Mitarbeiter: Prof. Dr. Karol Myszkowski

Abstract

This thesis contributes to capturing 3D cloth shape, editing cloth texture and altering object shape and motion in multi-camera and monocular video recordings. We propose a technique to capture cloth shape from a 3D scene flow by determining optical flow in several camera views. Together with a silhouette matching constraint we can track and reconstruct cloth surfaces in long video sequences. In the area of garment motion capture, we present a system to reconstruct time-coherent triangle meshes from multi-view video recordings. Texture mapping of the acquired triangle meshes is used to replace the recorded texture with new cloth patterns. We extend this work to the more challenging single camera view case. Extracting texture deformation and shading effects simultaneously enables us to achieve texture replacement effects for garments in monocular video recordings. Finally, we propose a system for the keyframe editing of video objects. A color-based segmentation algorithm together with automatic video inpainting for filling in missing background texture allows us to edit the shape and motion of 2D video objects. We present examples for altering object trajectories, applying non-rigid deformation and simulating camera motion.

Kurzfassung

In dieser Dissertation stellen wir Beiträge zur 3D-Rekonstruktion von Stoffoberfächen, zum Editieren von Stofftexturen und zum Editieren von Form und Bewegung von Videoobjekten in Multikamera- und Einkamera-Aufnahmen vor. Wir beschreiben eine Methode für die 3D-Rekonstruktion von Stoffoberflächen, die auf der Bestimmung des optischen Fluß in mehreren Kameraansichten basiert. In Kombination mit einem Abgleich der Objektsilhouetten im Video und in der Rekonstruktion erhalten wir Rekonstruktionsergebnisse für längere Videosequenzen. Für die Rekonstruktion von

Kleidungsstücken beschreiben wir ein System, das zeitlich kohärente Dreiecksnetze aus Multikamera-Aufnahmen rekonstruiert. Mittels Texturemapping der erhaltenen Dreiecksnetze wird die Stofftextur in der Aufnahme mit neuen Texturen ersetzt. Wir setzen diese Arbeit fort, indem wir den anspruchsvolleren Fall mit nur einer einzelnen Videokamera betrachten. Um realistische Resultate beim Ersetzen der Textur zu erzielen, werden sowohl Texturdeformationen durch zugrundeliegende Deformation der Oberfläche als auch Beleuchtungseffekte berücksichtigt. Im letzten Teil der Dissertation stellen wir ein System zum Editieren von Videoobjekten mittels Keyframes vor. Dies wird durch eine Kombination eines farbbasierten Segmentierungsalgorithmus mit automatischem Auffüllen des Hintergrunds erreicht, wodurch Form und Bewegung von 2D-Videoobjekten editiert werden können. Wir zeigen Beispiele für editierte Objekttrajektorien, beliebige Deformationen und simulierte Kamerabewegung.

Summary

Today's digital image processing tools have greatly advanced movie editing capabilities. However, considerable, time-consuming manual interaction is still necessary for post-production tasks like rotoscoping, segmentation etc. Replacement of non-rigid objects such as cloth is almost infeasible without automation, due to the high number of degrees of freedom of the surface. For general shape and motion editing of video objects, an easy-to-use interactive system which only requires a moderate amount of user interaction is desirable. This dissertation contributes to capturing cloth shape, editing cloth texture and altering object shape and motion in multi-camera and monocular video recordings.

We propose a technique to capture cloth shape from a 3D scene flow by determining optical flow in several camera views. Together with a silhouette matching constraint we can track and reconstruct cloth surfaces in long video sequences. Reconstructing the

surface is a prerequisite for further editing operations such as texture replacement.

In the area of garment motion capture, we present a system to reconstruct time-coherent triangle meshes from multi-view video recordings. It makes use of a specially designed color pattern which allows a unique identification of color features on the garment across different camera viewpoints. Texture mapping of the acquired triangle meshes is used to replace the recorded texture with new cloth patterns.

We extend this work to the more challenging single camera view case. Simultaneously extracting texture deformation and shading effects enables us to achieve texture replacement effects which are close to reality. We use the same color pattern as in the multi-camera approach. This method enables us to exchange fabric pattern designs worn by actors as a video post-processing step.

Finally, we propose a system for keyframe editing of video objects. A color-based segmentation algorithm together with automatic video inpainting for filling in missing background texture allows us to edit shape and motion of 2D video objects. We present examples for altering object trajectories, applying non-rigid deformation and simulating camera motion. Our vision is that a powerful video post-processing framework gives visual effects artists additional artistic freedom to tell the visual story of a film during editing.

To sum up, the key contributions of this thesis are:

- A method for the 3D tracking of cloth motion by optical flow in a multi-camera setting.
- The first system for multi-camera capture of garment motion that uses a color-coded pattern specially designed for robust observation.
- A video editing system for replacing cloth texture with texture deformation and lighting effects, which makes our color-coded approach useful for single camera recordings.

– The first system for keyframe editing of shape and motion of video objects which combines color-based object segmentation with video inpainting methods.

– A new algorithm for matting of video objects.

– A new, fast video inpainting method for static and moving cameras.

Zusammenfassung

Heutige Softwaretools zum Editieren von Bildern und Video ermöglichen vielfältige Bearbeitungsmöglichkeiten für das Editieren von Filmen. Dennoch sind einige Schritte in der Nachbearbeitung wie Rotoscoping und Segmentierung noch sehr zeitaufwendig und benötigen ein beträchtliches Maß an Benutzerinteraktion. Das Ersetzen von deformierbaren Objekten wie Stoff ist ohne Automatisierung praktisch unmöglich, weil die betreffende Oberfläche viele Freiheitsgrade hat. Will man die Form und Bewegung von Videoobjekten im Allgemeinen editieren, wäre eine Software wünschenswert, die mit wenig Benutzerinteraktion auskommt. In dieser Dissertation stellen wir Beiträge zur 3D-Rekonstruktion von Stoffoberfächen, zum Editieren von Stofftexturen und zum Editieren von Form und Bewegung von Objekten in Multikamera- und Einkamera-Videoaufnahmen vor.

Wir stellen eine Methode für die 3D-Rekonstruktion von Stoffoberflächen vor, die auf der Bestimmung des optischen Fluß in mehreren Kameraansichten basiert. Mittels bekannter Kamerakalibrierung wird daraus das 3D-Bewegungsfeld der Szene abgeleitet. In Kombination mit einem Abgleich der Objektsilhouetten im Video und in der Rekonstruktion erhalten wir Rekonstruktionsergebnisse für längere Videosequenzen. Diese Rekonstruktion ist eine Voraussetzung für spätere Editieroperationen wie das Ersetzen von Texturen.

Für die Rekonstruktion von Kleidungsstücken beschreiben wir ein System, das zeitlich kohärente Dreiecksnetze aus Multikamera-Aufnahmen rekonstruiert. Hierzu wird der

Stoff mit einem speziellen Farbmuster bedruckt, das eine einfache Identifikation von Farbfeatures zwischen acht verschiedenen Kameraansichten erlaubt. Mittels bekannter Kamerakalibrierung wird die Oberfläche für jedes Videobild über Triangulation rekonstruiert. Die Stofftextur in der Aufnahme wird durch texture mapping der erhaltenen Dreiecksnetze mit neuen Texturen ersetzt.

Wir setzen diese Arbeit fort in dem wir den anspruchsvolleren Fall mit einer einzelnen Videokamera betrachten. Um realistische Resultate beim Ersetzen der Textur zu erzielen, werden sowohl Texturdeformationen durch zugrundeliegende Deformation der Oberfläche als auch Beleuchtungseffekte berücksichtigt. Es kommt dasselbe Farbmuster wie im vorherigen Projekt zum Einsatz. Diese System könnte dazu eingesetzt werden, die Kleidungstextur von Schauspielern in der Postproduktion auszutauschen.

Im letzten Teil der Dissertation stellen wir ein System zum Editieren von Videoobjekten über Keyframes vor. Dies wird durch eine Kombination eines farbbasierten Segmentierungsalgorithmus mit automatischem Auffüllen des Hintergrunds erreicht. Dadurch können Form und Bewegung von 2D-Videoobjekten editiert werden. Wir zeigen Beispiele für editierte Objekttrajektorien, beliebige Deformationen und simulierte Kamerabewegung. Unsere Vision ist ein leistungsfähiges Softwaretool, das Künstlern mehr Freiheit in der Postproduktion von Filmen gibt.

Zusammengefasst sind die wesentlichen Beiträge dieser Dissertation:

- Eine Methode für das Tracking von Stoff über den optischen Fluß in einem Multikamera-System.
- Das erste System für die 3D-Rekonstruktion von Kleidungsstücken aus Multikamera-Aufnahmen, das auf einem speziell dafür konstruierten Farbmuster beruht.
- Ein System für das Ersetzen von Stofftexturen in monokularen Videoaufnahmen, das Texturdeformationen und Beleuchtungseffekte berücksichtigt.

– Das erste System zum Editieren von Form und Bewegung von Videoobjekten über Keyframes, das farbbasierte Objektsegmentierung mit Inpainting Methoden für Video kombiniert.

– Ein neuer Algorithmus für das Matting von Videoobjekten.

– Eine neue, effiziente Methode für Video Inpainting bei Aufnahmen mit statischer und bewegter Kamera.

Acknowledgments. First and foremost, I am grateful to Prof. Dr. Magnor for supervising my Ph.D. work. After my first job as a software engineer, I'm grateful for the opportunity to do research at an internationally renowned institute. Marcus was a great advisor motivating us for the major conference deadlines. Special thanks is due to Prof. Dr. Seidel who was my co-advisor after Marcus' move to Braunschweig.

Also, I want to thank my co-workers Timo Stich, Michael Keckeisen, Markus Wacker and Sascha El-Abed for helping me build the software behind the papers. Their work is included in this thesis and the corresponding sections are marked explicitly. Thanks to all members of the former Graphics-Optics-Vision group and the Computer Graphics Lab in Braunschweig for helping with administration issues and proofreading of paper drafts (Lukas Ahrenberg, Ellen Fries, Bastian Goldlücke, Ivo Ihrke, Joseph Klumpp, Andrei Lintu, Christian Linz, Anita Sellent and Timo Stich).

Furthermore, I would like to thank Edda Happ, Lukas Ahrenberg, Julia Luxenburger and Sarah Scherer from the Max-Planck-Institute for acting as models during our video recordings. Our tailor Tanja Frisch did also a great job in manufacturing the custom-designed clothing. Thanks also to Michael Repplinger and Prof. Dr. Slusallek from the Computer Graphics Lab, Saarland University for providing camera equipment for the last project.

Special thanks is due to Oliver Schall for regular discussions. Finally, I am most grateful to my family and Betty for their encouragement and support.

Contents

1

Introduction

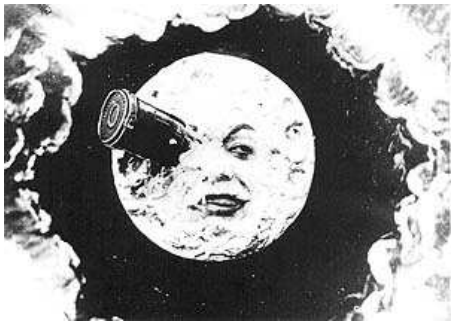


Figure 1.1: Le voyage dans la lune (George Méliès, 1902) [Wik07]. An artificial rocket hits the moon's eye.

From the beginning of cinema, invented by the Lumière brothers in 1895, visual effects as a means to manipulate and transform reality received a lot of attention from the pioneers of cinematography. George Méliès accidentally discovered the stop trick, one of the simplest special effects, in 1896 [Wik07]. It occurs when an object is filmed, then while the camera is off, the object is moved out of sight of the camera. Then the camera is turned back on. When the film is watched it thus seems to the viewer that the object disappears. George Méliès was one of the first filmmakers to use multiple exposures, time-lapse photography, dissolves, and hand-painted color in his films. His science fiction film "Voyage dans la lune" (A Trip to the Moon) from 1902 (Fig. 1.1) is considered to be the first visual effects film.The earliest films showed two important aspects: the amazing realism of the new medium and the ability to make up scenes that were impossible to create any other way.

Figure 1.2: King Kong © 2005 Universal Studios, used with permission.

Until the 1990s, special effects post-production for movies consisted of photo-chemical processing steps, which was very labor-intensive. The recent digitization of photography and video moves beyond the constraints of traditional analog film and poses new challenges for computer scientists. Commercial software packages such as Adobe Photoshop and After Effects [Ado07] now allow everyone to create visual effects on their desktop computer. Stunning compositing effects are now common in today's movie productions (Fig. 1.2). New methods for effective editing of video material are still attractive as this editing work is still considered as an art practiced by a small community. We address this challenge in four projects for editing cloth and general video objects.

Physically-based simulation is the traditional way of generating cloth animations in computer graphics. Recently, cloth capture methods have emerged which build cloth models from video data. Both methods have advantages and disadvantages. Simulation gives the user full control over the result. It can generate high-resolution meshes and commercial software packages are available. For high quality results, long computation times are required. Parameter tweaking and cloth tangling are other common problems. Also, cloth which is resistant to stretch creates instabilities in the simulation resulting from stiff differential equations [HE00]. In contrast, cloth capture does not depend on cloth parameters. Cloth interaction with the human body is implicit and there is no need for elaborate models of the human body. It does not require parameter tweaking and is relatively fast. We will describe a multi-camera approach for 3D reconstruction of real

garments. The obtained models are re-rendered with new textures into the original video frames, opening up new editing possibilities.

Image-based rendering is another alternative for rendering cloth. It operates in the image domain instead of building 3D models. It has several advantages for footage from a single camera. One major advantage is that accurate illumination can be obtained from the video recording. Also, an expensive multi-camera setup which needs time to calibrate is not needed. For this case we propose a system for retexturing cloth in video, based on an image-based technique.

In the last part of this thesis, we extend our scope from cloth editing to editing of general video objects in order to generate a larger variety of visual effects. We present a video editing framework which can be used to alter shape and motion of video objects. By combining color-based video segmentation with automatic inpainting for filling in background texture, general purpose editing of video objects becomes possible. The system is keyframe-based and we show various visual effects such as altering the motion and shape of objects and the simulation of camera motion.

This thesis is organized as follows: first, we will put the work presented here in perspective by giving an overview of related work (Chapter 2). In Chapter 3, an algorithm for tracking cloth motion based on optical flow is described. The subsequent chapter presents a system for capturing cloth motion with a multi-camera system and a custom-designed color pattern (Chapter 4). Chapter 5 deals with our approach for texture replacement of cloth in monocular video. In Chapter 6, we describe our video editing framework for altering object shape and motion. A general discussion concludes the thesis.

2

Related Work

This chapter summarizes work that is related to ours. The first section will describe several cloth capture methods (Section 2.1), which is an alternative to physically-based cloth simulation in computer graphics. Section 2.2 reports on literature relevant for texture replacement in still images and video. Finally, Section 2.3 describes previous research in the editing of video objects.

2.1 Cloth Capture

To motivate this approach, we continue our discussion of cloth simulation versus cloth capture from the introduction. A good survey of the basics of physically-based cloth simulation in computer graphics is given in [HE00] and [Bri03]. Tutorials describing the state-of-the-art in virtual clothing (garment design etc.) can be found in [MTCK$^+$04, MTVTW05]. Starting with the seminal work of Terzopoulos et al. [TPBF87], who presented a model for the animation of deformable surfaces based on continuum mechanics, the literature in this area has grown continuously. In the following years, this physically sound approach was not adopted by computer graphics scientists due to high computational complexity, and particle and mass-spring systems were introduced. To achieve physically plausible results, some parameter tweaking is necessary for these models. As observed by Choi et al. [CK02], the buckling behavior of cloth in simulation strongly depends on initial conditions, which can lead to different

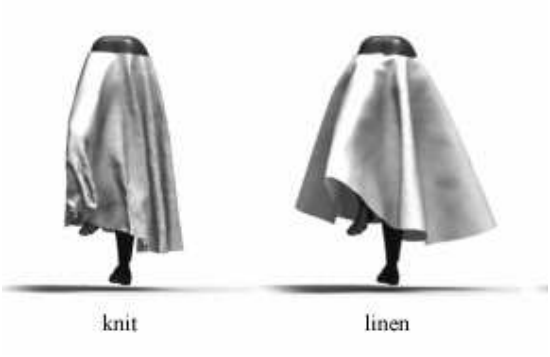


Figure 2.1: Left: reconstruction result from [PH03], © D. Pritchard, used with permission. Right: Results from [BTH+03a] for different materials, © K. Bhat, used with permission.

results for small parameter variations.

After two decades of cloth simulation research in computer graphics there are still major challenges in this area. The modeling of dynamic cloth behavior (hysteresis, damping) is an issue because only static cloth properties are currently measured and modeled. Folds and wrinkles lead to complicated self-collisions, which must be detected and handled robustly by collision detection algorithms. Due to high computational complexity, parallel implementations of these algorithms are considered. Garment design and modeling has been addressed by previous work [MTVTW05], and homogeneous textiles can be simulated by current cloth simulators. Garments composed of different fabrics and with differing seam properties are not accurately modeled by current techniques. We see cloth capture from video as a promising alternative to avoid the complexity of an accurate physically-based simulation. In the following, we describe several related cloth capture methods that build cloth models from video data.

Pritchard and Heidrich [PH03, Pri03] use an image-based approach to cloth motion. They use a calibrated stereo camera pair for shape and obtain the surface parameterization by using SIFT feature matching [Low04] and a region growing technique. Matching to a flat piece of cloth yields texture coordinates (Fig. 2.1 left). The used cloth has a non-repeating line drawing pattern which eases the task of finding feature correspondences.

Motion blur caused by fast motion reduces the accuracy of the matching and the animation lacks frame-to-frame coherence because reconstruction is performed for every frame separately.

Bhat et al. [BTH+03b] estimate the parameters for a cloth simulation by adjusting the simulation results to real world footage. This is an elegant way to avoid parameter tuning by hand. Results for fabrics with different material properties are shown (Fig. 2.1 right). By reducing non-rigid motion to several material parameters, this method is suitable mainly for qualitative reproduction.

Carceroni and Kutulakos [CK01] present a general method for obtaining shape, reflectance and non-rigid motion of a dynamic 3D scene by an algorithm called surfel sampling. Experimental results for complex real scenes (a waving flag, skin, shiny objects) are shown. The reconstructed surfels are quite large which gives a coarse sampling of the surface.

A flow-based tracking method which does not require prior shape models is described by Torresani et al. [TYAB01]. This method produces 3D reconstructions from single-view video by exploiting rank constraints on optical flow. They develop factorization of images of non-rigid scenes, where the non-rigidity is represented as a linear combination of basis shapes. Another example for this factorization approach is the work by Brand [Bra01]. Results for a shoe and a T-shirt tracking sequence are shown. They continued this work in [TB02, TH04]. They obtain reconstructions of non-rigid surfaces by tracking sparse feature sets. While the results are impressive for single camera reconstruction, the tracked features are too sparse for a detailed representation of cloth folds. To obtain accurate cloth models, a dense reconstruction would be necessary.

Lobay and Forsyth [LF04] show that shape-from-texture techniques can be applied to cloth reconstruction. The results are based on still images and a surface model with irradiance maps is reconstructed. Their shape from texture approach derives surface normals from the shape of the individual texture elements which requires a regular tex-

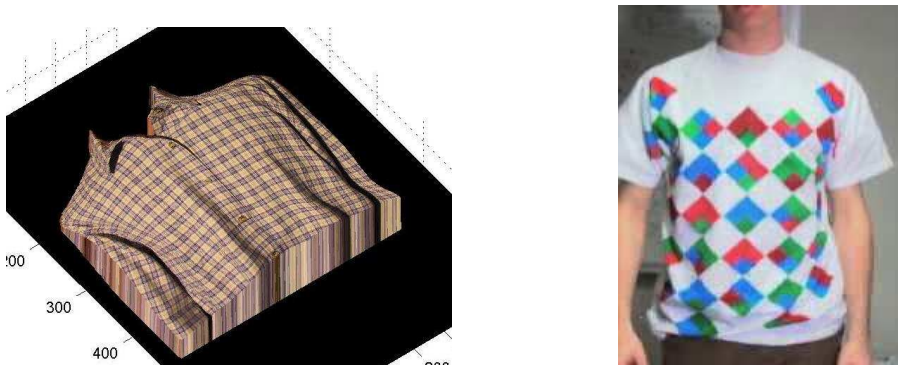


Figure 2.2: Left: reconstruction result from [LF04], © D. Lobay, used with permission. Right: Color-coded T-shirt from [GKB03], © I. Guskov, used with permission.

ture pattern. The results look smooth but lack detail (Fig. 2.2 left).

Han and Zhu [HZ05, HZ07] present a shape-from-shading (SFS) method to determine the 3D geometry of uniformly colored cloth. To improve conventional ill-posed SFS [ZTCS99] they propose a two-layer generative model of cloth folds, which requires some prior knowledge on folds. The upper level consists of a number of folds which generate ridge areas with a dictionary of fold primitives. These primitives are derived in a supervised learning phase based on 3D surfaces acquired through photometric stereo [Woo80]. The lower level consists of the remaining flat areas which are filled in between the folds with a smoothness prior. Compelling cloth surfaces are obtained from still images, but the supervised learning phase requires considerable manual effort.

Hasler et al. [HAR+06, HRA+07] use an analysis-by-synthesis approach to cloth tracking, where they present results for a square piece of cloth. They combine tracking by SIFT features [Low04] with a mass-spring model and optimize the parameters of the cloth simulation. Compelling results are obtained for a simple sequence (lifting the cloth from the floor), but the computational cost of their implementation is prohibitive (20-30 hours on 7 processors). The authors continue this work in [HRS07], where they segment laser scans of people into garment and non-garment regions. They employ a template fitting approach which also estimates garment dimensions. The obtained garment model is the input for a future cloth capture algorithm.

Ebert et al. [ESD03] use color-coded cloth textures for retexturing virtual clothing. Together with range scans of the garment a parameterization of the mesh is obtained. The authors use a color code which has a limited size of codewords so that the pattern is repeated over the whole fabric. In this method the color code is only used for the parameterization of the surface, not for surface reconstruction.

The work by Guskov et al. [Gus02, GKB03] is related to our cloth capture method in Chapter 4. In [Gus02] they introduce an algorithm that tracks checkerboard patterns printed on cloth. The black squares (quads) are tracked independently by temporal prediction of the associated homography. Heuristics to deactivate occluded squares and to activate new appearing squares are applied. Locally, each marker is indistinguishable meaning that correspondence relies on global reasoning and temporal tracking. In [GKB03] this work was extended to a multi-camera setup for 3D reconstruction. Color-coded quad markers allow the identification of position and orientation unambiguously. Results for different surface types, including a T-shirt are presented (Fig. 2.2 right). The used color code has a limited number of codewords, so that a tracking method based on Markov random fields is needed to identify individual quads. The system achieves real-time performance. Tracking performance deteriorates for fast motion and the quads have to be quite large which limits the achievable surface resolution.

Concurrent to our work presented in Chapter 4, White and Forsyth [WLF05, Whi05] propose a multi-camera method based on colored triangle patterns. They use a hierarchical code to disambiguate individual triangles which leads to large holes when the triangle hierarchy is not visible due to occlusion. A cloth simulator is used to fill in the missing data. Cloth silhouettes are used to constrain the surface shape to the visual hull [Lau94]. In [WFV06, WCF07], they continue this work by replacing the hierarchical code by a non-hierarchical code which uses a large number of colors, i.e. they use the whole printer color gamut. Due to camera noise and illumination effects the triangle markers cannot be identified from color alone. A belief propagation algorithm

which uses surface strain as an additional cue is employed. Compelling results for still images are obtained, which contain fine scale folds and wrinkles. We give a summary of the different cloth capture methods in Table 2.2. In comparison to previous work, our cloth capture method presented in Chapter 4 re-uses the idea of using color-coded markers introduced by Guskov et al. and introduces a new color pattern. This pattern allows us to use more markers and to obtain dense 3D models of garments in a circle-like multi-camera setup.

2.2 Texture Replacement

Several authors have worked on texture replacement in still images. Two main effects have to be considered for proper texture replacement: the geometric distortion of the texture due to the surface structure and the lighting effects which are present in the original image. Tsin et al. [TLR01] propose to replace near-regular texture patterns in a plane by learning a statistical texture model and lighting distributions from a sample image. Oh et al. [OCDD01] use texture replacement in their image editing system. Depth information is used to generate foreshortening distortions of the texture, and lighting changes are also extracted. Image Analogy [HJO$^+$01] and Image Quilting [EF01a] show texture transfer effects which preserve local appearance of the texture but do not model texture distortion and lighting effects explicitly. Liu et al. [LLH04] present an approach which builds on user-assisted lattice extraction for near-regular texture (e.g. a brick wall). A PCA analysis of the obtained geometric and lighting deformation fields allows control over texture regularity. Textureshop [FH04a] introduces the idea of using shape-from-shading to recover a rough set of normals for a non-textured surface in the image and using these normals to introduce distortion in the texture synthesis process. User interaction is required to fix normal recovery errors. Zelinka et al. [ZFGH05] continue this work and present a faster system with improved object selection, texture synthesis and

Figure 2.3: Left: statue with new brick texture from [FH04a], © H. Fang, used with permission. Right: T-shirt with superimposed logo from [PLF05b], © J. Pilet, used with permission.

shape-from-shading algorithms.

The major difficulty of replacing texture in video streams is temporal coherence. A single-frame method would inevitably lead to flickering artifacts. Pilet et al. [PLF05b] propose an algorithm for real-time non-rigid surface detection for arbitrary textures which detects a surface by per frame feature matching in conjunction with a deformable mesh model. Being a single frame method, however, temporal coherence is not considered. They extend this work in [PLF05a] by taking surface shading effects into account. Bradley and Roth [BR04] augment cloth and paper with texture and interpolated lighting by using augmented reality square markers. Concurrent to our work, White and Forsyth [WF06] re-texture special clothing with color patterns and natural clothing with a limited number of colors. Their irradiance estimation exploits the property that pixels can be classified into few color classes. Texture replacement for video data maintaining temporal coherence has been attempted only recently by Lin et al. [Lin05, LL06, LL07]. The method is based on user-assisted lattice extraction for near-regular texture on cloth (Fig. 2.4 left). The lattice structure is modeled by a Markov Random Field and tracked with an affine Lucas-Kanade algorithm [LK81a]. Temporal coherence of the texture deformation and shading maps is achieved by spatio-temporal smoothing as a post-

Figure 2.4: Left: texture replacement results from [Lin05], © W. Lin, used with permission. Right: retextured face in video [FH06], © H. Fang, used with permission.

processing step.

Fang et al. [FH06] propose a video editing system called RotoTexture, which can synthesize texture on nearly diffuse surfaces such as skin and a T-shirt (Fig. 2.4 right). It is an extension of the Textureshop paper [FH04a] to video. Additionally to the shape-from-shading method, the system uses a spring model to model the surface as it is deforming according to the recovered normal field. Optical flow and feature tracking are used to obtain a temporally coherent texture mapping result. Temporal smoothing is applied as a post-processing step. This is actually the first system that can re-texture homogeneous surfaces in video. The results show texture swimming artifacts, this is due to inaccuracies of feature point tracking.

To obtain realistic texture replacement results, the lighting conditions in the input data have to be considered. Determining reflectance and shading at each scene point is also referred to as the *intrinsic image* problem. The goal is to decompose an input image into two images, one containing the shading information and the other the reflectance information. Oh et al. [OCDD01] make the simplifying assumption that large-scale luminance variations are due to the lighting, while small-scale variations are due to texture. The texture features are blurred with an adaptive bilateral filter. A texture image with uniform lighting is obtained by dividing the initial image by the blurred image. The computer vision literature contains several algorithms to solve the general intrinsic image problem. Tappen et al. [TFA05] use machine learning for classifying image

pixels while Finlayson et al. [FDL04] rely on a projection of color onto gray images minimizing image entropy. In [FDB92], Funt et al. recover the shading field by removing reflectance changes in the gradient image. Integrating the manipulated gradient field by solving a Poisson equation leads to the shading image. Table 2.1 summarizes the different texture replacement methods. In comparison to previous work, our method presented in Chapter 5 uses a color-coded pattern to obtain dense texture replacement results with correct lighting and shading effects. Bradley et al. [BR04] also work with markers but obtain sparse marker coordinates which are interpolated. Other methods are targeted towards textured [PLF05b, LL06] or diffuse surfaces [FH06], which is a more general problem and works well in many cases, but not always high-quality results can be obtained.

Method	Texture	Features	Details	Restrictions
[PLF05b, PLF05a]	general	SIFT	deformable model, lighting	stiffness of the model
[Lin05, LL06, LL07]	near-regular	optical flow	retexturing, lighting	tracking robustness
[WF06]	few colors	color	retexturing, lighting	color restriction
[BR04]	markers	AR markers	retexturing, lighting	coarse resolution
[SM06b] (ours)	markers	color dots	retexturing, lighting	discontinuities
[FH06]	diffuse surface	shading	texture synthesis, optical flow	texture swimming artifacts

Table 2.1: Comparison of methods for texture replacement in monocular video.

2.3 Video Object Editing

To allow editing of video objects, the video has first to be segmented into objects. An automatic approach for *video segmentation* is described by Wang et al. [WTXC04] (Fig. 2.5 left). The mean-shift image segmentation method [CM02] is extended to video and applied to pixels in $6D$ (x,y,t,r,g,b) space. Adaptive anisotropic kernels allow better feature extraction than previous isotropic kernels. The algorithm finds homogeneous regions in video which are consistent with human visual saliency. Grouping the regions that belong to one video object is still an interactive step and not automated. The running time is on the order of several hours.

Figure 2.5: Left: mean-shift segmentation from [WTXC04]. Right: Cutout ballet dancer from [WBC+05]. © J. Wang, used with permission.

Faster, more interactive systems have been recently proposed. Wang et al. [WBC+05] compute a pre-segmentation with a 2D mean-shift algorithm (Fig. 2.5 right). A graph-cut based image segmentation algorithm is extended to video, with running times of a few seconds. This work is based on image segmentation with graph cuts, introduced for greyscale images by Boykov and Jolly [BJ01]. Later, this method was extended to color images by Rother et al. [RKB04]. Blake et al. [BRB+04] provide a theoretical analysis of the algorithm and propose a method to learn parameters which they evaluate with ground truth segmentation data.

Li et al. [LSS05] apply a 3D graph-cut segmentation algorithm to the spatio-temporal video cube. The result is refined with a 2D graph cut algorithm in localized windows around the object's border. Our system presented in Chapter 6 is inspired by this work and proposes several important extensions (steerable presegmentation and a new interactive boundary editing tool). In contrast, an automatic learning-based method by Criminisi et al. [CCBK06] which uses color and motion cues produces good quality results but requires ground truth segmentation masks, which is not practical for our primary goal, a general-purpose editing tool.

Editing the segmented objects can produce holes in the background. In the following, we review the literature on hole filling in images and video. There exists a large

body of work on *texture synthesis* [WL00, EL99, EF01b] and the closely related *image inpainting* problem [BSCB00]. Image inpainting propagates linear image structures (called isophotes) from a hole's circumference into the hole region by using a PDE-based method. It works well for small, smooth and low-textured regions. For larger missing regions or textured regions it may generate blurring artifacts. Exemplar-based texture synthesis fills unknown regions by copying image patches from the hole border. It is aimed at reproducing textural patterns but has problems with macrostructure in the image. Approaches that generate texture on a per-pixel basis [WL00, EL99] are computationally more expensive than patch-based methods [EF01b]. Criminisi et al. [CPT03] show how continuations of strong edges can be propagated inwards, which preserves simple structures. Our algorithm presented in Chapter 6 builds on the work by Criminisi et al. We extend it to video and contribute two valuable improvements: weighted matching and patch blending. We also focus on a time-efficient implementation. Compared to the global optimization approach proposed by Wexler et al. [WSI04, WSI07], our method is significantly faster. We can also handle fast camera motion with our method, while Patwardhan et al. [PSB05] (Fig. 2.6) and [WSI04] present results for a static camera only. Concurrent to our work, Patwardhan et al. [PSB07] later presented inpainting results for sequences with moderate camera motion and for moving objects that slightly change size.

To provide an easy-to-use user interface for object editing, we use a keyframe-based editing framework. *Keyframe animation* is a well-known technique from production systems like Autodesk's Maya [Aut07]. It offers the animator excellent control over the final motion and is used in high end production. For our general purpose video editor it is the ideal tool to specify object motion without having to consider motion laws from physics. Much work has been done on interpolating keyframes. Relevant to our approach, Kochanek et al. introduced interpolating splines with local tension, continuity and bias control [KB84]. This technique gives the user much control over the

Figure 2.6: Left: Original video frame from [PSB05]. Right: inpainting result (person was removed). © K. Patwardhan, G. Sapiro, M. Bertalmio, used with permission.

final result. To reduce the number of parameters we use cubic spline interpolation.

To composite the edited object back into the video, a *matting* algorithm is needed to compute alpha masks. The goal is here to extract a foreground element from a background image by estimating an opacity α for each pixel of the foreground. The pixel color C is modelled as a linear combination of a foreground color F and a background color B (compositing equation):

$$C = \alpha \cdot F + (1 - \alpha) \cdot B \tag{2.1}$$

Determining α, F and B for each pixel is the so-called matting problem. The image matting problem has been studied intensively, we just mention the most relevant publications. Chuang et al. [CCSS01] introduce Bayesian matting, a Bayesian framework for solving the matting problem. They model foreground and background color models with spatially varying sets of Gaussians and use a maximum-likelihood criterion to estimate foreground, background and opacity simultaneously. The user is required to supply a trimap that partitions the image into three regions: foreground, background and an unknown region. Shum et al. [SSY+04] propose a modification of the matting equations which they call coherence matting. By using a coherence prior for alpha, they

obtain more robust results for the case when foreground and background colors are similar. Sun et al. [SJTS04] introduce Poisson Matting, where they cast the matting problem as solving a Poisson equation with the matte gradient field. By interactively manipulating the matte gradient field using a number of filtering tools, the user can improve the matting results locally. An inherent limitation is the assumption that the foreground and background intensity varies smoothly, i.e. matting of textured objects is an issue. Recent methods by Wang and Cohen [WC05] and Levin et al. [LLW06] explore the case of limited user input, i.e. instead of specifying a full trimap the user only has to mark foreground and background regions with a few paint strokes. This is advantageous for images with large semi-transparent regions (e.g. a spider web), since it is difficult to create a trimap manually in this case. For single camera video matting, various techniques exist [CAC+02, CCBK06]. In this case, temporal coherence has to be considered to achieve convincing results. We choose the robust border matting method for images by Rother et al. [RKB04] and propose two modifications (a different color model and thin-plate spline regularization).

Related to our work is the motion magnification approach proposed by Liu et al. [LTF+05]. Their goal is to amplify subtle motions in a video sequence. The input is a sequence of images from a stationary camera, the output is a re-rendered video sequence with magnified motions of selected layers. Our system, in contrast, solves the more general problem of editing video object shape and motion in various ways while the camera is also moving.

Method	Cameras	Video	Features	Pattern	Details	Restrictions
[BTH+03b]	1	x	edges	-	parameter estimation	cloth models insufficient
[TYAB01, TB02]	1	x	corners	-	flow-based tracking	sparse reconstruction
[TH04]	1	x	corners	-	learning-based	sparse reconstruction
[LF04]	1	-	SIFT	periodic texture	shape from texture, shading	oversmoothed surface
[Gus02]	1	x	markers	checkerboard	homography tracking	coarse resolution
[HZ05, HZ07]	1	-	shading	-	two-layer SFS model	supervised learning phase
[PH03, Pri03]	2	x	depth, SIFT	-	stereo, SIFT matching	temporal coherence
[CK01]	7	x	raw	-	surfel sampling	coarse mesh
[GKB03]	4	x	markers	color quads	homography tracking	coarse mesh
[SM04] (ours)	3	x	silhouettes, raw	-	flow-based tracking	synthetic data
[SSK+05] (ours)	8	x	markers	color dots	triangulation	occluded regions
[WLF05, Whi05]	4	x	markers	color triangles	triangulation	large holes
[WFV06]	6-10	-	markers	color triangles	new triangle pattern	still image
[HAR+06]	8	x	silhouettes, SIFT	-	physical model-based tracking	simple cloth, resolution

Table 2.2: Comparison of cloth capture methods. The columns describe the number of employed cameras, video or still image, used features, special pattern or natural texture, algorithm details and restrictions.

3

Cloth Motion from Optical Flow

3.1 Introduction

This chapter presents an algorithm for capturing the motion of deformable surfaces, in particular textured cloth. In a calibrated multi-camera setup, the optical flow between consecutive video frames is determined and 3D scene flow is computed. We use a deformable surface model with constraints for vertex distances and curvature to increase the robustness of the optical flow measurements. Tracking errors in long video sequences are corrected by a silhouette matching procedure. We present results for synthetic cloth simulations and discuss how they can be extended to real-world footage. The following chapter is structured as follows. In Section 3.2, we give a short overview of our algorithm. Sections 3.3–3.5 describe the components of the algorithm in detail. Results are presented in Section 3.6, before we make concluding remarks in Section 3.7.

3.2 Algorithm Overview

We propose an approach using *optical flow* as the main component in our reconstruction algorithm. A prerequisite for the use of optical flow is a richly detailed cloth texture. If the initial position of the cloth is known, vertex motion can be tracked from frame to frame using optical flow information. Given a high frame rate, optical flow between consecutive frames is suitable to track rapid motion. Optical flow is a well-known prob-

lem and the quality of the available algorithms is sufficient for practical applications. Previous work by DeCarlo and Metaxas [DM96] applied optical flow tracking to the problem of determining human face shape and motion from a single camera. They combine this with edge information and use a deformable model. We extend this approach to the challenging case of cloth tracking and use a multi-camera setup to obtain 3D flow information.

Optical flow is not well-defined in poorly textured regions so we have to interpolate over these regions. We employ a deformable cloth model for this purpose. The model also makes the algorithm more robust against optical flow errors. Adding the interframe flow vectors over a long frame sequence is not feasible because flow errors accumulate and tracking errors are introduced. As a consequence, our model would drift away from the image data. In order to address this problem, the cloth silhouette is determined in the video frames and the boundary vertices of the model are matched to the silhouette.

3.3 Optical Flow

Optical flow is the apparent motion of brightness patterns between two frames of an image sequence. In [BFB94] and [GMN+98] several optical flow algorithms are evaluated. The method by Lucas and Kanade [LK81a] shows the best accuracy and noise tolerance. It was originally a stereo matching technique but is now mainly used for optical flow. We give a short description of the algorithm. The basic assumptions of the algorithm are:

- the brightness of the image pixels remains constant between successive video frames.
- the motion can be described by a pure translation in the image plane.

This can be summarized as

$$I(\mathbf{x},t) = I(\mathbf{x}+\mathbf{u},t+dt) \tag{3.1}$$

where I denotes image brightness, $\mathbf{x}$ the pixel location, $\mathbf{u}$ the pixel translation and t the time. The first order Taylor series expansion of image intensity I is given by

$$I(\mathbf{x}+\mathbf{u},t+dt) = I(\mathbf{x},t) + \nabla I \cdot \mathbf{u} + \frac{\partial I}{\partial t} \cdot dt \tag{3.2}$$

where ∇I denotes the spatial image gradient and $\frac{\partial I}{\partial t}$ is the temporal derivative of intensity. Plugging Eq. (3.1) into (3.2) leads to the optical flow constraint equation:

$$\nabla I \cdot \mathbf{u} + \frac{\partial I}{\partial t} \cdot dt = 0 \tag{3.3}$$

The Lucas-Kanade algorithm minimizes the left hand side of Eq. (3.3) in a window W around a pixel with respect to $\mathbf{u}$:

$$E_{min} = \min_{\mathbf{u}} \sum_{\mathbf{x} \in W} w^2(\mathbf{x}) (\nabla I \cdot \mathbf{u} + \frac{\partial I}{\partial t} \cdot dt)^2 \tag{3.4}$$

$w : W \rightarrow \mathbb{R}$ denotes a Gaussian kernel function which gives more weight to pixels near the window center. This function is minimized with the Newton method. The summation window W increases the robustness of the method, as the pixel translation $\mathbf{u}$ is assumed to be constant inside the window. The algorithm can find pixel displacements $\mathbf{u}$ in the subpixel range. Larger pixel displacements are handled by a multiresolution scheme on a Gaussian image pyramid which consists of four levels in our case. Optical flow is determined on the coarsest level and propagated as an initial solution to the next, more detailed level. This multiresolution strategy is also used to avoid local minima of the energy function. The implementation details can be found in [Bou00] and we use

their implementation. The optimization method finds suitable solutions as long as the two considered frames are similar enough so that a local minimum is sufficient.
In order to make the flow computation more reliable, we compute the flow in a projected 3D rectangular patch around the vertex positions of our cloth model and apply a median filter to the flow vectors in the patch for outlier removal (we choose the vector with minimum distance from the remaining vectors). The size of the window W in Eq. 3.4 is 5x5 pixels. Larger windows increase the robustness of the Lucas-Kanade algorithm but can also lead to oversmoothing of the flow vector field.
Two-dimensional optical flow is a projection of a three-dimensional range or scene flow to the image plane [VBR+99]. If the camera calibration and the initial vertex positions of the surface are known, the scene flow can be determined from several camera views by a method similar to triangulation. The vertex motion is optimized together with a deformable model which we describe in the next section.

3.4 Deformable Model

Deformable models have successfully been applied to motion tracking problems [PH91, TM91]. The goal is to provide additional constraints which make the motion estimation more robust. In [PH91], the number of degrees of freedom of a finite element method (FEM) model is reduced by an analysis of the vibration modes. This approach is suitable for simple shapes and difficult to apply to cloth with its fine scale, complex folds. [TM91] introduces superquadrics for deformable surfaces, a model which is targeted towards closed surfaces. A good introduction to physics-based deformable models is [Met96]. Our model minimizes deformation energy per frame while in previous work the temporal dynamics are also considered. This introduces model parameters for mass, damping etc. which have to be estimated. Our model only contains parameters for deformation strain.

Our cloth model consists of a rectangular grid of vertices. Cloth deformation can be described in terms of three basic deformations: stretching, bending and shearing [HE00]. Stretching is almost negligible for non-elastic materials and can be used as a constraint for the vertex positions. We propose the following energy function which penalizes compression and stretching of horizontally or vertically adjacent vertices $\mathbf{p}_i$, $\mathbf{p}_j$, where d_0 denotes the initial vertex distance (Fig. 3.1):

$$E_{stretch} = \sum_{i,j} \left(\frac{\|\mathbf{p}_i - \mathbf{p}_j\| - d_0}{d_0}\right)^2 \tag{3.5}$$

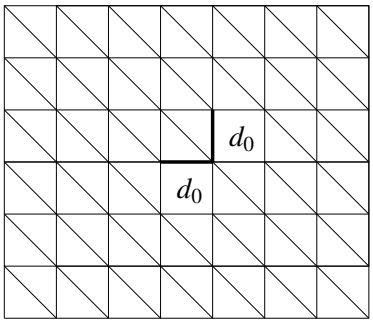


Figure 3.1: Uniform triangulation with distance constraint for adjacent vertices.

The diagonal mesh edges are not considered here as this would constrain shearing deformations. It is also reasonable to assume a smooth cloth surface. We use a discrete version of thin plate spline energy [MS97]

$$E_{curv} = \sum_{t_a,t_b} l_e \cdot (\mathbf{n}_{t_a} - \mathbf{n}_{t_b})^2 \tag{3.6}$$

where $\mathbf{n}_{t_a}$ and $\mathbf{n}_{t_b}$ are the normalized normals of adjacent triangles t_a and t_b and l_e is the length of their common edge (Fig. 3.2). In our implementation, we omit normalization as the analytical gradient would get too complex. This approximation assumes constant triangle areas, i.e. small shear but yields pleasing results. The summation in Eq. 3.4 is done over all triangle pairs with an adjacent edge. $E_{stretch}$ and E_{curv} are similar to the energy terms used in cloth simulation [HE00].

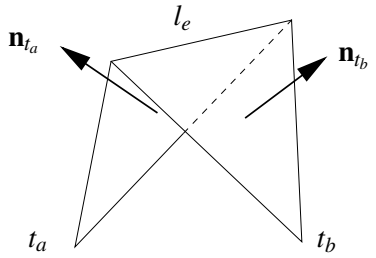


Figure 3.2: Bending constraint for adjacent triangles

The vertex error of the model at time t with respect to optical flow to the next video frame at time $t+1$ is given as

$$\begin{aligned} \hat{x}_i &= \frac{\mathbf{m}_0^c \cdot \mathbf{p}_i}{\mathbf{m}_3^c \cdot \mathbf{p}_i} \\ \hat{y}_i &= \frac{\mathbf{m}_1^c \cdot \mathbf{p}_i}{\mathbf{m}_3^c \cdot \mathbf{p}_i} \\ \varepsilon_i^c &= \sqrt{(\hat{x}_i - (x_i + u_i))^2 + (\hat{y}_i - (y_i + v_i))^2} \\ E_{flow} &= \sum_{c \in C} \sum_i v_i^c \varepsilon_i^c \end{aligned} \tag{3.7}$$

where $\mathbf{m}_0^c - \mathbf{m}_3^c$ are the rows of the 4x4 OpenGL projection matrix $\mathscr{M}^c$ of camera $c \in C$. $\hat{x}_i$ and $\hat{y}_i$ are the projected vertex coordinates. u_i and v_i are the components of the optical flow vector and ε_i^c the projection error of vertex i in camera view c. The inner sum is computed for all visible vertices in a camera view, indicated by a visibility variable $v_i^c \in \{0,1\}$. Visibility is determined with an OpenGL depth buffer test.

All energy terms are combined with weighting factors into one energy function

$$E = E_{flow} + \lambda \cdot E_{stretch} + \mu \cdot E_{curv} \tag{3.8}$$

and optimized with the Polak-Ribière conjugate gradient method [PFTV92]. The energy function gradient can be computed analytically. If the interframe differences are small (high frame rate), conjugate gradient minimization is suitable because we are already near the optimum.

Figure 3.3: Input images from two of three camera views. The third camera is located in the middle of the two views.

3.5 Silhouette Matching

The optical flow errors accumulate over longer frame sequences and the 3D model drifts away from the video frames. To correct for this, we determine the cloth silhouettes in the input images by a border following algorithm [SA85]. With our synthetic test data, the contour can be directly determined from the input images. Real video data would require background subtraction [FP02] as a preprocessing step. For every vertex on the mesh boundary of our model, the nearest contour point is determined. Its position corresponds to the new vertex position $(x_i + u_i, y_i + v_i)$ in Eq. (3.7), i.e. the contour generates *flow* vectors for the boundary vertices. The inner vertices do not contribute to E_{flow}, they are only constrained by $E_{stretch}$ and E_{curv}. The mesh is adjusted by optimizing the energy function from Eq. (3.8). A limitation of the contour matching procedure is that the boundary vertices have to stay on the cloth silhouette during the whole video sequence.

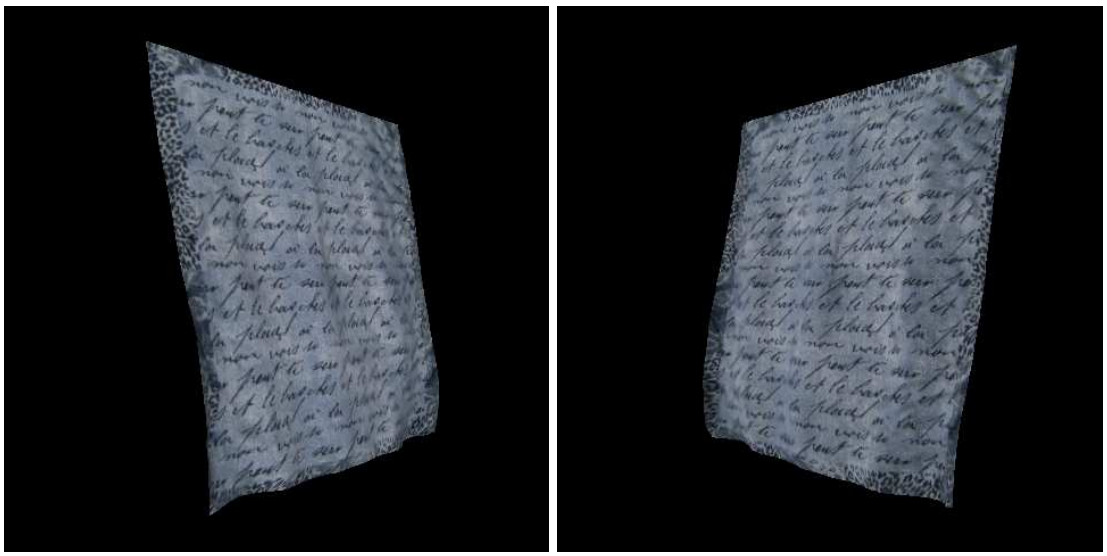


Figure 3.4: Reconstruction from novel viewpoints.

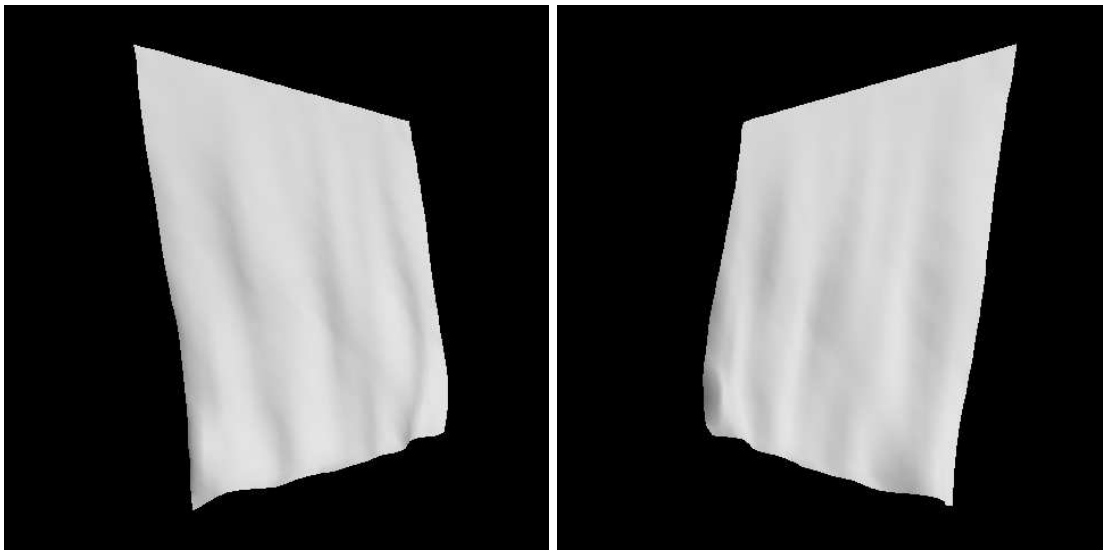


Figure 3.5: Shaded views showing the cloth folds which are found by our method.

3.6 Results

Experiments with our video cameras with a resolution of 320x240 pixels at that time showed that a higher image resolution is necessary for an accurate surface reconstruction from optical flow. For this reason, we use synthetic data generated with a cloth simulator [Ebe03] to test our algorithm.

The sequence consists of 300 frames, recorded by three camera views with a resolution

of 640x480 pixels. Fig. 3.3 shows the test sequence, a piece of cloth flapping in a breeze. The cloth texture was acquired with a still image digital camera and mapped on the cloth mesh. A directional light source was added to the scene. The resulting shading effects are challenging for the optical flow algorithm because the brightness constancy constraint is violated.

The triangle mesh used for reconstruction has a resolution of $v = 33x33$ vertices, i.e. the optimization problem has $n = 3v = 3267$ variables. The parameters λ and μ in Eq. (3.8) are chosen empirically. We choose $\lambda = 1.0$ and $\mu = 0.1$ for our experiments. These values are not fine-tuned but sufficient for a pleasing result. The parameter μ is adjusted so that the bending deformation of the cloth is preserved but temporal noise is removed. The average computation time for one video frame of the sequence is 53 seconds on a Pentium IV 2.4 GHz. The different stages of the algorithm have the following average time requirements:

- optical flow computation: 18 s
- vertex flow optimization: 17 s
- contour matching: 5 s
- correction step optimization: 13 s

Fig. 3.4 shows the surface rendered from novel viewpoints and Fig. 3.5 the reconstructed surface. In Fig. 3.8 and 3.9 the tracking accuracy of the algorithm over the whole sequence is depicted. The object silhouettes are preserved by the contour matching algorithm. In Fig. 3.6, a reconstructed frame and the difference to the corresponding input image are shown. The difference image shows a pixel displacement in the range of 2-3 pixels between reconstruction and input image. The error is concentrated near the image gradients.

In Fig. 3.7 the root mean squared error (RMSE) between the input images $I_i =$

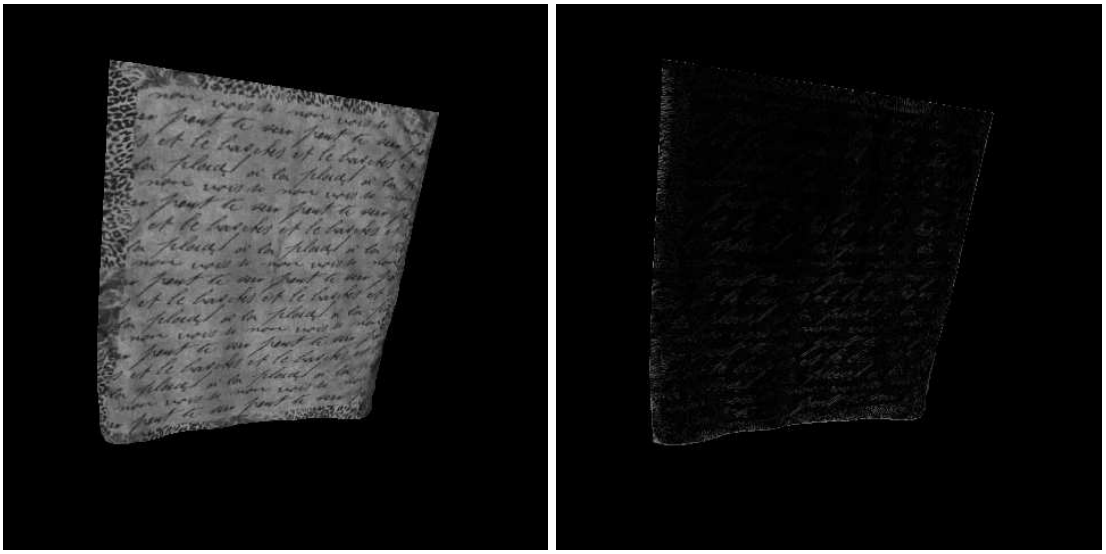


Figure 3.6: Reconstructed frame and the difference to the left input image in Fig. 3.3

(R_i, G_i, B_i) and the reconstructed images $I_r = (R_r, G_r, B_r)$ over the whole frame sequence is depicted. The mean is computed over all camera views and color channels:

$$RMSE = \sqrt{\frac{1}{3|C|}\sum_{c\in C}\sum_{x=1}^{W}\sum_{y=1}^{H}(R_i(x,y)-R_r(x,y))^2+(G_i(x,y)-G_r(x,y))^2+(B_i(x,y)-B_r(x,y))^2} \tag{3.9}$$

We have added the upper curve showing the behavior without silhouette matching. The average error is higher in this case. The lower curve (with silhouette matching) shows several local minima corresponding to frames where the cloth motion is minimal (the test sequence contains periodic motion). This shows that the deformable model constraints are able to reduce the error at these points (E_{flow} is small compared to the other error terms in this case). The average error is constant for about 200 frames and grows towards the end of the sequence as small tracking errors accumulate over time. The quality of the result should be assessed in the accompanying video. [1]

[1] `http://www.mpi-inf.mpg.de/~vscholz/vmv04/result.mpg`

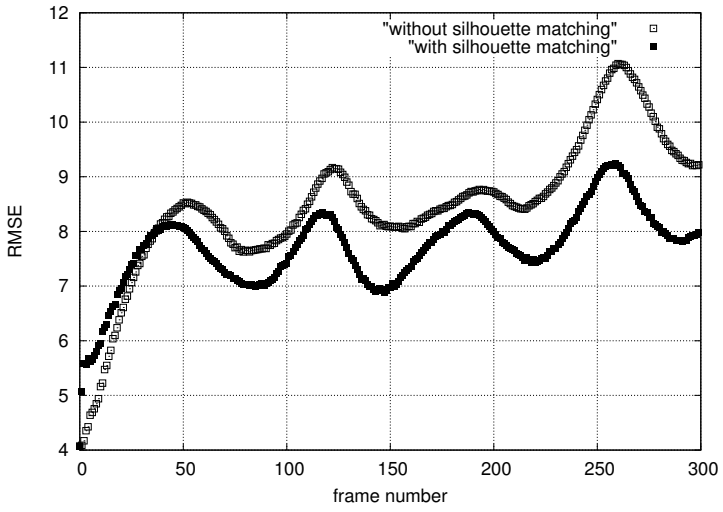


Figure 3.7: Per-pixel error with and without silhouette matching. Pixel values range from 0 to 255.

3.7 Conclusions

We have presented a method that is capable of reconstructing cloth motion for synthetic test data. A combination of optical flow and a deformable model is used to track motion robustly. We obtain photo-realistic results and can track motion over several hundred frames. Frame-to-frame coherence is achieved by our incremental approach using optical flow, adding to the realism of the captured motion. The approach is applicable to cloth with richly detailed texture.

In our current implementation we assume a flat initial cloth position. This limitation could be overcome by reconstructing the initial cloth position with a stereo method (as in [PH03]). Additionally, the parameterization of the surface with *uv* texture coordinates must be determined in the first frame by matching the cloth features to a flat reference cloth ([PH03]). Our results are based on synthetic test data, so the next step would be the application to video data from high-resolution cameras. This requires background subtraction for the silhouette matching step.

There are several limitations which prevented us from pursuing this tracking approach further. Tracking errors accumulate over time and must be corrected by silhouette

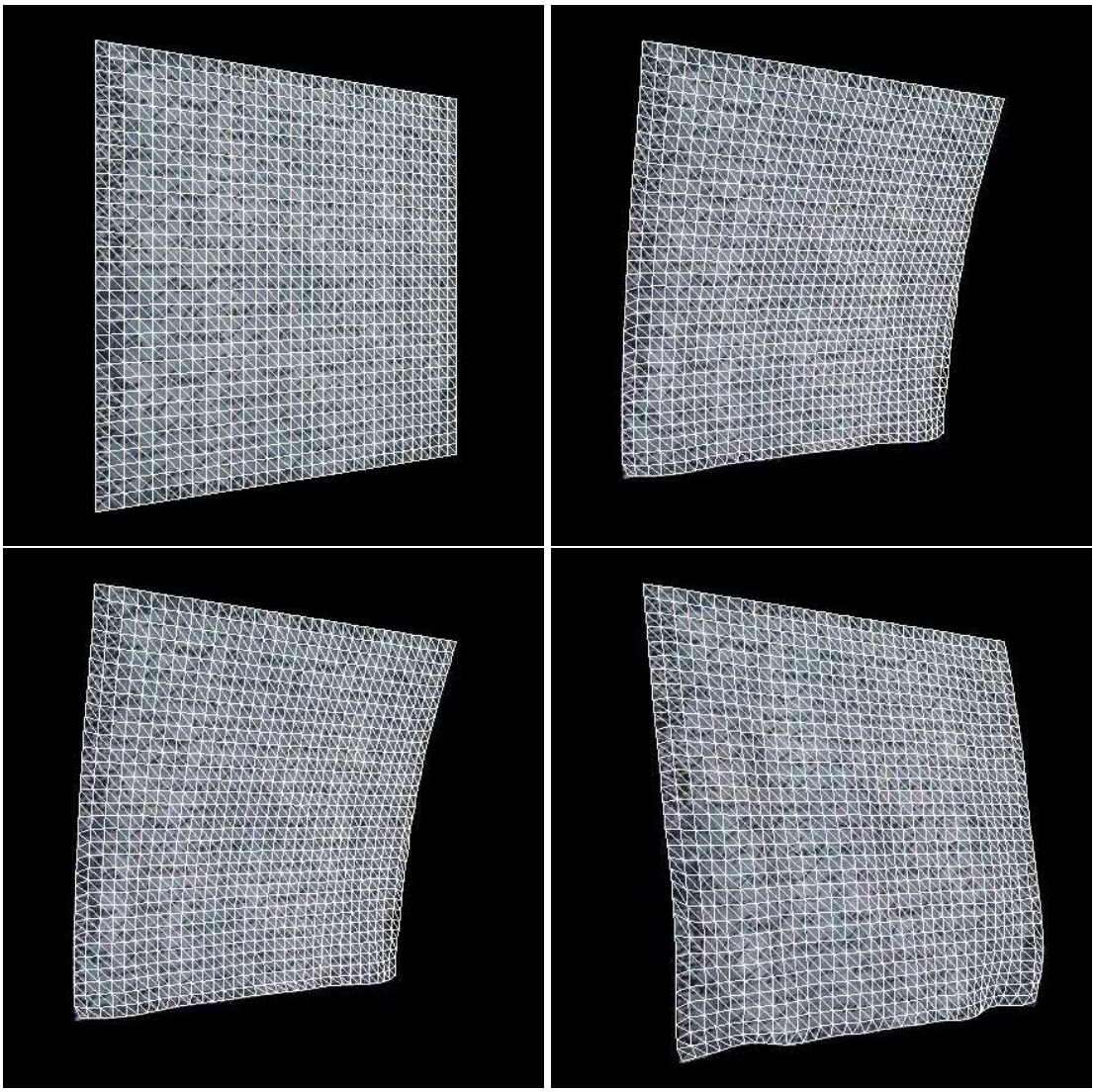


Figure 3.8: Wireframe models overlayed on the input images show the tracking accuracy of the algorithm (we show every 40th frame). A good match between the silhouettes of the input frames and the reconstructed mesh is obtained with our optimization approach.

matching. This constraint is not always available (e.g. for garments) and feature matching techniques (e.g. SIFT features) are not reliable enough for this purpose. Occlusions are also a difficult problem as tracking must be resumed after an occlusion event. The next chapter proposes a robust method which reconstructs cloth shape on a frame-by-frame basis, avoiding the difficulties of frame-to-frame tracking.

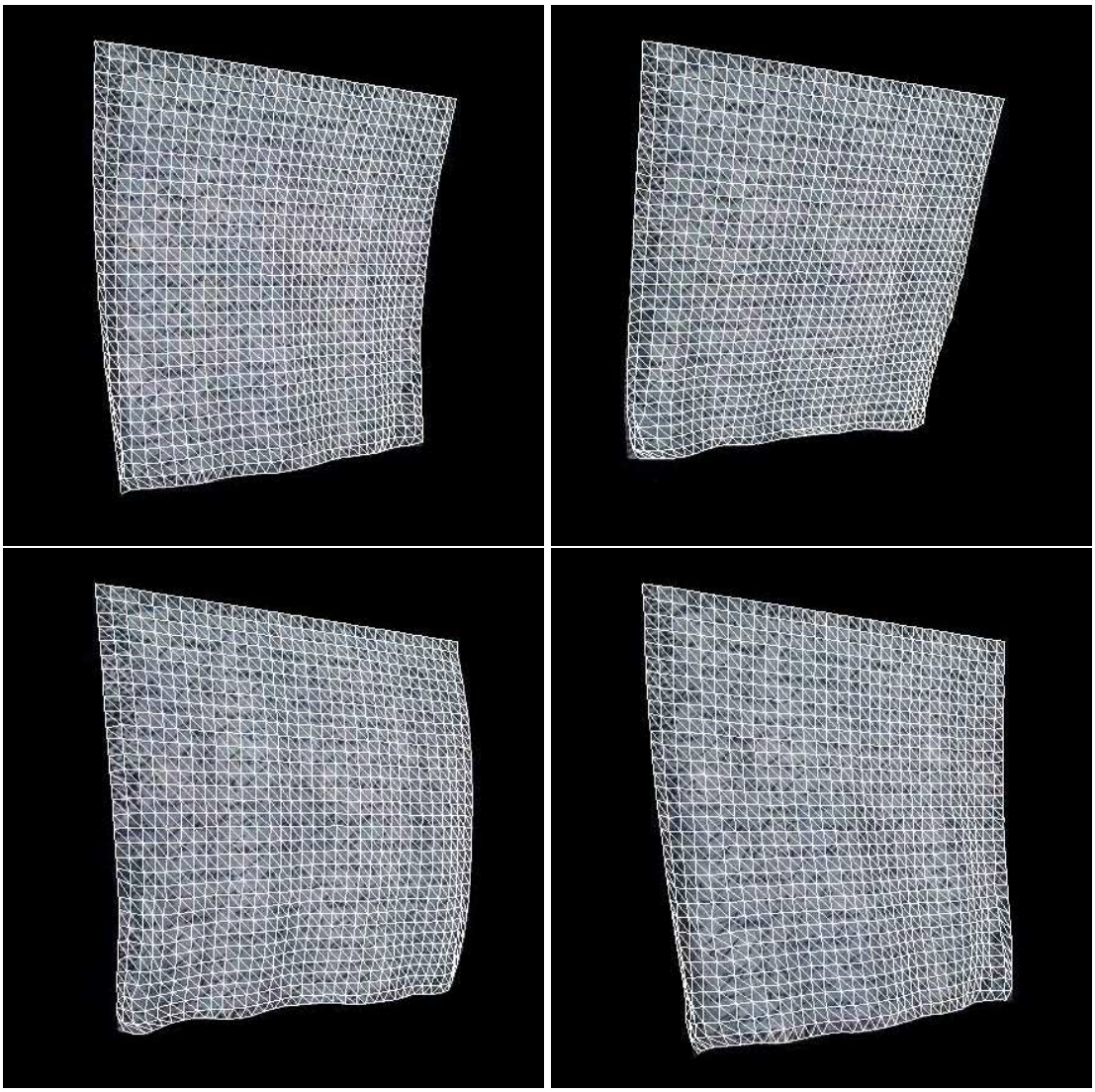


Figure 3.9: Wireframe models (remaining frames).

4

Garment Motion Capture Using Color-Coded Patterns

4.1 Introduction

After reconstructing a piece of cloth in the last chapter, we now move on to build a system capable of reconstructing large garments. We present an image-based algorithm for surface reconstruction of moving garment from multiple calibrated video cameras. Using a color-coded cloth texture, we reliably match circle-shaped features between different camera views. As surface model we use an a priori known triangle mesh. By identifying the mesh vertices with texture elements we obtain a consistent parameterization of the surface over time without further processing. Missing data points resulting from occlusion and self-shadowing are plausibly interpolated with a thin-plate spline. The deforming geometry can be used for different graphics applications, e.g. for realistic retexturing. We show results for real garments demonstrating the accuracy of the recovered flexible shape.

The following chapter is structured as follows: Section 4.2 explains the process of garment production. We then move on to describe our method for shape reconstruction in Section 4.3. Section 4.4 explains our rendering method and Section 4.5 presents the obtained results. Finally, we conclude in Section 4.6.

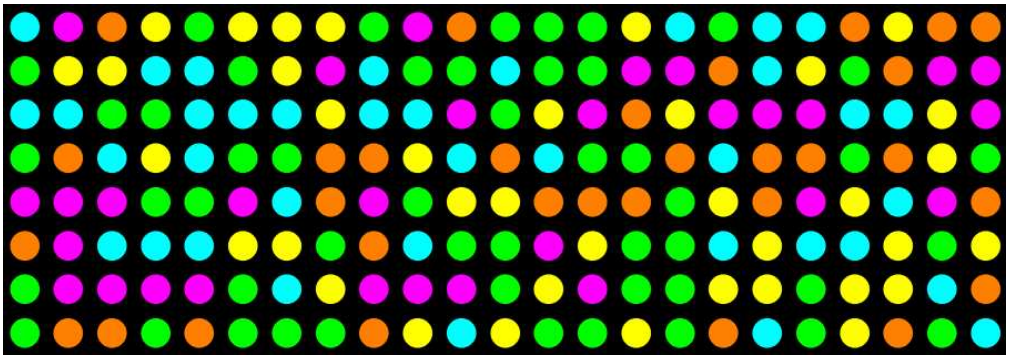


Figure 4.1: The pseudo-random color pattern used for our garments contains five colors: cyan, magenta, yellow, orange and green.

4.2 Preliminary work

Our approach requires a costum-printed cloth pattern. We describe its production in the following. Additionally, a triangle mesh for the garment is constructed as input for the acquisition algorithm.

4.2.1 Color-coded patterns

We print a pattern on our cloth which is carefully chosen to allow robust observation. Our pattern is a set of colored circular dots where the neighborhood coloring of each dot identifies the location and orientation on the cloth. The pattern elements are highly distinctive and the neighborhoods are unique over the entire cloth. Some cameras may see only a small fraction of the entire cloth due to self-occlusion, so that correspondence must be determined locally. Distinctive neighborhoods allow reconstruction even in this difficult case. Additionally, our pattern offers a high degree of spatial accuracy.

Color codes are well-known in the context of structured light reconstruction techniques [ZCS02]. In [PSGM03], a good overview of projection patterns including color codes is given. For two-dimensional cloth textures we need a pattern which encodes both dimensions. The generated pattern should be large enough for manufacturing garments while containing distinctive neighborhoods. We have chosen M-arrays [MOC+98], a

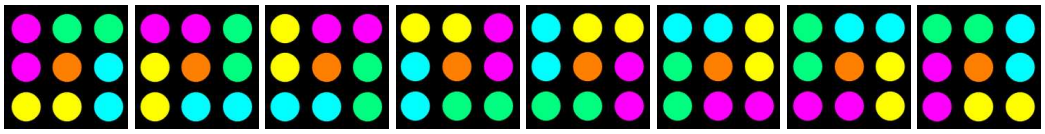


Figure 4.2: Example for a local neighborhood (leftmost image) and the seven patterns obtained by rotations in 45 degree steps. The pattern construction algorithm guarantees that only one of these eight neighborhoods is used in the garment pattern.

color code which encodes each point in the pattern by its spatial neighborhood (Figure 4.1). In this code, each 3×3 neighborhood of a point is unique and can be used for point identification (the window property). By choosing five well distinguishable colors, including the color printer primaries cyan, magenta and yellow, we are able to construct a pattern with a reasonable size for textile printing (76×251 points). For pseudo-random code generation we adopt an incremental algorithm described in [MOC+98]. It begins by seeding the top-left 3×3 window of the pattern matrix with a random color assignment and fills up the matrix by incrementally adding random colors. In each step, the window property is verified. In our case, the windows may be rotated in the camera images, i.e. the orientation of the local neighborhood is unknown. In order to make point identification invariant to rotations in the image plane, all windows are also verified against rotated versions in 45 degree steps (Figure 4.2). This reduces the number of possible codewords but still allows patterns of reasonable size. The number of possible codewords depends on the number of colors c, an upper bound is $\frac{c^9}{8}$ (≈ 480.000 for $c = 5$). The output of the algorithm is a pattern matrix M with entries for the five colors. The generated color pattern is printed on polyester fabric with a high-quality textile inkjet printer. The grid spacing between dots is 2 cm with diameter measuring 1.3 cm. The two garments, a skirt and a T-shirt are manufactured by a tailor. During this process, we take photographs of the garment panel outlines for triangle mesh construction (Figure 4.3).

4.2.2 Mesh construction

This section describes work done by Michael Keckeisen and Markus Wacker [SSK+05]. Based on the photographs of the panels, we design the virtual garment counterparts. To this end we use the cloth simulation plugin tcCloth for Alias' Maya software [GMP+04]. With this tool we construct the corresponding meshes for the garment. First, we draw the border curves of each panel using Nurbs curves. These curves are assembled to cloth panels from which single triangle meshes are constructed for the panels. We meet the additional constraint that interior mesh points are situated at the center of the color dots. This is achieved by triangulating a regular quadratic mesh, representing the color-coded texture, inside the border curves drawn in the first step. Then, the necessary seams between the cloth panels are specified, and the complete piece of cloth is constructed. In this step, we assert that the panels are triangulated in such a way that two corresponding seam lines have the same number of vertices. More precisely, each seam is given by a list of pairs of vertices being the corresponding vertices on two not necessarily distinct planar panels. Afterwards, the integrated cloth simulation based on [EKS03] is used to achieve a smooth triangle mesh as input for the reconstruction algorithm. A more detailed description of the mesh construction process can be found in [GMP+04, KFW04]. The correspondence between the mesh vertices inside the borders and the colored pattern dots is also established during mesh construction. The *uv* texture coordinates of a vertex correspond to its index in the pattern matrix M.

4.3 Cloth Motion Capture

Our system consists of eight synchronized video cameras arranged all around the person wearing the garment. In a first step, the acquired video images are processed for feature identification and matching between different camera views. Using geometric camera calibration, 3D surface points are reconstructed from the image feature positions. The

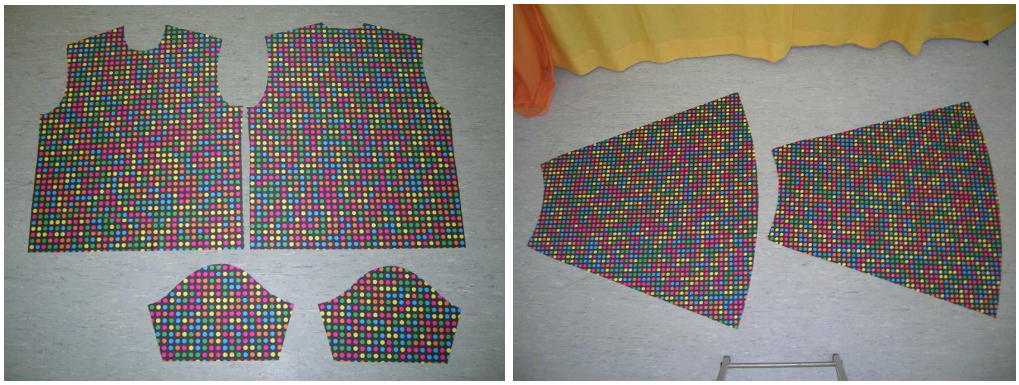


Figure 4.3: Left: the four garment panels used for the T-shirt. Right: garment panels used for the skirt. After sewing, the outlines are further reduced due to the seams which is important for mesh construction.

acquired surface is then processed by hole filling and smoothing algorithms. In the following, we go through the processing pipeline.

4.3.1 Feature recognition

The first step in our method is the recognition of colored ellipses, the image projections of our pattern dots. Color classification should be robust against illumination variations. For this purpose, we convert RGB color into the HSV color space to separate luminance from chrominance properties. Hue is representing the color information, while Saturation (whiteness) and Value (brightness) vary with illumination intensity. A common model for this variability are Gaussian distributions used in Bayesian color classifiers [VSA03]. This method requires training data for the classifier under different illumination conditions, often hand-labeled in the images [GGW+98]. For a multi-camera setup this is a tedious procedure. We use a simpler approach which uses only hue for color classification. We assume that hue remains constant over a wide range of brightness levels and use nearest neighbor classification to distinguish between five color classes. No additional thresholds are needed. For estimation of the color class hues, a training pattern with the five cloth pattern colors is recorded for every camera (Figure 4.4) as a color calibration step.

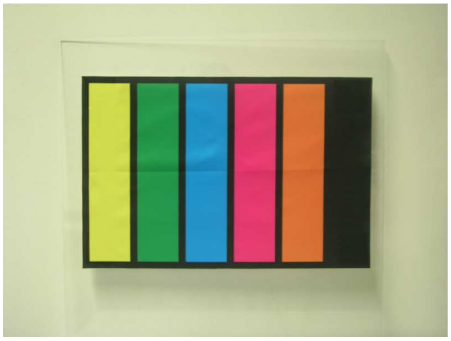


Figure 4.4: Test pattern with five colors for color hue calibration.

Color detection is affected by camera noise and does not allow to exactly estimate the projected dot shapes. We increase the robustness of feature detection by combining color with edge information. A Canny edge detector [Can86] is applied to the luminance images. This yields well-defined ellipse contours due to the high brightness contrast between the color dots and the black cloth background. The contours are further processed by outer contour following [SA85]. A lower and upper threshold for the feature area is used for filtering. For color classification, every pixel inside an ellipse votes for a color class and the ellipse color is determined by a winner-take-all strategy. Color detection works at an error rate of 5-10 percent misclassified dots. Performance deteriorates mainly in dark, shadowed areas where the color information becomes unreliable. The centers of the color dots are calculated as the center of gravity of the filled dot contours and are later used for reconstruction.

4.3.2 Feature labeling

In the next step every colored dot has to be identified by its window in the pattern matrix M and assigned an index (i, j). We use the garment panel outlines from Figure 4.3 to mask out dots which are not in the garment or clipped dots at the seams and use this as knowledge for labeling. The algorithm uses a region growing strategy. As a first step, a seed color dot has to be identified in the image. For determining its neighbors, the

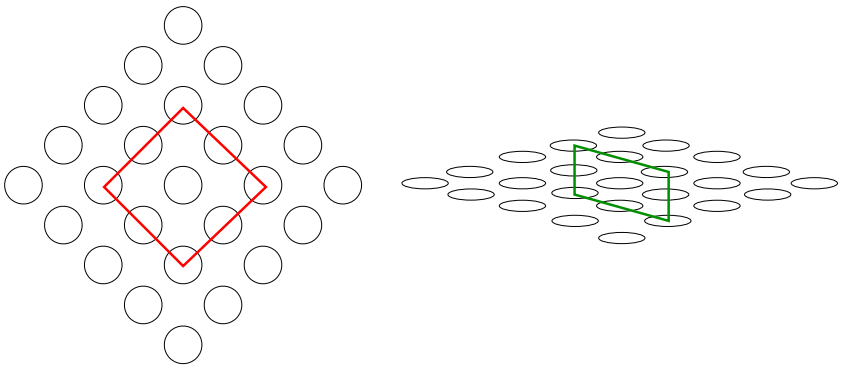


Figure 4.5: Left side: frontal view, nearest eight neighbors of the center dot. Right side: oblique view, different neighbors due to camera viewpoint foreshortening.

simplest approach would consider the nearest eight dots in the image. However, this may give incorrect results when the dots are observed under oblique angles. Dots from an extended 5×5 neighborhood can come closer (Figure 4.5). This has to be taken into account for oblique regions at cloth folds. While this complication can be avoided if a hexagonal grid with uniform neighbor distances is used, the reduced number of code-words (c^7) in this case would require additional colors which makes color classification less reliable.

Thus, we have to examine the 5×5 neighborhood of the center dot and generate three hypotheses for its 3×3 neighborhood and the local lattice directions $\mathbf{u}$, $\mathbf{v}$ and $\mathbf{u}+\mathbf{v}$ in the image (Figure 4.6). More hypotheses could be examined by mirroring the blue and green parallelograms in Figure 4.6 vertically and horizontally, but due to the symmetry of the grid this can be reduced to three main hypotheses.

The algorithm for seed finding is as follows:

1. Determine the three nearest neighbors of the center to estimate the local $\mathbf{u}$, $\mathbf{v}$ and $\mathbf{u}+\mathbf{v}$ directions (their opposite lying neighbors are determined as well).

2. Generate three different 3×3 hypotheses by shifting the base lines AB and CD along the local directions (Figure 4.6).

3. Filter out hypotheses which are not in the pattern matrix M. For searching in the

pattern, the neighbor dots are sorted by the polar angle around the center dot.

After this, we have several seed hypotheses available which are verified in the following region growing step. In this manner, seed dots can also be found for cloth areas observed under oblique angles, which is important at cloth folds. The region growing algorithm iterates over the following steps until no further dots are found and considers a 4-neighborhood (horizontal and vertical grid neighbors) for every dot:

1. Compute the local vectors $\mathbf{u}$, $\mathbf{v}$ from a 3×3 neighborhood

2. Calculate a search window for the neighbor dot (dashed window in Figure 4.6, right side). The window center is given by adding $\mathbf{u}$ (horizontal neighbors) or $\mathbf{v}$ (vertical neighbors) to the current dot center. The search window has a parallelogram shape and is spanned by $\mathbf{u}$ and $\mathbf{v}$.

3. Label an image dot in the search window with an index (i, j) if it has the correct color and the re-estimated vectors $\mathbf{u}'$, $\mathbf{v}'$ satisfy $\|\mathbf{u}' - \mathbf{u}\| \leq c\|\mathbf{u}\|$ and $\|\mathbf{v}' - \mathbf{v}\| \leq c\|\mathbf{v}\|$ with $0 \leq c \leq 1$. We use $c = 0.33$ in our experiments.

4. Propagate $\mathbf{u}'$, $\mathbf{v}'$ to the newly labeled neighbor in the direction of growth so that step 1 is skipped in the next iteration.

Step 2 implies a smoothness constraint for the lattice structure because the neighboring dot has to lie in the specified search window. This is a reasonable assumption for smooth cloth surfaces. The parameter c in step 3 is a smoothness criterion for neighboring lattice vectors. The color test in step 3 uses the a priori known pattern matrix M. At depth discontinuities, region growing should stop. This is enforced by steps 2 and 3 because in this case, the local lattice orientation is likely to change and the chance of finding a dot with the correct color on the other side of the discontinuity is only 20 percent for five colors. The growing algorithm is also likely to stop at oblique regions because the feature detection becomes more difficult due to foreshortening as can be seen in

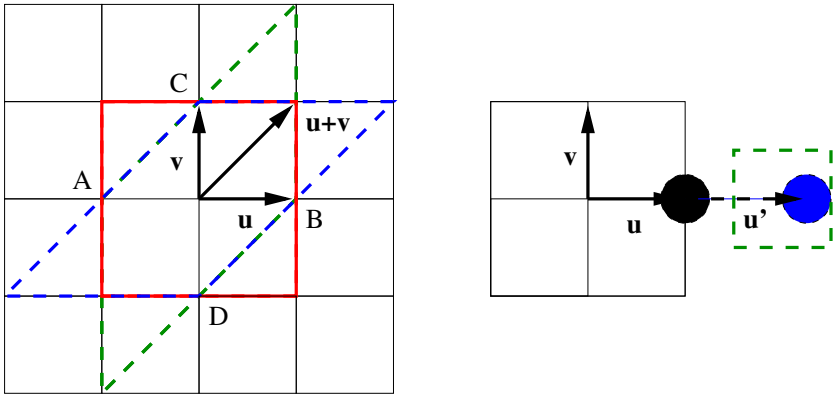


Figure 4.6: Left side: The local vectors **u**, **v** and $\mathbf{u}+\mathbf{v}$ can change roles under certain viewing angles. In addition to the obvious eight neighbors (red), the green and blue neighborhoods (dashed lines) have to be considered as well (cf. red and green parallelogram in Fig. 4.5). For example the blue parallelogram can be obtained by shifting the base line AB in the $\mathbf{u}+\mathbf{v}$ and $-(\mathbf{u}+\mathbf{v})$ directions. Right side: search window (dashed) calculated from **u**, **v**.

Figure 4.7. However, some dots are wrongly labeled which can be detected later in the reconstruction step.

A seed hypothesis is verified if a region of sufficient size is found by the region growing algorithm. Otherwise, the region growing is restarted with another seed hypothesis. After one region has been found successfully, region growing is restarted with the next seed candidate until no further regions can be labelled.

The region growing approach can cope with arbitrary image background since it exploits the regular lattice structure in the image (i.e. no explicit segmentation of the garment region is necessary). Figure 4.7 shows the output after region growing. The region growing algorithm considers only vertical and horizontal neighbors which increases robustness. For most images only a few seed points (regions) are needed to label all dots. Algorithm performance deteriorates for oblique angles, but these areas don't deliver accurate measurements for reconstruction anyway. Fig. 4.13 summarizes the feature recognition and labeling steps.

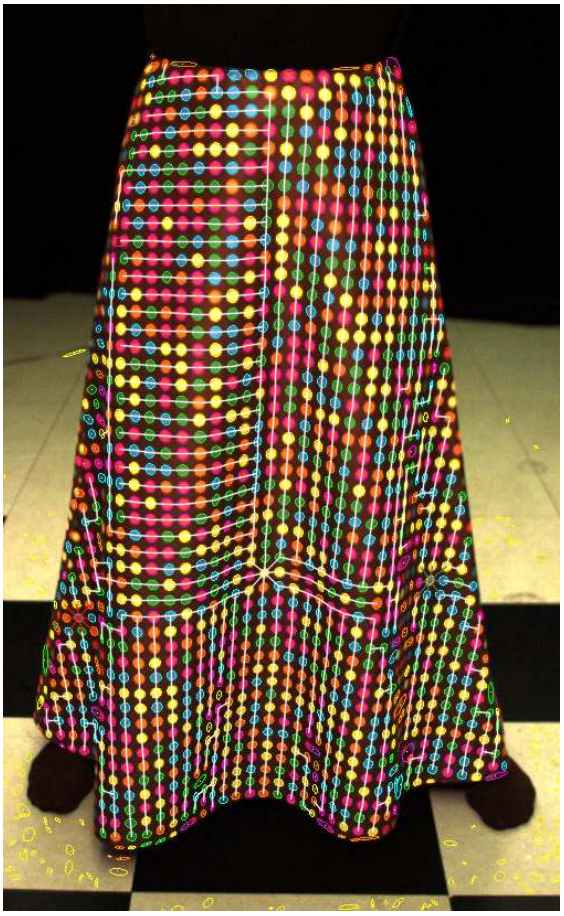


Figure 4.7: Output of the labeling algorithm. The star-shaped crosses (left side, center) mark seed points where region growing starts. Every color dot is connected with its predecessor, yielding the tree structure of the iterative algorithm shown as white lines. In oblique regions, the feature detection becomes unreliable.

4.3.3 Reconstruction

After labeling, the image projections of the visible pattern dots are known. We employ the linear triangulation method from [HZ00] for reconstruction. All image measurements $\mathbf{x}_i = (x_i, y_i) = P_i\mathbf{X}$, $i \in \{0 \ldots n-1\}$ of a pattern point in n camera views are used. P_i denotes the 3×4 camera projection matrix of camera i with rows $\mathbf{p}_i^{1T}, \ldots, \mathbf{p}_i^{3T}$ and $\mathbf{X}$ the 3D reconstruction. A linear system $A\mathbf{X} = 0$ is solved with singular value decompo-

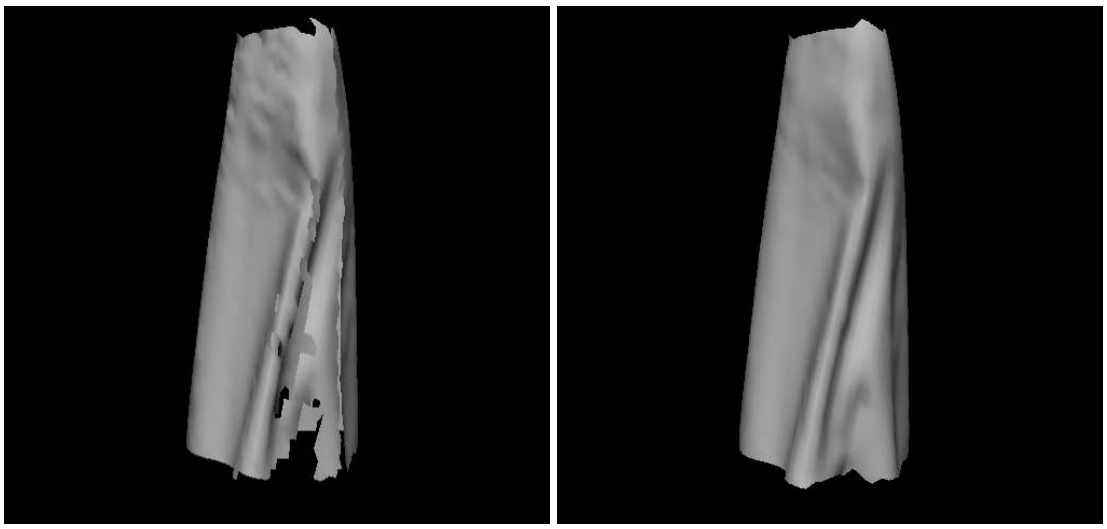


Figure 4.8: Missing data after reconstruction due to self-occlusion, and result after thin-plate hole filling and smoothing.

sition:

$$\begin{pmatrix} x_0\mathbf{p}_0^{3T} - \mathbf{p}_0^{1T} \\ y_0\mathbf{p}_0^{3T} - \mathbf{p}_0^{2T} \\ \dots \\ x_{n-1}\mathbf{p}_{n-1}^{3T} - \mathbf{p}_{n-1}^{1T} \\ y_{n-1}\mathbf{p}_{n-1}^{3T} - \mathbf{p}_{n-1}^{2T} \end{pmatrix} \mathbf{X} = 0 \tag{4.1}$$

The solution $\mathbf{X}$ minimizes the reprojection error in the images in a least squares sense. The maximum reprojection error is computed for every reconstructed point in all views. Outlier points are removed by comparing the error with an upper threshold. The reconstructed points deliver the vertex coordinates for the garment's triangle mesh.

4.3.4 Hole interpolation

The resulting surface contains holes in areas of missing data (Figure 4.8). One reason for missing data is self-shadowing. In these areas, color classification does not work properly. Deep cloth folds may only be seen by one camera, in which case a reconstruc-

tion due to self occlusion is not possible. As a post-processing step we employ mesh interpolation with thin-plate splines to fill hole areas. Thin-plate interpolation yields smooth surfaces which makes this approach suitable for cloth surfaces. The thin plate energy

$$E(f) = \int_{\mathbb{R}^2} f_{uu} + 2f_{uv} + f_{vv}\, du\, dv \tag{4.2}$$

for a function $f : \mathbb{R}^2 \to \mathbb{R}$ punishes strong bending and the corresponding minimum energy surface is given by $\Delta^2 f = 0$ [Wah90] where Δ^2 denotes the Bilaplacian operator. For triangle meshes the uniform Laplacian for a mesh vertex $\mathbf{p}$ with vertex neighbors $\mathbf{p}_i$ is discretized as [KCVS98]:

$$L(\mathbf{p}) = \frac{1}{n} \sum_{i=0}^{n-1} (\mathbf{p}_i - \mathbf{p}) \tag{4.3}$$

The Bilaplacian operator Δ^2 is thus

$$L^2(\mathbf{p}) = \frac{1}{n} \sum_{i=0}^{n-1} (L(\mathbf{p}_i) - L(\mathbf{p})) \tag{4.4}$$

For hole filling, we solve a linear system $A\mathbf{P} = \mathbf{b}$ for the vertex coordinates $\mathbf{P} = (\mathbf{p}_0, \ldots, \mathbf{p}_{n-1})$. We require two nested rings of boundary vertices around the hole which are fixed. We add for these vertices $\mathbf{p}_i$ the equation $\mathbf{p}_i = \mathbf{c}_i$ in the linear system, where the $\mathbf{c}_i$ are the original vertex positions. This imposes a C^1-continuous boundary condition on the problem. The hole vertices can move freely. $L^2(\mathbf{p}_i) = 0$ is added to the linear system for every free vertex. The corresponding matrix is sparse and the linear system can be solved efficiently with iterative methods like GMRES [GL96].

The mesh holes are determined by a region growing algorithm which traverses the mesh topology. For each hole, a separate linear system is solved. The uniform Bilaplacian operator in Equation (4.4) leads to uniform edge lengths. To preserve the quadratic mesh structure with its longer diagonal edges, the vertex neighborhood is restricted to

Figure 4.9: Camera setup using eight cameras and two HMI lamps with softboxes.

the horizontal and vertical neighbors for computing the Bilaplacian. In order to remove noise artifacts, the mesh is slightly smoothed by Laplacian smoothing [Tau95] using the same neighborhood structure in the spatial and temporal domain. This method adjusts the location of each mesh vertex to the geometric center of its neighbor vertices iteratively and eliminates high-frequency noise quickly. Outlier vertices are removed and interpolated from neighbors by analyzing the spatial and temporal neighborhood.

4.4 Rendering

This section describes work done by Timo Stich [SSK+05]. The visualization of the dynamic surfaces is implemented in OpenGL which results in interactive framerates ($>$ 30 fps) on a GeForce4 graphics adapter. Rendering the reconstructed meshes as wireframe, shaded or with various textures is possible (Fig. 4.12). The textures are created from cloth photographs which improves the visual appeal of our renderings. Furthermore the viewpoint can be interactively chosen from one of the original camera viewpoints or a freely adjustable user-defined view. If the user chooses an original camera viewpoint, the recorded images are used as background images. The background

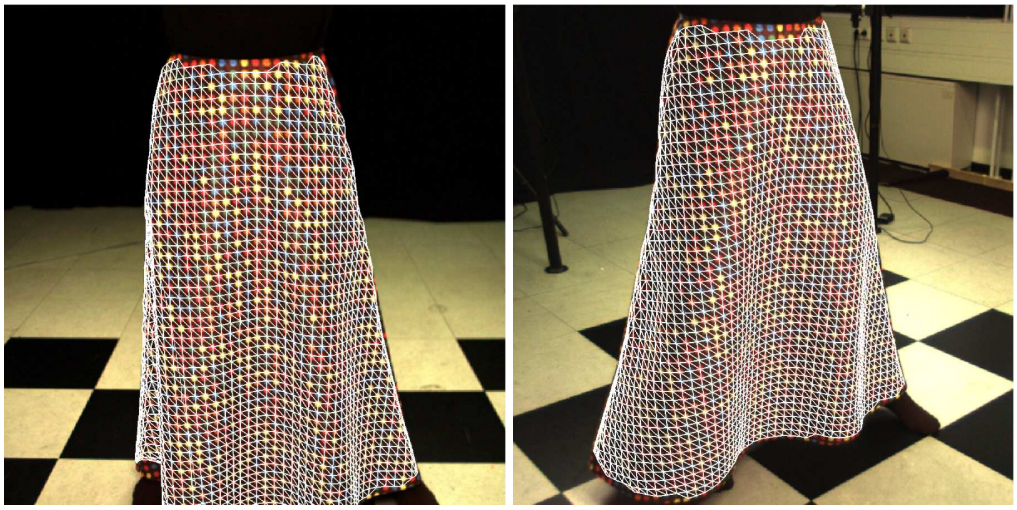


Figure 4.10: Overlayed wireframe renderings of the surface in two camera perspectives.

images are warped in a pre-processing step to remove lens distortion effects [Bou05]. These are estimated during camera calibration. Since the reconstructed garments are stored in a standard 3D mesh file format (Alias Wavefront OBJ), it is easy to integrate the textured dynamic garments into virtual environments using 3D animation software.

4.5 Results

The scenes were recorded with eight synchronized Imperx MDC-1004C video cameras arranged in a circle around the scene (Fig. 4.9, 4.14 and 4.16). We recorded with a frame rate of 25 frames per second and an image resolution of 1004×1004 pixels. A controlled lighting environment improves the reconstruction results significantly by reducing the effects of self-shadowing. The scene is illuminated by two HMI (Hydrargyrum Medium-arc Iodide) lamps with softboxes in order to obtain diffuse light sources. We use HMI lamps because color recognition works best with a spectral distribution similar to daylight. Standard tungsten halogen lamps have a higher output at the red end of the spectrum and reduce the distance of the observed colors in color space.

For camera calibration we use the Camera Calibration Toolbox for MATLAB [Bou05] which uses Zhang's calibration method [Zha99]. Our complete reconstruction method

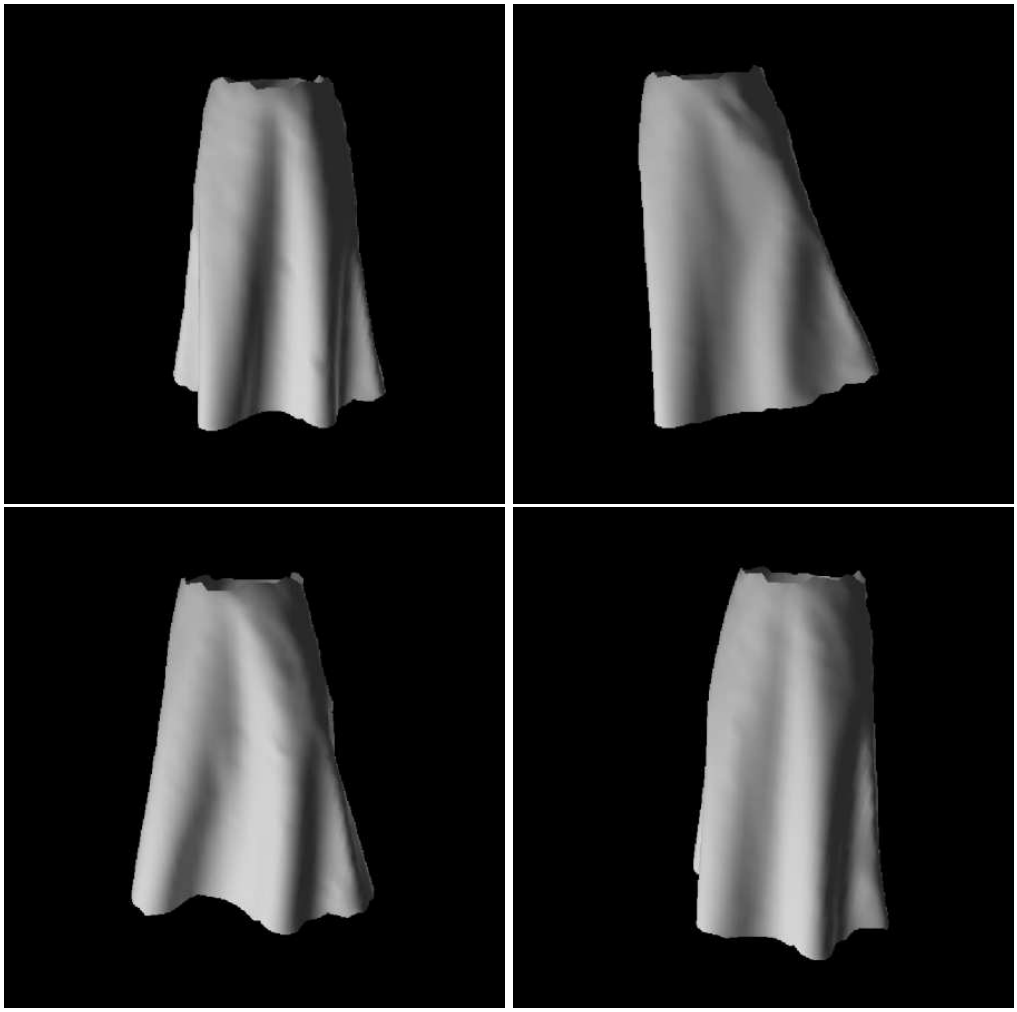


Figure 4.11: Reconstructed surface for different time instances.

requires from feature recognition to mesh post-processing approximately 30 seconds per video frame on a Pentium IV 3.2 GHz. This time includes image processing of eight video frames with resolution 1004×1004. The garment triangle meshes have 3000-3500 mesh vertices. It is difficult to evaluate the reconstruction results when ground truth is not available. We render the obtained surface into the original video images to estimate the accuracy of the acquired shape qualitatively (Figure 4.10). The matching between vertex positions and dot centers looks reasonable. The largest discrepancies occur in dark cloth areas and near the image borders due to decreasing camera calibration accuracy. Furthermore, the visible folds and the overall fall of the garment are

Figure 4.12: Arbitrary texture can be applied to the reconstructed dynamic surface.

recovered (Figures 4.11, 4.15 and 4.17). Results are best assessed in the accompanying video, which shows examples for slow and fast motion.[1]

4.6 Conclusions

We have presented a method for robustly acquiring complex cloth motion. By using color-coded textures, we can establish reliable point correspondences between different

[1] http://www.mpi-inf.mpg.de/~vscholz/eg05/ccgarment.avi

camera views. A prior triangle mesh model enables us to plausibly fill in missing data. Our method provides a surface parameterization that allows retexturing and rendering the dynamic surfaces realistically from arbitrary viewpoints.

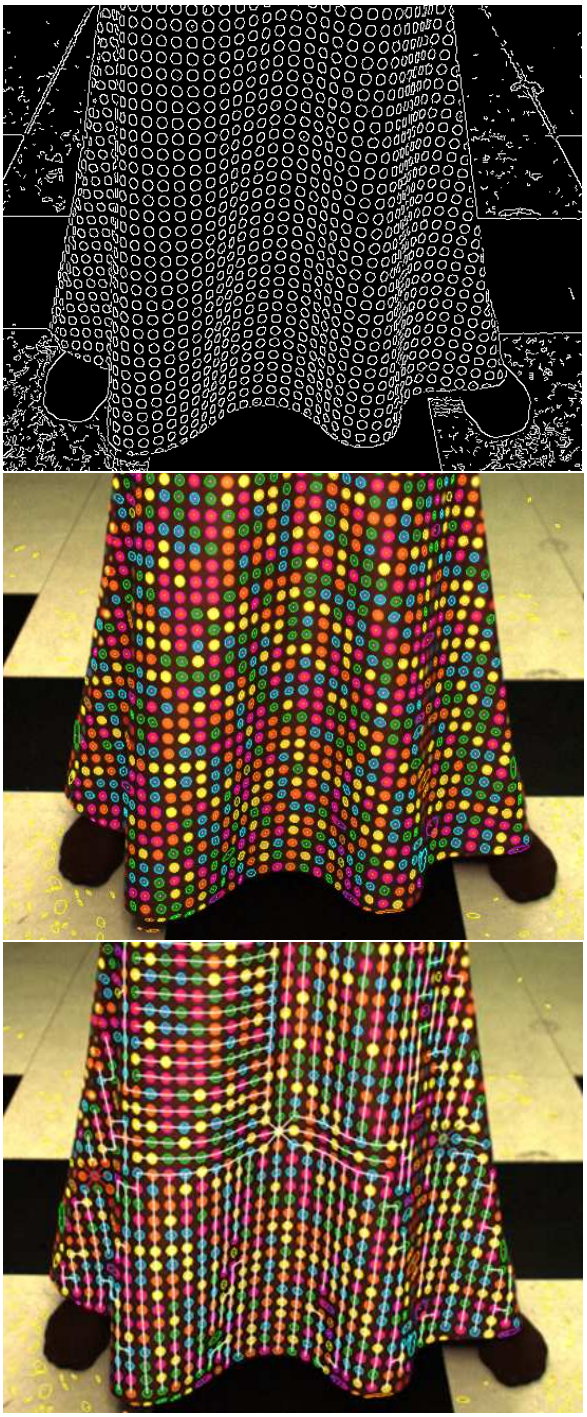


Figure 4.13: Image processing steps from top to bottom: edge detection, color classification and labeled features.

Figure 4.14: Four input camera views for the same moment in time.

Figure 4.15: The reconstructed surface faithfully represents the cloth folds visible in the input frames.

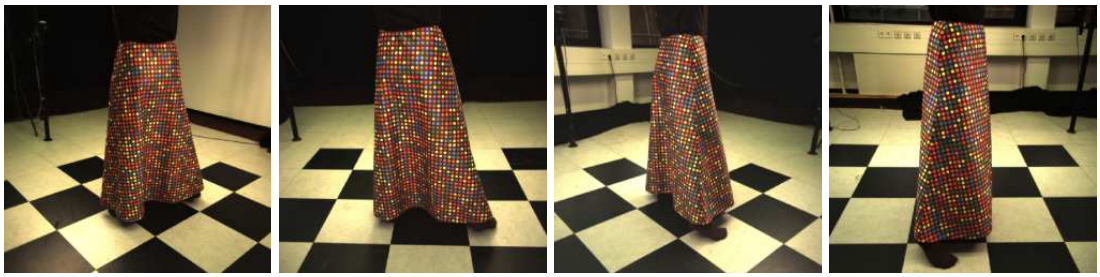


Figure 4.16: The other four camera views.

Figure 4.17: Reconstruction results for the T-shirt.

5

Texture Replacement of Garments in Monocular Video Sequences

5.1 Introduction

Figure 5.1: Input frame (left) and texture replacement result (right). Notice how shading adds an important visual cue.

In this chapter, we leave the domain of multi-camera video systems and present a video processing algorithm for texture replacement of moving garments in monocular video recordings. We use our color-coded pattern from the last chapter to determine the geometric deformation of the texture. A time-coherent texture interpolation is obtained by the use of 3D radial basis functions. Shading maps are determined with a surface

interpolation technique and applied to new textures which replace the color pattern in the video sequence. Our method enables exchanging fabric pattern designs of garments worn by actors as a video post-processing step (Figure 5.1). It could also be useful for virtual fashion presentation in e-commerce.

Current rotoscoping software allows tracking edges or single features in videos for tasks like matting of computer-generated imagery (CGI) objects, selective filtering and creating cartoon animation from video [AHSS04]. For our purpose, however, an automatic approach is needed which can track several hundred texture features in parallel while handling occlusions automatically. Manual texture editing is in this case almost infeasible. We propose such a system to enable texture replacement with correct texture deformation and lighting effects.

The chapter is organized as follows. Section 5.2 gives an overview of our system. In Section 5.3-5.6 we detail our proposed method. Section 5.7 presents results. We end in Section 5.8 by drawing conclusions from our work.

5.2 Overview

Figure 5.2 shows an overview of our system. For proper texture replacement, we need a segmentation of the images into garment and background sections. For this purpose we use the rotoscoping software by Agarwala et al. [AHSS04] for contour tracking which requires the user to specify contour curves at keyframes. The contour is tracked through the remaining frames by minimizing an energy functional based on optical flow [LK81a] and active contours [KWT88]. In general, any other video segmentation method [LSS05, WBC$^+$05] could also be used. These methods all require some amount of user interaction. This preprocessing step yields boundary curves which are converted into a binary mask for the foreground where all further processing is done. Next, we perform image processing for feature classification. We use garments with the custom-

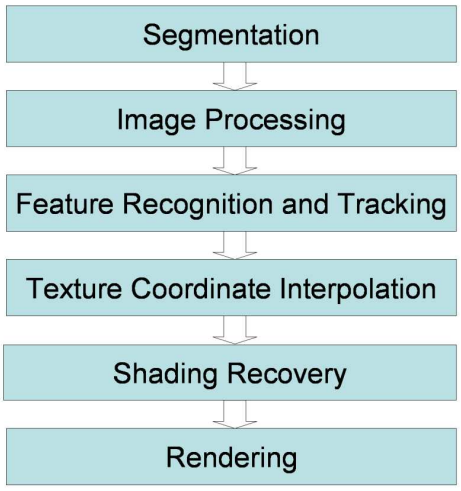


Figure 5.2: Overview of the processing steps of our method.

designed color-coded pattern introduced in the last chapter. We use the single-frame method from Section 4.3.2 which identifies the dots by their local neighborhood with a region-growing approach. Texture coordinates can then automatically be assigned to each dot. The results of the labeling algorithm are complemented by a feature tracker and fed into a texture interpolation algorithm which determines a time-coherent image texture from the feature positions. For realistic texture shading we determine the shading image. The new texture is rendered into each video frame by multiplying texture color with the corresponding shading image.

5.3 Image Processing

We convert the input video images into HSV color space in order to increase color recognition robustness against illumination changes. For color classification, we only use hue and learn the five color classes from an example image taken from the video sequence. Thus there is no need to record a training pattern as in Section 4.3.1. The feature pixels are identified with an adaptive threshold algorithm [GW02] in the luminance image. It computes a binary image by choosing a threshold based on local image brightness and

Figure 5.3: Result of color classification algorithm, visualized with artificial colors.

can therefore adapt to local brightness variations. From these pixel positions we collect the hue values and fit a one-dimensional Gaussian distribution to each color class (Gaussian mixture model GMM) with a statistical technique [FH04c]. This can be cast as a clustering problem where each cluster corresponds to a color class. First, we run the k-means algorithm with random initial centers on the input data and apply the EM (expectation maximization) algorithm for determining GMMs [HTF01]. As a local optimization technique the EM algorithm can stagnate in local minima. This procedure is restarted 10 times and the fitting result with the best log-likelihood is kept as the final result. After this step, the pixels segmented by the adaptive thresholding method can be classified into five color classes. For this purpose we compute maximum-likelihood decision boundaries from the Gaussian parameters μ_i, σ_i of each color class. The color-classified image is labeled with a connected component algorithm for every color separately [HS92]. The obtained features are filtered by an upper and lower bound for their area. Finally, for every feature the center of mass is calculated. We now have 2D image coordinates of a number of color dots on the garment (Figure 5.3).

Figure 5.4: Our dress has three panels: one front panel and two back panels (top). The skirt has a front and a back panel (bottom).

5.4 Feature Labeling and Tracking

We use two different pieces of apparel for our experiments, a dress and a skirt. A known condition is the pattern matrix M which contains a color label for each dot in the pattern. We identify the outline of the individual cloth panels (three for the dress and two for the skirt, Fig. 5.4) in the pattern matrix manually and identify the boundary dots adjacent to the seams. Panel boundaries are interactively identified only once per garment. The algorithm proposed in Section 4.3.2 labels the features obtained in the image processing

step with their indices i, j in the pattern matrix M. The obtained indices i, j yield the texture coordinates for the feature dots. The original feature labeling algorithm is a robust single-frame method which does not use tracking history. Algorithm performance deteriorates at oblique surface angles and it requires also that a seed with a $3x3$ neighborhood can be identified for each connected texture component in the image. In order to increase the number of recognized features we apply a Lucas-Kanade feature tracker to fill in missing features after labeling [LK81b, Int01]. We track the features known from labeling with image patches and set the patch size to the mean distance of neighboring features. In order to handle feature occlusions between two video frames, we run the tracker forward in time and track the obtained result backwards. As occlusion test, we compute the deviation from the original feature position. If it is below some threshold (1 pixel in our experiments), the feature was tracked successfully and is added to the list of labeled features. Feature tracking is applied to the whole video sequence forward and backward in time. In the forward tracking phase, feature occlusions are detected while in the backward tracking phase, disocclusions (new appearing features) are added.

5.5 Texture Coordinate Interpolation

Our garments consist of several panels (Fig. 5.4). Texture interpolation is done separately for every panel so our method is also applicable to several pieces of clothing at the same time. In the following we assume that we have no tears in the fabric, i.e. the panels are continuous. In views where several panels are visible in the image, we have to find the seams between the panels in order to determine the panel segments. We determine the visible seams in the image by identifying dots at the panel boundaries which are known from the pattern matrix. As additional information we know which boundary dots of different panels are adjacent to each other at the seams. The boundary dots lie inside the panel, not on the seam and do not define a smooth boundary due to

Figure 5.5: Texture Coordinate Interpolation: The feature positions (left) are interpolated to obtain a *uv* parameterization of the garment (right).

the discrete nature of the pattern. A smooth boundary polyline is obtained by interpolating a new seam point between each pair of adjacent boundary dots. We use membrane interpolation $\Delta f = 0$ where Δ is the Laplacian operator to interpolate these seam points. The position of the boundary dots are fixed and used as boundary condition. In order to obtain a smooth polyline, we use a weighted Laplacian stencil in the corresponding linear system which assigns a higher weight to neighboring seam points. The seam points define an estimate of the panel seam which cannot be determined from the images directly (Fig. 5.6). The seam polylines are used to cut out a mask for each visible panel from the foreground mask.

Our goal is a temporally smooth parametrization of the garment region with texture coordinates (Fig. 5.5). Near the silhouettes, the feature trajectories are not stable enough (due to failure of detection and occlusions). Smoothing individual trajectories would

Figure 5.6: Boundary dots (white) and interpolated seam boundary (red) necessary for texture replacement of separate garment panels.

not be helpful in this case. Therefore we integrate the smoothing into the interpolation function. Radial Basis Functions (RBF) are commonly used for scattered data interpolation problems like reconstructing surfaces from point clouds [CBC+01]. A trivariate scalar RBF is defined by a set of centers $\mathbf{c}_i \in \mathbb{R}^3$ and weights $w_i \in \mathbb{R}$ as [CBC+01]

$$f(\mathbf{x}) = p(\mathbf{x}) + \sum_i w_i \cdot \phi(\mathbf{x} - \mathbf{c_i}) \tag{5.1}$$

where ϕ is the basis function and $p(\mathbf{x})$ is a polynomial of low degree. Since basis functions with local support do not provide the same degree of extrapolation and hole filling capabilities as functions of global support [CBC+01], we use the global basis function $\phi(\mathbf{x}) = \|\mathbf{x}\|$ where $\|\,.\,\|$ is the Euclidean norm, and a linear polynomial p. The resulting surface is a biharmonic thin-plate spline. For interpolating texture coordinates $(u\,v)^T \in \mathbb{R}^2$ we use a vector-valued RBF

$$\mathbf{f}(\mathbf{x}) = \mathbf{p}(\mathbf{x}) + \sum_i \mathbf{w_i} \cdot \phi(\mathbf{x} - \mathbf{c_i}) \tag{5.2}$$

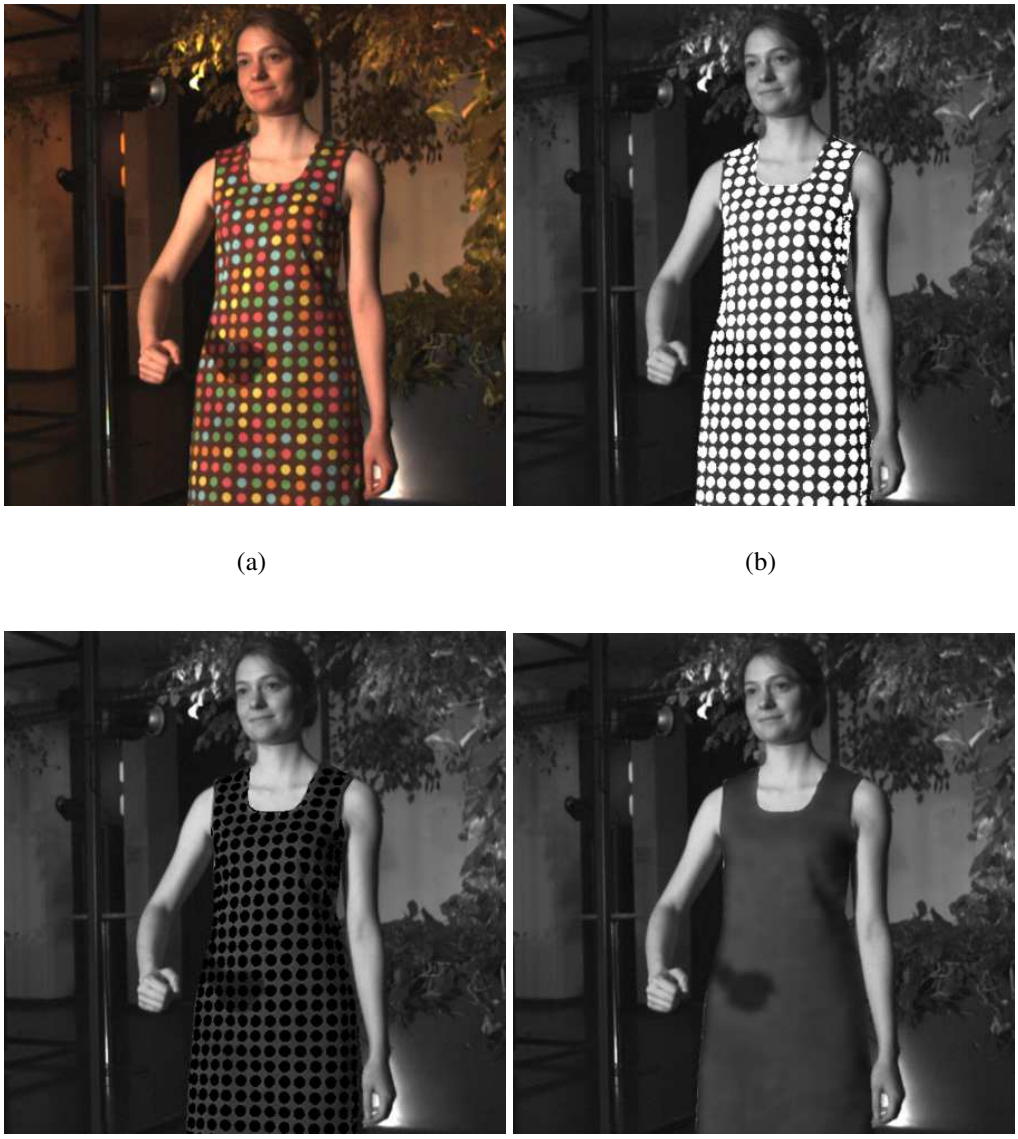
with $\mathbf{f}: \mathbb{R}^3 \rightarrow \mathbb{R}^2$ and $\mathbf{w_i} \in \mathbb{R}^2$, $\mathbf{p} \in \mathbb{R}^2$ are vectors. $\mathbf{f}$ is defined in spatiotemporal 3D space (x,y,t) for temporally smooth texture interpolation. This means we have to add a time coordinate to the obtained feature positions x, y. The difference $t_{n+1} - t_n$ of adjacent video frames is set to the mean distance of neighboring features in frame n in order to make the method adaptive to feature scale. We now use RBF approximation (also known as spline smoothing [Wah90]) by solving

$$\begin{pmatrix} \Phi - 8N\pi\rho I & P \\ P^T & 0 \end{pmatrix} \begin{pmatrix} \mathbf{w}_i \\ \mathbf{q}_i \end{pmatrix} = \begin{pmatrix} \mathbf{f} \\ 0 \end{pmatrix} \tag{5.3}$$

where $\Phi_{ij} = \phi(\mathbf{c_i} - \mathbf{c_j})$, $P_{ij} = p_j(\mathbf{c_i})$ for the polynomial basis $\{p_1, p_2, p_3\} = \{1, x, y\}$. The $\mathbf{q}_i$ are polynomial coefficients, N is the number of centers and I is the identity matrix. ρ is a parameter that determines the trade-off between smoothness of the surface and fidelity of the data. This parameter is found empirically. We use $\rho = 0.005$ for all examples (the smallest amount of smoothing which leads to reasonable results). The resulting matrix is dense due to the global nature of ϕ and can be solved directly with LU decomposition for our problem size of $N \leq 1000$ centers. For larger problems special solvers like the Fast Multipole Method are required [CBC+01]. RBF approximation is used for overlapping time windows of three video frames to ensure temporal smoothness. The texture coordinates are interpolated for every pixel in the foreground mask of the middle frame of the temporal window. Larger time windows do not improve the results significantly.

5.6 Shading Maps

The goal of the shading algorithm is to remove the reflectance contribution of the color dots from the luminance images I while preserving shading effects. We interpolate the dot regions with smooth thin-plate splines in order to get a homogeneous shading map.



(a) (b)

(c) (d)

Figure 5.7: Input image (a), detected dots (b), removed dots (c), and the shading map (d). Although the input contains strong shadow edges, the interpolation results are satisfactorily close to the input frame (a). The result video also reveals a faint shadow from a secondary light source.

The dot pixels identified by adaptive thresholding from Section 5.3 are used as input for shading map computation. We assume here that the dot edges have a higher contrast than shadow edges on the garment which is the case in most practical situations. The detected dots are dilated with a circle-shaped morphological structure element [GW02]

in order to remove the dots reliably (Fig. 5.7b). The inverse binary image yields a mask which is multiplied with the image I. We interpolate the deleted dot regions (Fig. 5.7c) by using a surface interpolation method for height fields [Ter88]. An approximating thin-plate surface is fitted to the luminance values of the dark garment background and interpolates the deleted dot regions. In our first experiments we tried to extract shading information from the dot regions as well but the results where not satisfactory [SM06a]. The surface fitting is done by minimizing the energy functional

$$E = \int_{x_{min}}^{x_{max}} \int_{y_{min}}^{y_{max}} \frac{\alpha(x,y)}{2}(I-J)^2 + (J_{xx}^2 + 2J_{xy}^2 + J_{yy}^2)\,dx\,dy \tag{5.4}$$

where J is the thin-plate surface interpolant, the integration region is a bounding box of the segmented garment region and $\alpha(x,y)$ a weight for the data compatibility term $(I-J)^2$. As a smoothness function we use the thin plate model, which minimizes surface curvature. Another possible smoothness functional is the membrane model [Ter84]

$$E_{smooth} = \int_{x_{min}}^{x_{max}} \int_{y_{min}}^{y_{max}} J_x^2 + J_y^2\,dx\,dy \tag{5.5}$$

which minimizes surface area and behaves like a rubber surface, i.e. the surface is only C^0-continuous at the border of interpolated regions. We prefer the C^1-continuity of the thin plate model for shading maps without artifacts.

We set $\alpha = 0$ in dot regions (interpolation) and $\alpha = 0.1$ for the remaining pixels (approximation). The value of this regularization parameter was found empirically to obtain shading maps without artifacts at the dot borders. A larger α value would yield an exact fit to the existing data. On the other hand data noise has to be regularized as the transition to the noise free interpolated regions would become visible otherwise. The energy function 5.4 is quadratic, and hence has only one local energy minimum. The

Figure 5.8: Texture map, shading map and result of multiplication.

corresponding Euler-Lagrange equation is

$$\alpha(I-J)+\Delta^2 J=0 \tag{5.6}$$

where Δ^2 denotes the Bilaplacian operator. Eq. 5.6 is solved with finite differences on the pixel grid and uses a $5x5$ stencil for the Bilaplacian [Ter88] (5-point stencil for the Laplacian and 13-point stencil for the Bilaplacian):

$$\begin{pmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{pmatrix} \qquad \begin{pmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 2 & -8 & 2 & 0 \\ 1 & -8 & 20 & -8 & 1 \\ 0 & 2 & -8 & 2 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{pmatrix} \tag{5.7}$$

A bounding box of the foreground mask is computed and Eq. 5.6 is solved on this rectangle. At the bounding box borders not all 13 neighbors in the $5x5$ stencil might exist so we recompute the stencil for the existing neighbors [Ter88]. Here is an example

where the missing entries are marked with an asterisk (which are actually zero):

$$\begin{pmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 2 & -6 & 2 & 0 \\ 1 & -6 & 11 & -6 & 1 \\ 0 & * & * & * & 0 \\ 0 & 0 & * & 0 & 0 \end{pmatrix} \tag{5.8}$$

This leads to a sparse linear system $Ax = b$ where the number of variables equals the number of reconstructed pixels. The linear systems have up to $n = 400.000$ variables and are solved in MATLAB. The single-frame shading maps show temporal fluctuations. Therefore we filter the shading maps with a temporal Gaussian filter per pixel (window size 3-5 frames). In order to get an accurate result for fast image motion, we build pixel correspondences between different frames by using the feature correspondences obtained during feature tracking. The features deliver a sparse set of flow vectors (we compute forward flow to the next frame and backward flow to the previous frame). This set is interpolated per pixel by fitting a 2D thin-plate smoothing spline [Wah90] with $\alpha = 10$. The obtained flow fields are used as spatial offsets during temporal filtering.
The obtained shading maps are applied to the new texture during rendering by multiplication per-texel (Fig. 5.8). For the texture lookup we use bilinear interpolation. As our garments have a dark background color, we adapt the maps to a higher albedo by rescaling. The maps are rescaled by dividing with a reference white value which is obtained by recording a reference image with maximum brightness of the fabric. Note that this is correct for fabric with Lambertian reflectance only. Fortunately, our fabric is close to Lambertian. As final step we apply a gamma correction to the rendered images as the camera sensor has an almost linear response.

Figure 5.9: Accuracy of texture distortion visualized as overlayed checkerboard texture.

5.7 Results

We record our sequences with an Imperx MDC-1004C vision camera (1004x1004 pixels) at 25 frames per second with raw output in order to avoid compression artifacts. We put the camera on a tripod. However, our method is not limited to static camera position and works just as well for hand-held camera sequences. The camera is color-calibrated with a reference Gretag MacBeth color checker DC. A linear regression model is fitted to yield a priori known color values, which results in a $3x4$ color correction matrix [Ihr07]. This step improves the separation of the garments' dot colors in color space for feature classification. For the garments we use a cotton fabric with a custom-printed color pattern using a medium gray tone as background. A high-brightness contrast between the color dots and background is needed for robust feature recognition (adaptive thresholding), whereas the shading algorithm needs a reasonably bright fabric, so we meet both requirements with a medium gray tone. While this is a decent choice, the background turned out darker than expected after printing. In retrospect, we would choose a brighter shade of gray to simplify the shading map extraction.

The dot spacing is 3.2 cm and the diameter is 2.1 cm which is a compromise between

Figure 5.10: Shadows are preserved in our renderings. They appear softer because we regularize the solution. The shadow contrast is higher than in Fig. 5.7 (d) because the shading map is rescaled during rendering.

high sampling rate of the surface and sufficient dot size in the image when capturing a whole person. Digital printing makes it easy to design such a pattern and send it to a company specialized on fabric printing. The garments are manufactured by a professional tailor for the recorded person.

All experiments are performed on a Pentium IV 3.2 GHz with 2 GB RAM. The average computation time for the automatic processing steps (Fig. 5.2) in our unoptimized MATLAB implementation for a $1004x1004$ video frame is 60 seconds (45 seconds for the shading map). Selected algorithms are implemented in C++: labeling, RBF evaluation, bilinear texture lookup and optical flow. Fig. 5.9 shows the accuracy of our texture interpolation algorithm. The corners of the overlayed checkerboard texture lie on the geometric centers of the color dots, although the RBF approximation has a smoothing effect. At the garment borders, the parameterization is less accurate because fewer features are detected (Fig. 5.9, right example). The temporal smoothness of the result can be assessed in the accompanying video. [1] We obtain realistic shading maps which preserve shadows and the shading of cloth folds (Fig. 5.7, 5.10-5.15). While our pattern

[1] `http://www.mpi-inf.mpg.de/~vscholz/egsr06/texturereplace.wmv`

cannot capture fine folds, the shading maps contain this information. The catwalk sequence (Fig. 5.12-5.13) shows also the robustness of feature recognition against lighting changes (the garment is rather dark in the beginning of the sequence). Two sequences show fast jumping motion to validate the feature tracking ability (Fig. 5.14).

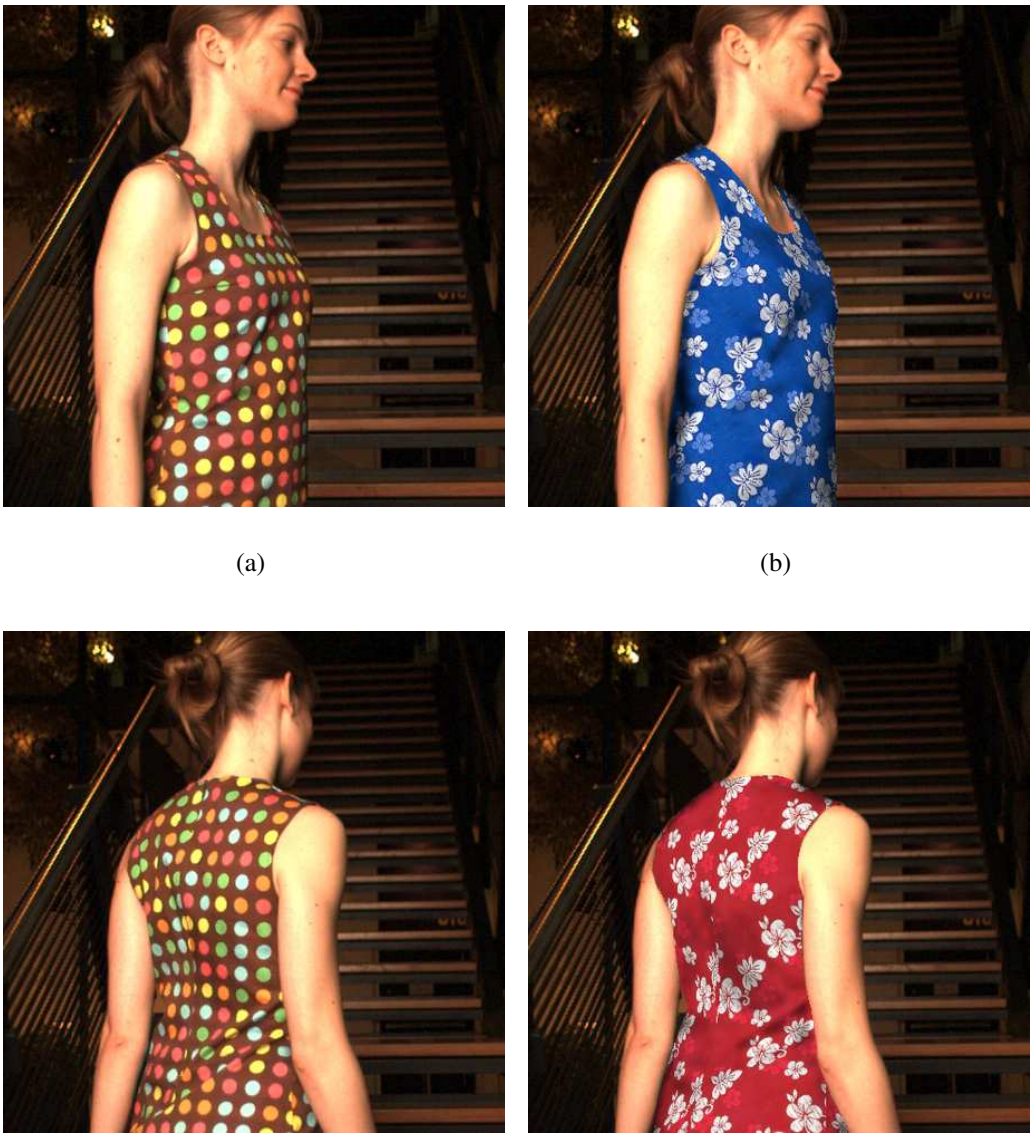
One limitation of our method is that video segmentation still requires user interaction. This is a notoriously difficult problem (e.g. due to shadows adjacent to the garment borders) where most automatic approaches require manual correction for an accurate result. However, segmentation is not the main focus of our work. Our RBF model handles discontinuities at self-occlusions only in an approximate way (the discontinuities are smoothed). For very loose garments the results might not be visually satisfactory in this case. In our experiments self-occlusion due to folding is barely observable because the dots are quite far apart. Self-occlusions between different garment panels however (e.g. between two legs for trousers) are not a problem as the RBF fitting is done separately for every panel. Our texture maps require spatiotemporal smoothing at the garment borders because feature detection is affected by foreshortening, especially when the overall feature size is small (full person capture). This is an inevitable drawback of a monocular method. Still, our proposed method is able to replace the fabric texture with realistic deformation and lighting for a wide range of real-world scenes. It can robustly deal with deformation, fast motion, lighting changes and feature occlusion.

5.8 Conclusions

We have presented a system for automatic texture replacement of color-coded garments. Visually convincing replacement results are obtained by using 3D spatiotemporal RBF approximation. We also show that our shading maps can capture small details at cloth folds. A single-view method is more challenging than a multi-view approach but opens up new applications for video post-processing in TV and film production.

Figure 5.11: Replacement results with different patterns.

Figure 5.12: Catwalk sequence. We are able to replace the original, recorded color-dot texture on the apparel with arbitrary pattern designs. Note that feature recognition works also for oblique surface angles (right side, left back panel of the dress) and the lighting changes in the video when the actor comes closer to the camera.



(a) (b)

(c) (d)

Figure 5.13: Closeup zooms show the shading at cloth folds.

(a) (b)

(c) (d)

Figure 5.14: Our method can track fast motion because it can re-initialize at every video frame (see video).

(a) (b)

(c) (d)

Figure 5.15: Shading effects near wrinkles and cloth folds. The shading maps faithfully represent the main cloth folds.

6

Keyframe Editing of Video Objects

In this chapter, we extend the scope of our work from cloth editing to general video editing. We present a framework for interactive video editing, where our focus is on footage from a conventional camcorder. After introducing texture replacement methods in the last two chapters, we now explore further transformations and manipulations of video objects. By relying on image-based, spatio-temporal techniques, we do not need to recover 3D scene geometry. Our framework is capable of removing and inserting objects, object motion editing, non-rigid object deformations, keyframe interpolation, as well as emulating camera motion. For evaluation, we show how movie shots can be persuasively modified during post-processing.

Figure 6.1: Camera push motion towards the actor by differently scaling foreground and background.

6.1 Introduction

Digitally retouching photographs has become routine in the publishing industry. For still images, a number of professional editing products such as Photoshop exist, offering a wide variety of different manipulation techniques that leave little to ask for [BC02]. In contrast, retouching video recordings is still a tedious per-frame editing procedure. Existing software tools that modify the content of video footage, like Adobe's After Effects [Ado07] and Apple's Shake [App07], offer tools for matte extraction [CAC+02] and rotoscoping [AHSS04]. However, more advanced retouching operations like editing object motion and shape are not supported. Similarly, tools for object segmentation do not address the challenges of altering the motion and shape of objects in video footage, either [WTXC04, WBC+05, LSS05, CCBK06]. The potential applications of a powerful video editing framework are, nevertheless, imposing.

In this project we address the challenge of convincingly altering the content of real-world video footage. For example, a tool to post-process movie sequences enables removing any accidentally visible crew member or equipment. Another common flaw of many movies is missing continuity between film shots. Such annoyances are easily eliminated if scene objects can still be rearranged during post-processing. Finally, a powerful video post-processing framework gives the movie director additional artistic freedom to tell the visual story of a film during editing.

To enable sophisticated video editing operations on general, real-world footage, our processing framework consists of several modules. Instead of attempting to augment real-world scenes with synthetic 3D models, we opt for editing spatio-temporal video objects. Video segmentation is needed to select the objects which the user wants to alter. To fill in the holes left behind by (re)moved objects, we rely on a video inpainting algorithm. Finally, compositing is applied to authentically blend modified objects into inpainted video frames. Object shapes and motions are specified interactively at keyframes. Our video editing framework achieves visually convincing results while re-

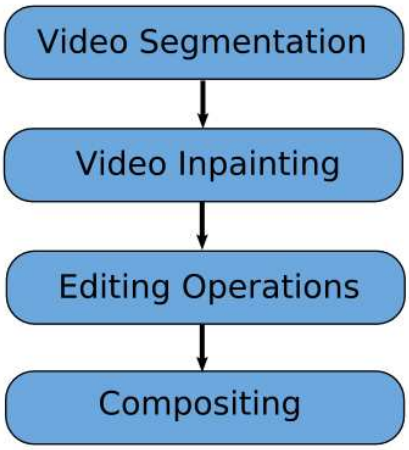


Figure 6.2: System overview. The segmentation and editing steps are interactive, while inpainting and compositing are automatic.

quiring only moderate user effort. Technical contributions of our framework include

1. a color-based video segmentation algorithm exploiting temporal coherence, including an improved boundary-editing user interface,
2. an efficient spatio-temporal background hole filling method that can handle fast camera motion,
3. a keyframe-based interpolation method for 2D object animation in video,
4. a new border matting method, and
5. various visual effects: editing of object trajectories, non-rigid deformation and emulation of camera motion.

We present the first video processing system that offers a wide variety of object transformation effects to the user where missing background inpainting is performed automatically. Our goal is an easy-to-use interactive system that leaves the user in full control of the editing result.

Figure 6.3: Result of segmentation during preprocessing. Left: input image, right: segmentation result, each colored region is a superpixel.

6.2 Overview

Our framework consists of several processing components (Fig. 6.2), which we describe in the following sections. The first task is interactive video segmentation for simple object selection (Section 6.3). As for still images, this interactive step is the key to any object editing operation. The second task is automatic video inpainting (Section 6.4), to fill in the holes in the video background that are left behind when the segmented objects are cut out and edited. In Section 6.5, we present the interactive editing operations of our framework, before in Section 6.6 we describe the final compositing step. We show how our system can be applied to edit camera motion, to modify object trajectories, and to apply non-rigid deformations (Section 6.7), before we conclude in Section 6.8.

6.3 Video Segmentation

In order to be able to separately edit individual objects in a video, the recorded video frames must first be segmented.

6.3.1 Preprocessing

In the preprocessing step we use the image segmentation algorithm by Felzenszwalb et al. [FH04b] in a batch procedure to create regions of homogeneous color, so-called superpixels, for each video frame (Fig. 6.3). The main idea of this algorithm is to merge the most similar pixels first in a pixel graph data structure. As merging criterion the current graph edge which connects two regions is compared with the internal color variation of the regions. The merging threshold can be steered by a parameter, which controls the coarseness of the segmentation result. A detailed description can be found in [FH04b]. This preprocessing reduces the amount of data for the subsequent steps and makes the following graphcut video segmentation feasible. Segmenting an image at VGA resolution takes a few seconds. Previous video segmentation systems [LSS05] used the watershed algorithm, which generates a larger amount of superpixels, or mean-shift segmentation [WBC+05], which requires more computational effort.

In order to group the computed 2D regions into 3D regions for video segmentation, we apply the same merging algorithm to a newly constructed 3D graph in a second step. Every superpixel region is connected with an edge to its superpixel neighbors in the same frame. We determine region adjacency by a contour following algorithm [SA85] for every superpixel and the spatially overlapping regions in the adjacent frames (Fig. 6.4, blue edges between different frames). We use a search radius of typically 25 pixels to connect a region in frame t to its neighbors in frames $t-1$ and $t+1$. A graph edge for every overlapping pair, weighted with the squared RGB color difference of the two regions, is added to the graph. Finally the region merging algorithm [FH04b] computes 3D regions from the 2D regions.

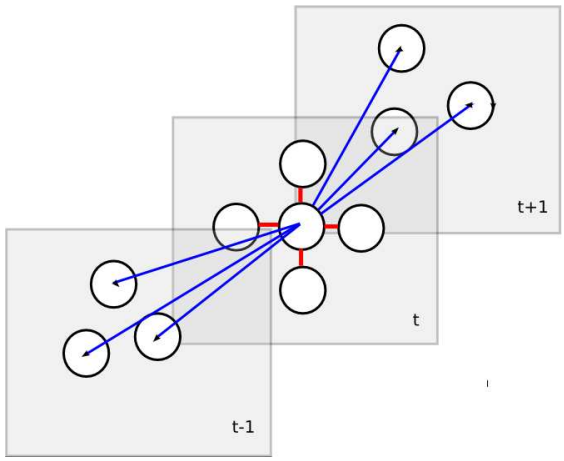


Figure 6.4: 3D graph for graphcut minimization. Every region is connected to neighbor regions within a frame (red edges E_1) and to neighbor regions in adjacent frames (blue edges E_2).

6.3.2 Min-cut Segmentation on Superpixels

After this data reduction step we apply a min-cut minimization [BJ01, LSS05] to the following energy function for all 3D regions $r_i \in V$ in the graph $G = (V \cup \{S,T\}, E_1 \cup E_2 \cup E_3)$ (Fig. 6.4):

$$E = \sum_{r_i \in V} E_D(x_i) + \lambda \sum_{(r_i,r_j)\in E_1} E_N(x_i,x_j) + \mu \sum_{(r_i,r_j)\in E_2} E_N(x_i,x_j) \tag{6.1}$$

In the graph G, the node set V is augmented with two terminal nodes S and T, which represent foreground and background labels respectively. Every node in V is connected with S and T (edge set E_3). The edge weights of E_3 are given by $E_D(x_i)$, the color likelihood for region r_i, a measure of conformity with foreground/background color models. The edge weights of $E_1 \cup E_2$ are given by the second and third term (neighborhood energy E_N). $x_i \in \{0,1\}$ denotes the background/foreground label of the region. A min-cut between the terminal nodes partitions the node set V in a foreground and a background region and minimizes the energy in Eq. 6.1. Typical values for the neighborhood energy weights are $\lambda = 20$ and $\mu = 10$. This energy function is similar to [LSS05, WBC$^+$05], but we use a different color model. We get color samples for fore-

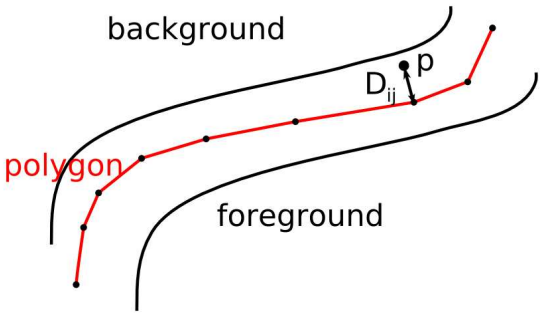


Figure 6.5: User-defined polygon (red) in unknown region and distance D_{ij} of pixel p to polygon.

ground and background regions from user input and build two color models from them by clustering them with the k-means algorithm. When k is large enough (a typical value is $k = 64$) we get more robust classification results than with standard Gaussian mixture models (GMM), where often a fixed number of mixture components is used (like in [LSS05, WBC+05]). We also avoid convergence problems of the EM algorithm which is normally used to fit the GMM. The cluster centers K_k^F and K_k^B from the k-means clustering are then used to compute the minimum distance from a region color c_i to foreground clusters $d_i^F = min_k \|c_i - K_k^F\|$ and similarly for the background $d_i^B = min_k \|c_i - K_k^B\|$.

$$E_D(x_i = 0) = \frac{d_i^F}{d_i^F + d_i^B} \tag{6.2}$$

$$E_D(x_i = 1) = \frac{d_i^B}{d_i^F + d_i^B} \tag{6.3}$$

is the color likelihood E_D [LSTS04], a proximity measure to cluster centers normalized by the sum of distances to foreground and background clusters. $E_N(x_i, x_j) = g(C_{ij})$ is the neighborhood energy E_N between neighboring regions [LSTS04], where $C_{ij} = \|c_i - c_j\|^2$ is the squared RGB color difference of the regions r_i and r_j, and $g(x) = \frac{1}{1+x}$ is a weighting function. We choose this neighborhood energy instead of the one proposed in [BJ01] as it has no additional parameter and gives slightly better segmentation results. The user marks foreground and background regions with some paint strokes

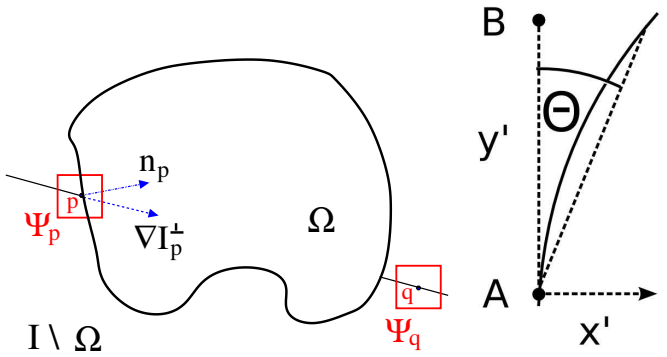


Figure 6.6: Left: Schematic overview of inpainting terminology, see text for details. Right: bending parameters and the result obtained from bending AB.

(markers) in keyframes which are treated as hard constraints, i.e. for these regions $E_D \in \{0,\infty\}$ [BJ01]. The marked pixels are also used as color samples for computing the color models. The min-cut algorithm [BK04] computes the labels x_i which minimize Eq. 6.1. The user can add additional strokes to refine the result, where only the color likelihood E_D of the affected regions is updated. By defining spatio-temporal slices of the video cube (x,y,t) [WBC+05] the user can also put markers in several video frames simultaneously which makes interaction more efficient.

6.3.3 Min-cut Refinement

To obtain fine segmentation on the pixel level, we employ the same graphcut technique in a corridor around the segmentation boundary obtained in the first step. A typical corridor width is 12 pixels. The min-cut graph $G = (V \cup \{S,T\}, E_1 \cup E_2 \cup E_3)$ for the corridor pixels V is constructed with a 10-neighborhood (8 spatial and 2 temporal neighbors) and edge weights according to Eq. 6.1. The diagonal edge weights are multiplied with a factor $\frac{1}{\sqrt{2}}$ which gives smoother segmentation boundaries [BJ01]. Typical parameter values for this step are $\lambda = \mu = 0.1$.

6.3.4 Boundary Editing Tool

The color-based segmentation algorithm fails at low contrast edges and the min-cut minimization does not preserve thin structures well. We provide two boundary editing tools for the user to correct these errors. First, the segmentation result is converted into boundary polygons with a contour following algorithm [SA85]. With the overriding brush as described in [LSTS04] the user can specify a corridor for the boundary with a paint stroke and min-cut optimization is used to find the actual boundary for one video frame. The neighborhood energy E_N is now defined differently. In addition to the color difference, it also uses the paint stroke (polygon) as a soft constraint in order to deal with low contrast edges. Similar to [LSTS04], the neighborhood energy for two pixels p_i, p_j with labels x_i, x_j is

$$E_N(x_i, x_j) = g((1-\beta)\cdot C_{ij} + \beta \cdot \eta \cdot g(D_{ij}^2)) \tag{6.4}$$

g is the weight function introduced earlier, D_{ij} is the distance of the center of the edge (p_i, p_j) to the polygon (Fig. 6.5) and η a scaling factor (typical value $\eta = 10$). By varying β the user can control the influence of D_{ij}, a typical value is $\beta = 0.5$. We drop the color likelihood term E_D in Eq. 6.1 as it is often misleading if the automatic segmentation fails and reduce the energy equation to

$$E = \sum_{(x_i,x_j)\in E_1} E_N(x_i, x_j) \tag{6.5}$$

which we minimize per video frame. The min-cut optimization is performed in a corridor around the whole segmentation boundary and only the paint stroke region is updated. We extend this image editing method to video. The user can put paint strokes at two different keyframes and the system interpolates the strokes linearly for the frames in between. We use a nearest neighbor matching algorithm to obtain vertex correspon-

dences for the two polylines. The boundary refinement is then computed for all frames so that the user only needs to edit keyframes. The segmentation result is temporally smooth for the interpolated frames.

Thin structures are difficult to obtain by min-cut minimization, so we also provide a user interface (UI) tool where the user can add vertices to the boundary polygons. The result of the video segmentation step is a binary alpha mask for each video frame. The different segmentation steps are summarized in Fig. 6.7.

6.4 Video Inpainting

We use two approaches for the hole filling problem: a texture synthesis algorithm and a mosaicking technique. This section describes work done by Sascha El-Abed during his master thesis [EA07].

6.4.1 Image Inpainting Revisited

We start with the patch-based image inpainting algorithm proposed by Criminisi et al. [CPT03]. Given an image I, the pixels of the hole Ω to be filled are given by a binary mask. Let p be a pixel located on the hole's boundary, and Ψ_p be a squared patch centered at p. To guide the filling order, a priority value $P(p)$ is computed for each pixel p on the hole's boundary:

$$P(p) = C(p) \cdot D(p), \tag{6.6}$$

where the $confidence$ term $C(p)$ and the $data$ term $D(p)$ are computed by

$$C(p) = \frac{\sum_{q \in \Psi_p \cap \bar{\Omega}} C(q)}{|\Psi_p|} \quad \text{and} \quad D(p) = \frac{|\nabla I_p^{\perp} \cdot n_p|}{\alpha}. \tag{6.7}$$

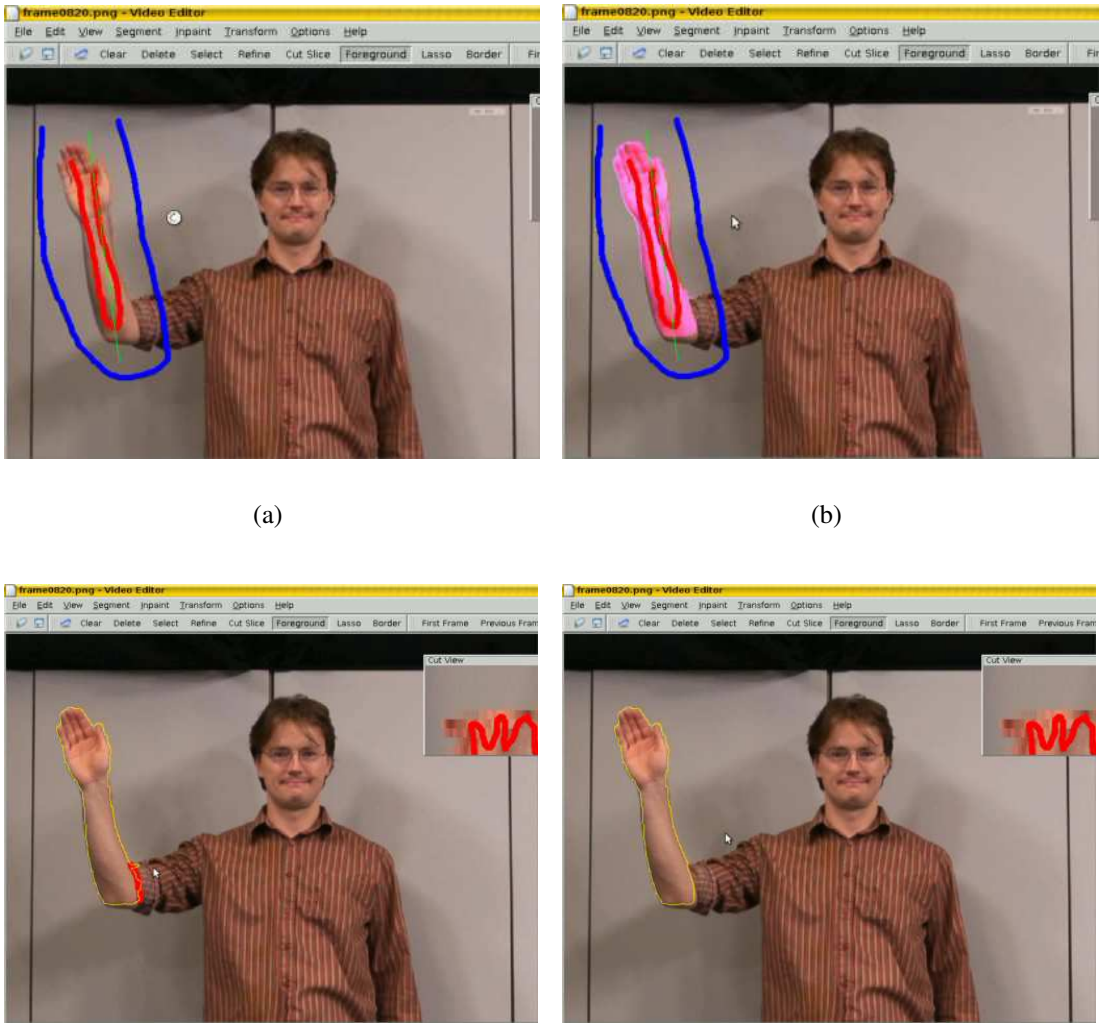


Figure 6.7: Segmentation steps. First row: User-defined paint strokes for foreground (red) and background (blue) (a) and segmentation result at the superpixel level (b). Second row: Pixel-level segmentation with red paint stroke near the T-shirt border which serves as input for our boundary editing tool (c), and result after correction (d).

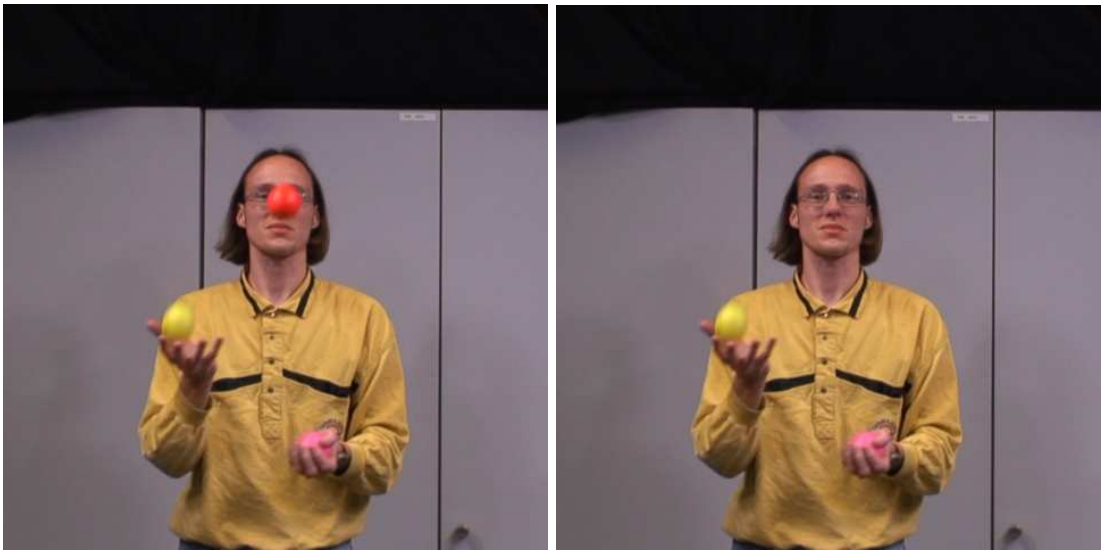


Figure 6.8: Video inpainting results. Left: Original frame. Right: inpainted frame. The missing facial texture was reconstructed from other frames.

$|\Psi_p|$ is the area of Ψ_p, α is a normalization factor, n_p is the unit vector orthogonal to the hole boundary at p and $\nabla I_p^{\perp}$ is the isophote direction at p, see also Fig. 6.6 (left). $C(p)$ is initialized to $C(p) = 0$ for $p \in \Omega$ and $C(p) = 1$ for $p \in I \backslash \Omega$. This priority-based filling aims at propagating linear structures in the texture into the hole region. For the pixel with the highest priority, the region $\bar{\Omega} = I \backslash \Omega$ is searched for a patch Ψ_q that is most similar to Ψ_p in terms of a simple sum-of-squared differences (SSD) error measure. Finally, the missing pixels of Ψ_p are replaced with pixels from Ψ_q; this is repeated until all pixels of the hole are filled in.

6.4.2 Video Inpainting with Spatio-temporal Patches

To solve the problem of inpainting holes in a spatio-temporal video cube, we extend this idea by using 3D spatio-temporal patches. The main motivation behind this is to achieve temporal coherence of the inpainting result, a typical patch size is $9x9x3$. Analogously to the pixels in the 2D case, a priority value is computed for each voxel $p = (x, y, t)$ on

the spatio-temporal hole's boundary as in Eq. 6.6. For the *data* term, we compute

$$D(p) = \frac{|\nabla I_p \times n_p|}{\alpha} \tag{6.8}$$

where α is a normalization constant, ∇I_p is the spatio-temporal gradient vector of the video cube $I(x,y,t)$ and n_p is the normal vector of the spatio-temporal hole at p. We compute n_p using the 3D structure tensor J:

$$J = \begin{pmatrix} M_x^2 & M_x M_y & M_x M_t \\ M_x M_y & M_y^2 & M_y M_t \\ M_x M_t & M_y M_t & M_t^2 \end{pmatrix}, \tag{6.9}$$

where $M \in \{0,1\}$ is a binary representation of the spatio-temporal hole with M_x, M_y and M_t being the spatial and temporal derivatives, respectively. The eigenvector corresponding to the largest eigenvalue of J gives us the direction of the highest variance within the neighborhood of the considered voxel. This eigenvector equals the normal vector direction at this voxel, which is equivalent to the normal vector of the surface formed by the spatio-temporal hole.

The inpainting procedure remains the same as in the 2D case: taking the boundary voxel p with highest priority, we search for a spatio-temporal patch Ψ_q being most similar to the spatio-temporal patch Ψ_p centered at p in terms of an SSD error measure, and transfer only those voxel locations of Ψ_q being empty in Ψ_p. This is repeated until all voxels are filled in. The search for a matching patch Ψ_q for the patch Ψ_p is restricted to the spatial and temporal vicinity of the location of Ψ_p in order to speed up the search process. For the results shown here, we use a spatial range of 15 voxels and a temporal range of 8 frames. We perform a grid search on the video cube with spatial mesh size Δx and temporal mesh size Δt. From these samples, the n best matches are determined and refined by a second pixelwise search in the local neighborhood. The best matching result

is taken for inpainting. Although this might not necessarily find the optimal match, the huge speed-up justifies the small loss of quality. Typical parameter values are $\Delta x = 3$, $\Delta t = 2$ and $n = 10$. However, this search procedure relies on the assumption that the neighborhood of Ψ_p contains texture which should be used for inpainting. Alternatively, the user can also mark the search area and influence the quality of the inpainting result. Our SSD error measure is summed over all color channels (we use Lab color space) and is evaluated only for those voxels of the patch Ψ_p that are already filled in. To improve the matching quality, each voxel Ψ_p^i of the patch is assigned a weighting factor depending on its location within the spatio-temporal patch. Assigning a high weighting factor in the center of Ψ_p helps to find matching patches Ψ_q that have a similar color in the patch's center. On the other hand, high weights on the patch's corners penalize huge color deviations on the corners. To this end, we combine both approaches and set the weights such that each weighting factor is the maximum of Gaussian kernel functions centered at each patch corner as well as the patch's center. For simplicity, these Gaussians have an extent reaching over the whole patch. Finally, we normalize by dividing by the sum of weights and the number of voxels used for matching:

$$SSD(\Psi_p, \Psi_q) = \frac{\sum_{i \in F} w_i \cdot (\Psi_p^i - \Psi_q^i)^2}{|F| \cdot \sum_{i \in F} w_i} \tag{6.10}$$

Here, $F = \Psi_p \cap \bar{\Omega}$ is the set of all voxels in patch Ψ_p that are already filled in. To achieve temporal smoothness of the inpainted region Ω, we blend between old voxels that have already been filled and the voxels from the new patch. This patch blending is done by averaging between the old and the new voxel values. Fig. 6.8 shows a video inpainting result. The face behind the removed ball is reconstructed faithfully. Observe that merely copying the affected region from neighboring video frames would not give a satisfactory result, since the person is moving.

6.4.3 Camera Motion

Filling spatio-temporal holes in a video sequence with fast camera motion can also be handled with the approach described in the previous section, but this might introduce artifacts if the holes are large. In this case, for example in the *trampoline* sequence, we fill holes in the background using a background mosaic. For general camera motion we make the assumption that the background is planar. This is a good approximation if the distance between camera and background is large enough as in our example. In the first step, we use the RANSAC algorithm [HZ00] to estimate homographies H_n between all pairs of subsequent frames $n, n+1$ from Harris corner feature correspondences ($\mathbf{x_{n+1}} = H_n \cdot \mathbf{x_n}$ for image coordinates $\mathbf{x_n}, \mathbf{x_{n+1}}$). The homographies are estimated with the normalized Direct Linear Transform (DLT) algorithm [HZ00]. The products $\Pi_{i=0}^{n} H_i^{-1}$ are used to project all frames onto the reference frame 0. We filter out moving objects by employing median filtering in the temporal domain. To obtain a globally accurately aligned mosaic, the homographies are refined by computing feature correspondences between each frame n and the common mosaic plane. The mosaic is then recomputed. If parts of the background are occluded throughout the whole sequence, unknown regions remain. These empty mosaic pixels are filled in using the image inpainting method of [CPT03]. Finally, we assign to each pixel of the spatio-temporal hole the color value of its corresponding pixel in the background mosaic.

6.5 Editing Operations

We provide a novel *keyframe-based interpolation* method for video object animation, similar to keyframe animation in 3D graphics. The user specifies a rigid transform (translation, rotation and scale) of the video object at keyframes interactively and the system interpolates the transformations of the object pixels for the remaining video frames.

Figure 6.9: Editing results from our bending operator. Left: original frame. Right: the forearm was edited by applying the bending operator.

6.5.1 Translation

To preserve the characteristics of the original motion, we warp the motion by a smooth offset function. For the case of *translation*, the object trajectory $\mathbf{p}(t) = (p_x, p_y)$ is warped to $\mathbf{p}'(t) = \mathbf{p}(t) + \mathbf{d}(t)$. $\mathbf{d}(t)$ is a 2D natural cubic spline [BBB98], a C^2-continuous curve which interpolates the user constraints $\mathbf{d}(t_0), \ldots, \mathbf{d}(t_k)$ at keyframes. At the curve endpoints, the second derivative is set to zero. Linear interpolation would not give satisfactory results in most cases. While Kochanek-Bartels splines [KB84] with more manipulation options could be used, we found that interpolation with natural cubic splines was satisfactory in our examples.

6.5.2 Scaling and Rotation

For uniform *2D scaling and rotation* we have to interpolate one transformation parameter (scaling factor and rotation angle, respectively). We use again a $2D$-spline, where the first coordinate corresponds to frame time t and the second coordinate to the transformation parameter. Also, the scaling and rotation center is user-defined and interpolated

in the same way. For *mirroring* objects vertically or horizontally a negative scale factor is used.

6.5.3 Non-rigid Deformation

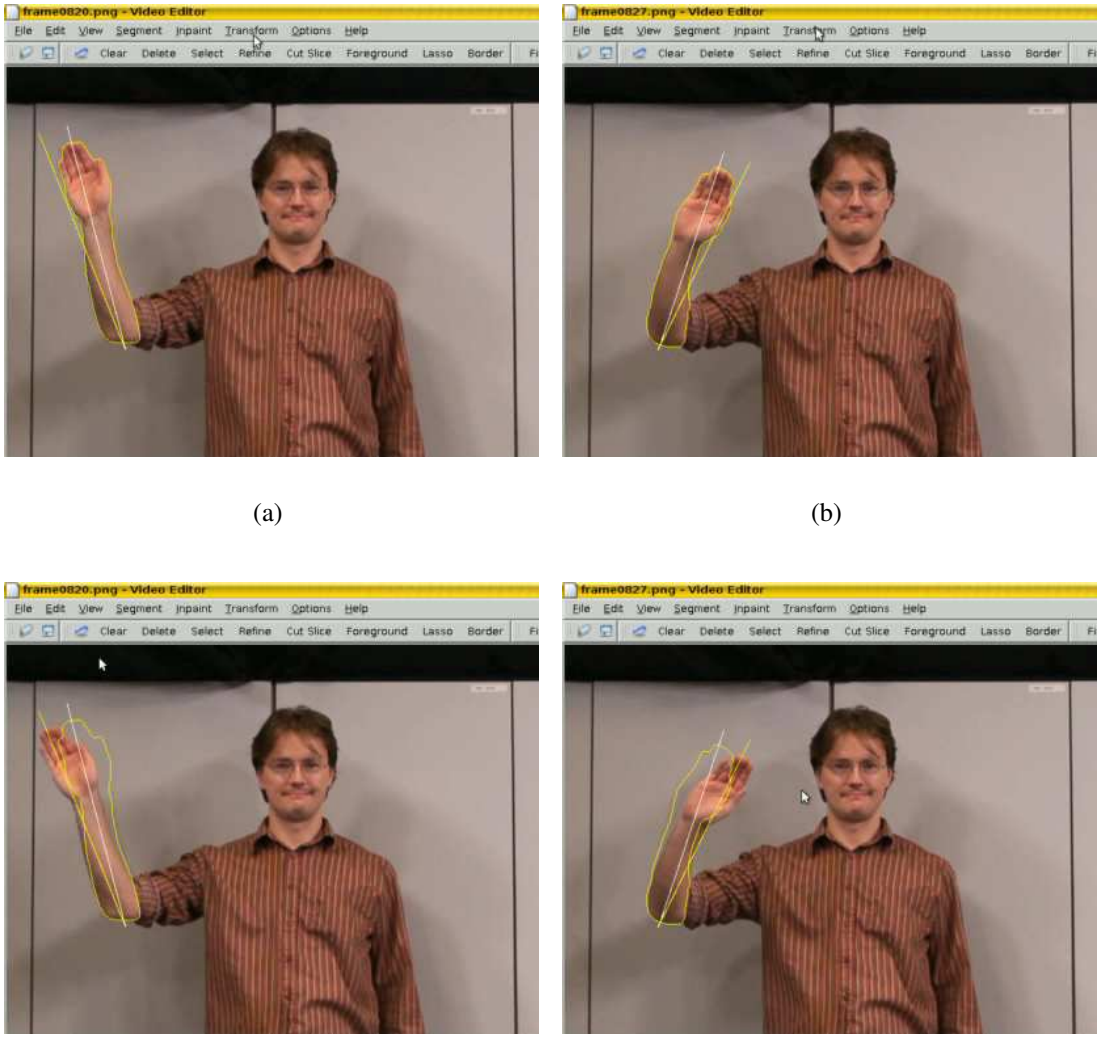
As an example for *non-rigid deformation*, our system contains a bending operator. It can be used to bend an articulated structure, e.g. the forearm in the *waving* example (Fig. 6.9). Our use of this image editing operation [BC02] for video is novel. The bending transform contains three user-specified parameters, two points A and B defining the y-axis of a local coordinate system (x', y') centered at A and a bending angle θ (Fig. 6.6, right). A and B are interpolated with a separate 2D cubic spline between keyframes. For interpolating θ we augment it again with the time t. The bending operation essentially rotates the object, but with a linear decreasing attenuation factor $a(y') = c \cdot y'$ [BC02] towards the pivot point A. The rotation is applied to all pixels of the selected object with $y' \geq 0$:

$$(x'', y'') = \begin{pmatrix} cos(a \cdot \theta) & -sin(a \cdot \theta) \\ sin(a \cdot \theta) & cos(a \cdot \theta) \end{pmatrix} \begin{pmatrix} x' \\ y' \end{pmatrix} \tag{6.11}$$

Fig. 6.10 shows the user interface for the non-rigid bending operator.

6.5.4 Simulated Camera Motion

To emulate *camera motion*, we use the alpha mask from segmentation to construct foreground and background depth layers. We simulate two different camera motion effects. Since image disparity is inversely proportional to object depth (parallax effect), we generate a *freeze-and-translate* effect by shifting foreground and background layers by different amounts. We apply this operation to a frame of the sequence and interpolate user-specified shift vectors $\mathbf{dx_B}$ of the background at keyframes with a cubic spline. The foreground shift is then $\mathbf{dx_F} = \alpha \cdot \mathbf{dx_B}$ where the α parameter controls the paral-



(a) (b)

(c) (d)

Figure 6.10: Editing steps. First row: User-specified bending angles at keyframes (a), (b). Second row: Compositing result after applying the bending transform (c), (d).

lax effect. The second effect, camera motion along the z-axis of the camera, is depth-dependent scaling according to the pinhole equation $x' = f \cdot \frac{x}{z}$. We simulate this effect by linear interpolation of z, which results in different scaling factors for the foreground and background layers.

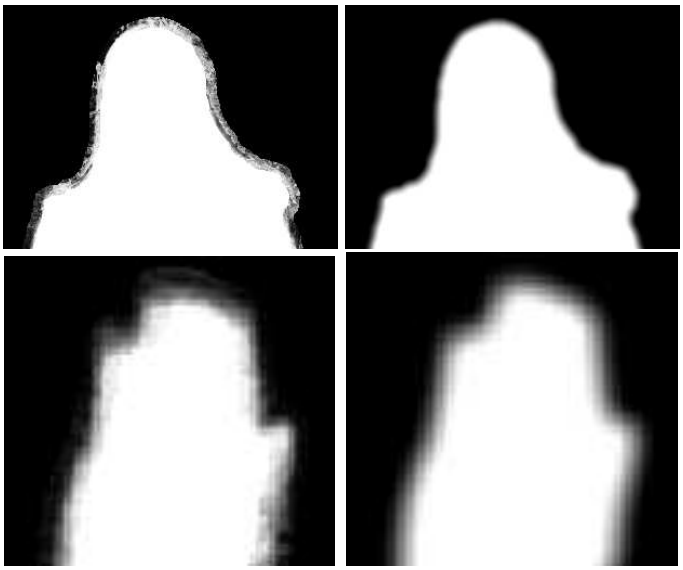


Figure 6.11: Results from our border matting algorithm. Left column: raw alpha mattes. Right column: regularized alpha mattes.

6.6 Compositing

Our segmentation results in binary alpha masks. To obtain a spatio-temporally coherent *alpha matte* with continuous alpha values, we first parameterize the boundary with a contour following algorithm. We then run a new border matting algorithm to generate smooth alpha mattes. We modify border matting [RKB04] in two important aspects. Border matting constructs a local foreground and background color model for each contour point by fitting a Gaussian to each of the distributions. In contrast, we use the color model from Section 6.3 to also handle the case when multiple colors are present. Then, we compute a raw alpha matte α_0 by using the right-hand side of Eq. 6.3 which is depicted in Fig. 6.11 (left). The original border matting algorithm defines a soft α-profile function along each normal of the contour which is fitted by dynamic programming. We use a different regularization approach by casting this as a surface fitting problem. An approximating thin-plate surface [Ter88] (cf. Section 5.6) is fitted to the raw alpha matte in a corridor Ω (typically 10 pixels wide) around the contour. In our experience, the smoothness of the matte can be controlled more easily this way, since no smoothness

parameters for adjacent profiles are needed. The sparse linear system

$$\alpha - \alpha_0 + \lambda \cdot \Delta^2 \alpha = 0 \tag{6.12}$$

is solved on Ω, where α is the resulting alpha matte, Δ^2 denotes the Bilaplacian operator (cf. Eq. 5.7) and λ is a regularization parameter. We use Dirichlet boundary conditions $\alpha = 0$ near the background and $\alpha = 1$ near the object on $\partial\Omega$ to constrain the solution and use the solver CHOLMOD [DH05] for sparse Cholesky factorization. The result of this regularization is shown in Fig. 6.11 (right).

The foreground object color can be estimated with the Bayesian matting algorithm [CCSS01]. The transforms obtained from keyframe interpolation (Section 6.5) are then applied to the original video object. We use bilinear interpolation for the pixel lookup. The transformed object is composited into the video inpainting result with alpha blending.

6.7 Results

The accompanying video shows video editing results for four sequences, *juggling*, *waving*, *girl* and *trampoline*. For the first three sequences we use a Sony HDR-HC3E camcorder recording in HDV format 1080i (frame size 1440x1080 interlaced, frame rate 25 fps). Each sequence contains 100 frames. To avoid artifacts from interlaced recording we downsample the images by dropping every second image row and resize the frames to VGA resolution (640x480 pixels). The fourth sequence *trampoline* is MPEG-2 compressed at 720x576 pixels and contains 50 frames. We present four different applications of our video editing system to demonstrate the capabilities of our system (Fig. 6.12-6.15). The quality of the editing results can best be assessed from the accompanying video. [1]

[1] `http://www.mpi-inf.mpg.de/~vscholz/eg07/keyframe_editor.avi`

(a) (b)

(c) (d)

Figure 6.12: Juggling sequence. The position of the **pink** ball was altered. (a) and (c): original frames, (b) and (d): edited frames.

Motion editing is shown in the *juggling* sequence (Fig. 6.12). The trajectory of the pink ball has been scaled. Only a few keyframes are used to generate the results. We place more keyframes at the upper part of the ball path to preserve acceleration effects. Notice that the edited motion looks as natural as if it had been recorded.

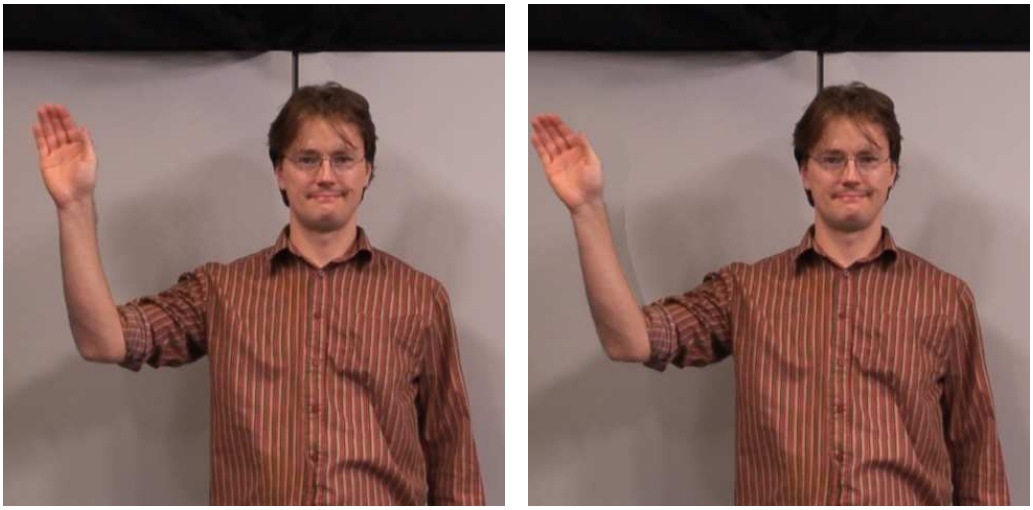
Non-rigid deformation is applied to the *waving* sequence (Fig. 6.13). The bending operator is applied to the forearm of the person so that the amplitude of the waving motion is exaggerated. Keyframes specifying the bending parameters at the turning points of the motion are sufficient. To avoid inpainting artifacts, the shirt region was removed from the search region by the user. Despite non-rigid deformations, the modified motion appears authentic.

The *trampoline* sequence contains considerable *camera motion*, motion blur, and additionally also exhibits strong MPEG-2 compression artifacts. Nevertheless, our inpainting algorithm is able to accurately fill in the background texture from the background mosaic. We apply a horizontal mirroring transform to reverse the rotation of the athlete (Fig. 6.14). Note that the background is not mirrored. To increase the accuracy of the motion estimation algorithm, the foreground object was masked with a uniform color to eliminate foreground motion.

Simulated camera motion is shown in the *girl* sequence (Fig. 6.15). To emphasize this effect, we freeze the video sequence from a handheld camera and apply a parallax effect in the middle of the sequence. At the end of the sequence we simulate a camera push motion in z-direction towards the actor (Fig. 6.1). The visual difference to a simple zoom-in operation is apparent in the accompanying video.

All computations are done on a Pentium IV 3.2 GHz with 2 GB RAM. Depending on the amount of user interaction, total processing time for video segmentation takes between 5 and 30 minutes for the presented examples. After segmentation, video inpainting is the only off-line processing step. Our video inpainting algorithm performs automatically, taking 5 minutes for the *juggling* sequence and 20 minutes for the *waving* example. For the *trampoline* sequence, mosaic construction and inpainting takes 7 minutes. Interactive animation editing and compositing takes between 3-7 minutes while border matting takes 2 minutes. One current limitation of our system is the color-based min-cut segmentation which does not perform well if foreground and background have similar color

(a) (b)



(c) (d)

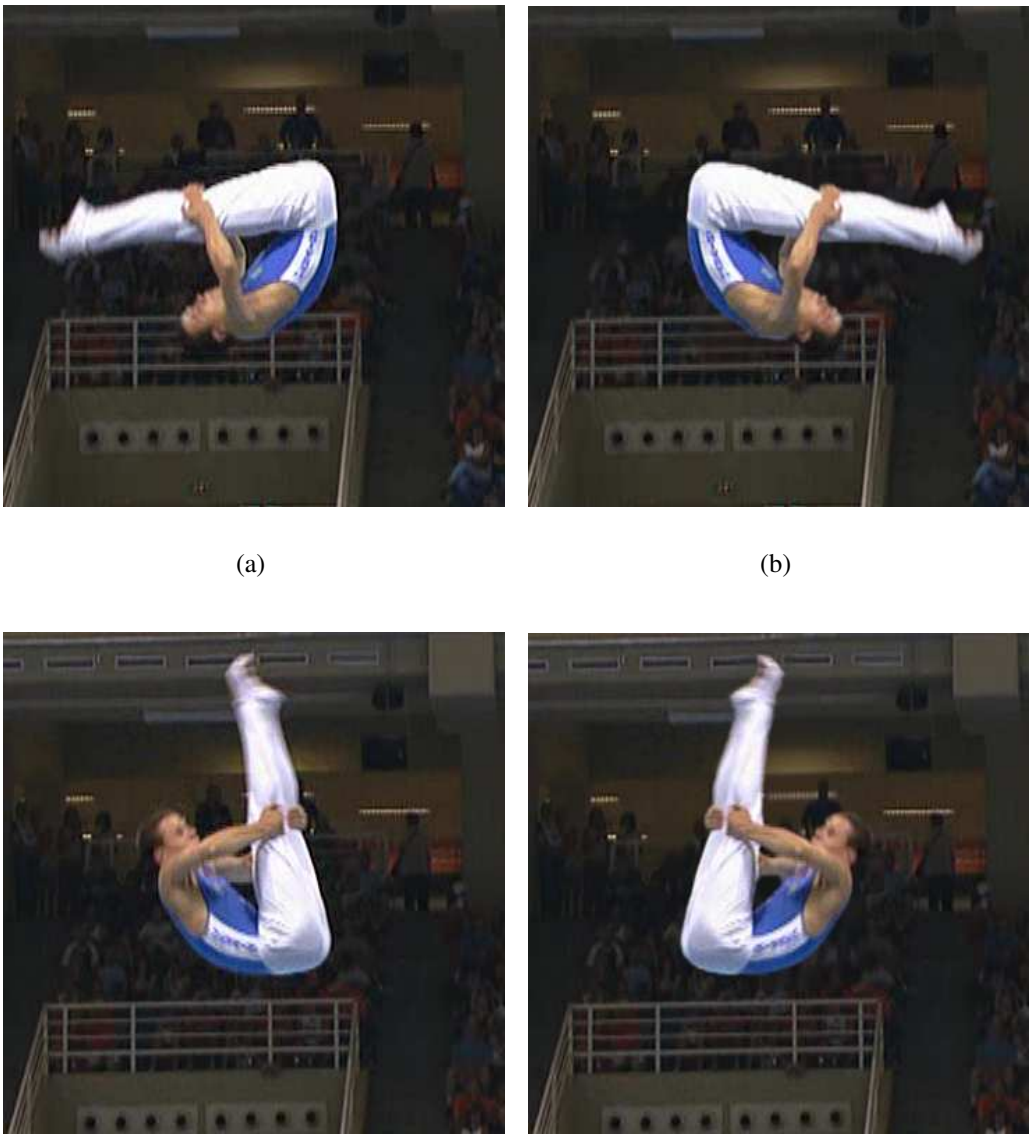
Figure 6.13: Waving sequence. The original motion was magnified by applying a non-rigid bending deformation to the forearm. (a) and (c): original frames, (b) and (d): edited frames.

distributions. Furthermore, thin structures are smoothed and must be corrected by the user. The goal of our boundary editing tool is to make user interaction as efficient as possible. Our method is generally limited to 2D editing, out-of-plane editing would require depth information. The mosaicking technique for inpainting assumes a static background and cannot preserve object motion in the background. The response times

of our user interface for the segmentation and editing steps can be evaluated from the accompanying video which shows that it is fast enough for interactive editing (a few seconds).

6.8 Conclusions

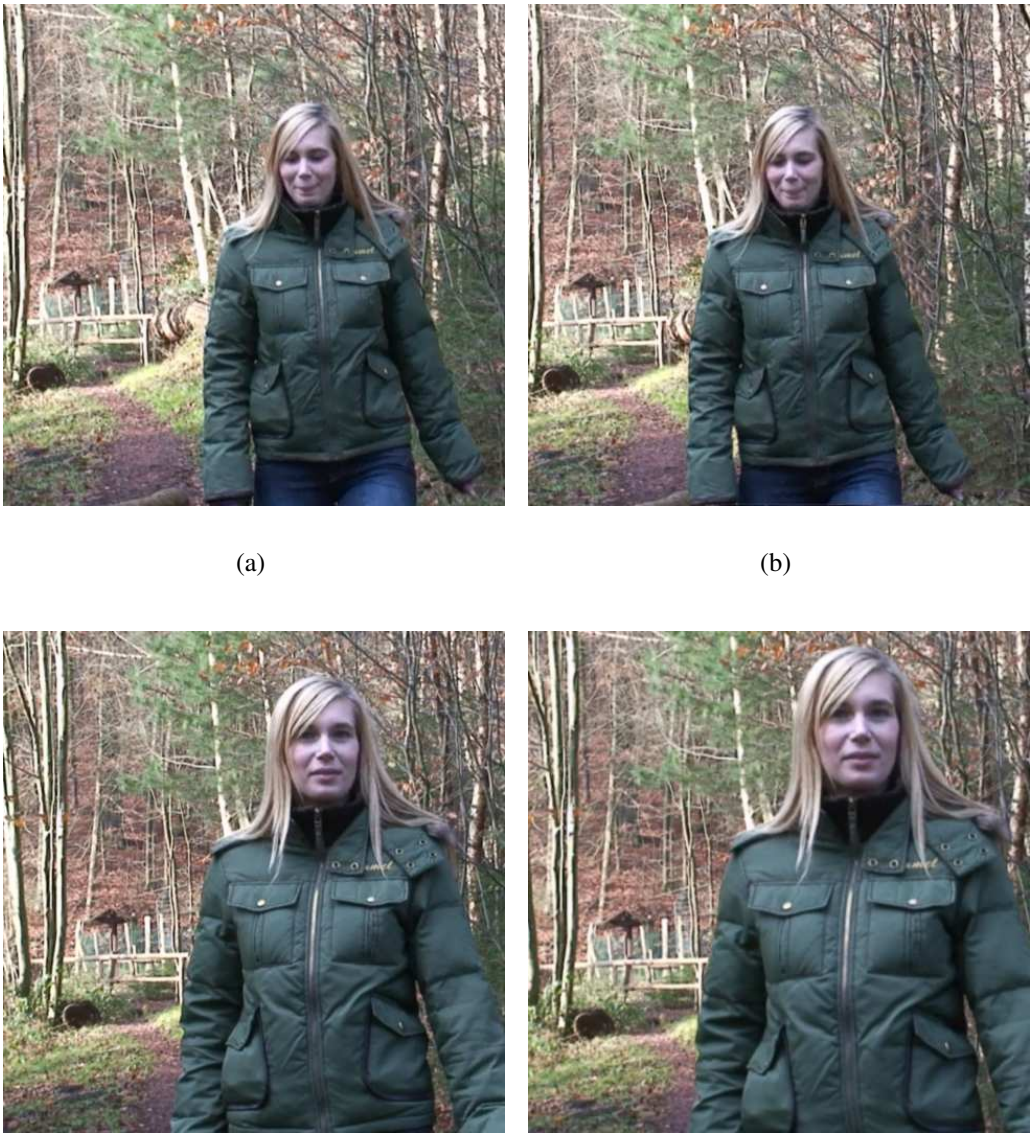
We have presented a system for video object segmentation, inpainting and keyframe animation of video objects. With our framework, a number of new video editing options become possible using conventional camcorder footage as input. While video object editing is conceptually simple, performing it without introducing objectionable artifacts requires considerable attention to detail. We have shown results for object motion editing, non-rigid object deformation and emulation of camera motion. Our system could be extended in several directions, for example object-based image filters or various painting effects. Furthermore, computing long-term pixel correspondences as described by Sand and Teller in their particle video system [ST06] would enable the user to modify pixel regions in one frame which are automatically tracked throughout the video sequence.



(a) (b)

(c) (d)

Figure 6.14: Trampoline sequence. The rotation of the athlete was reversed and the missing background was reconstructed. (a) and (c): original frames, (b) and (d): edited frames.



(a) (b)

(c) (d)

Figure 6.15: Simulating camera motion. (a) and (b): camera motion from left to right. (c) and (d): camera push motion towards the scene.

7

Conclusions

This thesis presents various contributions aimed towards editing video objects. We have presented a method that is capable of reconstructing cloth motion for cloth with richly detailed texture. A combination of optical flow and a deformable model is used to track motion robustly. We obtain photo-realistic results and can track motion over several hundred frames. Frame-to-frame coherence is achieved by our incremental approach using optical flow, which adds to the realism of the captured motion.

Furthermore, we have presented a system for acquiring garment motion in a multi-camera setup. By using color-coded textures, we can establish reliable point correspondences between different camera views. A prior triangle mesh model enables us to plausibly fill in missing data. Our method provides a surface parameterization that allows retexturing and rendering the dynamic surfaces realistically from arbitrary viewpoints.

We have also presented a novel approach to texture replacement of color-coded garments for monocular video. Visually convincing replacement results are obtained by using 3D spatiotemporal RBF approximation. We also show that our shading maps can capture small details at cloth folds. A single-view method is more challenging than a multi-view approach but opens up new applications for TV and film production.

Finally, with our video editing framework comprising segmentation, inpainting and keyframe animation, we can generate convincing visual effects such as motion editing, non-rigid deformations and simulation of camera motion. We rely on footage from

a single camera and can generate plausible visual effects. So far, we have only presented editing results which still look realistic, but generating scenes which are virtually impossible would be just another application of our system. The ability to make up such scenes was discovered early by the first filmmakers.

To sum up, the key contributions of this thesis are:

- A method for 3D tracking of cloth motion by optical flow in a multi-camera setting.
- The first system for multi-camera capture of garment motion that uses a color-coded pattern specially designed for robust observation.
- A video editing system for replacing cloth texture with texture deformation and lighting effects, which makes our color-coded approach useful for single camera recordings.
- The first system for keyframe editing of shape and motion of video objects which combines color-based object segmentation with video inpainting methods.
- A new algorithm for matting of video objects.
- A new, fast video inpainting method for static and moving cameras.

Since our work is based on video, videos naturally provide a better impression of the results than still images. The movies documenting the described projects can be found at the following project web pages. [1] [2] [3] [4]

[1] http://www.mpi-inf.mpg.de/~vscholz/vmv04/ClothMotion.html
[2] http://www.mpi-inf.mpg.de/~vscholz/eg05/GarmentMotionCapture.html
[3] http://www.mpi-inf.mpg.de/~vscholz/egsr06/TextureReplace.html
[4] http://www.mpi-inf.mpg.de/~vscholz/eg07/KeyframeEdit.html

7.1 Future Research

7.1.1 Cloth Capture

There are several areas for future work in cloth capture which are more or less engineering problems: increasing the pattern resolution to get fine scale folds, using more cameras to resolve occlusion problems, different garments, more specialized multi-camera calibration methods such as by Ihrke et al. [IAM04] etc. In research, new methods for editing and re-using the acquired deformable surfaces could be explored. Re-using the acquired garments by combining them with skeletal motion capture data is also an interesting research area [WCF07]. A reconstruction method for garments with normal texture (often uniform or with a periodic pattern) would be desirable, since this would not require special garments. While image-based 2D editing methods for this kind of textures exist, a full 3D surface reconstruction remains challenging.

7.1.2 Texture Replacement

A better automation of the segmentation step would improve the usability of our system and is mainly an engineering problem. Some amount of user interaction, however, will always be required for video segmentation without prior knowledge about the scene. Currently, the performance bottleneck in our implementation is the computation of the shading maps. Using faster solvers (e.g. multigrid) would improve our system's time and memory consumption. In research, considering discontinuities in the interpolation of texture maps would be an improvement over our current RBF approach. By detecting lines of discontinuity in the images, this information could be used in the fitting algorithm. Existing texture editing methods for uniform [FH06] and periodic textures [LL05] could be extended in several directions. More automation and explicit handling of several garment patches would be desirable.

7.1.3 Video Editing

On the engineering side, our system could be improved by a segmentation method which requires less user interaction. On the research side, we would like to give the user more possibilities to edit the results of automatic video inpainting. There are open issues for the video hole filling problem, such as video sequences with camera zoom and objects changing size. In these cases, more elaborate search models are needed. Furthermore, the introduction of 3D Television in the near future should make acquisition equipment for full 3D scenes more accessible. A stereo camera would be able to provide additional depth information, offering even more editing possibilities for video. An additional depth cue would facilitate tasks such as the segmentation and editing of camera motion.

Appendix A

Publications

The work presented in this thesis was published in the following papers.

[1] Volker Scholz, Sascha El-Abed, Marcus Magnor and Hans-Peter Seidel. *Keyframe Editing of Video Objects.* Under review.

[2] Volker Scholz and Marcus Magnor. *Texture Replacement of Garments in Monocular Video Sequences.* In: Rendering Techniques '06 (Proc. of the 17th Eurographics Symposium on Rendering), Nicosia, Cyprus, pp. 305-312, 2006.

[3] Volker Scholz, Timo Stich, Michael Keckeisen, Markus Wacker and Marcus Magnor. *Garment Motion Capture Using Color-Coded Patterns.* In: Computer Graphics Forum (special issue Eurographics 2005), vol. 24, no. 3, pp. 439-448, August 2005.

[4] Volker Scholz, Timo Stich, Michael Keckeisen, Markus Wacker and Marcus Magnor. *Garment Motion Capture Using Color-Coded Patterns. ACM SIGGRAPH Sketches and Applications*, August 2005.

[5] Volker Scholz and Marcus A. Magnor. *Cloth Motion from Optical Flow.* In: Proc. Vision, Modeling and Visualization 2004, Stanford, USA, pp. 117-124, November 2004.

Bibliography

[Ado07] Adobe Systems Inc. `http://www.adobe.com`, 2007.

[AHSS04] Aseem Agarwala, Aaron Hertzmann, David Salesin, and Steven M. Seitz. Keyframe-based tracking for rotoscoping and animation. *ACM Transactions on Graphics (Proc. of ACM SIGGRAPH 2004)*, 23(3):584–591, 2004.

[App07] Apple Computer, Inc. `http://www.apple.com`, 2007.

[Aut07] Autodesk Inc. `http://www.autodesk.com`, 2007.

[BBB98] R. H. Bartels, J. C. Beatty, and B. A. Barsky. *An Introduction to Splines for Use in Computer Graphics and Geometric Modelling*. Morgan Kaufmann, 1998.

[BC02] William A. Barrett and Alan S. Cheney. Object-based image editing. In *Proc. of ACM SIGGRAPH 2002*, pages 777–784, 2002.

[BFB94] J.L. Barron, D.J. Fleet, and S. Beauchemin. Performance of optical flow techniques. *International Journal of Computer Vision*, 12(1):43–77, 1994.

[BJ01] Yuri Boykov and Marie-Pierre Jolly. Interactive graph cuts for optimal boundary and region segmentation of objects in n-d images. In *Proc. IEEE International Conf. on Computer Vision*, pages 105–112, 2001.

[BK04] Yuri Boykov and Vladimir Kolmogorov. An Experimental Comparison of Min-Cut/Max-Flow Algorithms for Energy Minimization in Vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(9):1124–1137, 2004.

[Bou00] Jean-Yves Bouguet. Pyramidal Implementation of the Lucas Kanade Feature Tracker. `http://www.sourceforge.net/projects/opencvlibrary`, 2000.

[Bou05] Jean-Yves Bouguet. Camera Calibration Toolbox for MATLAB. `http://www.vision.caltech.edu/bouguetj/calib_doc/`, Aug 2005.

[BR04] D. Bradley and G. Roth. Augmenting Non-Rigid Objects with Realistic Lighting. Technical Report NRC 47398, National Research Council Canada, Institute for Information Technology, 2004.

[Bra01] Matthew Brand. Morphable 3d models from video. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (2)*, pages 456–463, 2001.

[BRB+04] Andrew Blake, Carsten Rother, M. Brown, Patrick Pérez, and Philip H. S. Torr. Interactive Image Segmentation Using an Adaptive GMMRF Model. In *Proc. European Conference on Computer Vision (1)*, pages 428–441, 2004.

[Bri03] R. Bridson. *Computational aspects of dynamic surfaces*. PhD thesis, Stanford University, 2003.

[BSCB00] M. Bertalmio, G. Sapiro, V. Caselles, and C. Ballester. Image inpainting. In *Proc. of ACM SIGGRAPH 2000*, pages 417–424, 2000.

[BTH+03a] K. S. Bhat, C. D. Twigg, J. K. Hodgins, P. K. Khosla, Z. Popovič, and Steven M. Seitz. Estimating Cloth Simulation Parameters From Video. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation* , pages 37–51, 2003.

[BTH+03b] K. S. Bhat, C. D. Twigg, J. K. Hodgins, P. K. Khosla, Z. Popovič, and Steven M. Seitz. Estimating cloth simulation parameters from video. In *SCA '03: Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer animation*, pages 37–51. Eurographics Association, 2003.

[CAC+02] Yung-Yu Chuang, Aseem Agarwala, Brian Curless, David Salesin, and Richard Szeliski. Video matting of complex scenes. In *Proc. of ACM SIGGRAPH*, pages 243–248, 2002.

[Can86] J. Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(6):679–698, 1986.

[CBC+01] Jonathan C. Carr, R. K. Beatson, J. B. Cherrie, T. J. Mitchell, W. Richard Fright, B. C. McCallum, and T. R. Evans. Reconstruction and representation of 3D objects with radial basis functions. In *Proceedings of ACM SIGGRAPH 2001*, pages 433–442, 2001.

[CCBK06] Antonio Criminisi, G. Cross, A. Blake, and Vladimir Kolmogorov. Bilayer Segmentation of Live Video. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, pages 53–60, 2006.

[CCSS01] Yung-Yu Chuang, Brian Curless, David Salesin, and Richard Szeliski. A bayesian approach to digital matting. In *Proc. of IEEE Conf. on Com-*

puter Vision and Pattern Recognition 2001, volume 2, pages 264–271, December 2001.

[CK01] Rodrigo L. Carceroni and Kiriakos N. Kutulakos. Multi-View Scene Capture by Surfel Sampling: From Video Streams to Non-Rigid 3D Motion, Shape & Reflectance. In *Proc. IEEE International Conference on Computer Vision*, pages 60–67, 2001.

[CK02] Kwang-Jin Choi and Hyeong-Seok Ko. Stable but Responsive Cloth. In *Proc. SIGGRAPH '02*, pages 604–611, 2002.

[CM02] D. Comaniciu and P. Meer. Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(5):603–619, 2002.

[CPT03] Antonio Criminisi, Patrick Pérez, and Kentaro Toyama. Object removal by exemplar-based inpainting. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (2)*, pages 721–728, 2003.

[DH05] T. A. Davis and W. W. Hager. Row modifications of a sparse cholesky factorization. *SIAM Journal on Matrix Analysis and Applications*, 26(3):621–639, 2005.

[DM96] Douglas DeCarlo and D. Metaxas. The integration of optical flow and deformable models with applications to human face shape and motion estimation. In *Proc. of the 1996 Conference on Computer Vision and Pattern Recognition (CVPR '96)*, page 231, 1996.

[EA07] Sascha El-Abed. Hole filling in images and video sequences (to appear). Master's thesis, Universität des Saarlandes, 2007.

[Ebe03] D. H. Eberly. *Game Physics*. Morgan Kaufmann Publishers, 2003.

[EF01a] Alexei A. Efros and William T. Freeman. Image quilting for texture synthesis and transfer. In *Proceedings of ACM SIGGRAPH 2001*, pages 341–346, 2001.

[EF01b] Alexei A. Efros and William T. Freeman. Image quilting for texture synthesis and transfer. In *Proc. of ACM SIGGRAPH 2001*, pages 341–346, 2001.

[EKS03] O. Etzmuß, Michael Keckeisen, and Wolfgang Straßer. A Fast Finite Element Solution for Cloth Modelling. *Proc. Pacific Graphics*, pages 244–251, 2003.

[EL99] Alexei A. Efros and Thomas K. Leung. Texture synthesis by non-parametric sampling. In *Proc. IEEE International Conf. con Computer Vision (2)*, pages 1033–1038, 1999.

[ESD03] A. Ebert, J. Schädlich, and A. Disch. Innovative Retexturing Using Cooperative Patterns. In *IASTED International Conference on Visualization, Imaging and Image Processing (VIIP 2003)*, 2003.

[FDB92] Brian V. Funt, Mark S. Drew, and Michael Brockington. Recovering shading from color images. In *Proc. European Conference on Computer Vision*, pages 124–132, 1992.

[FDL04] Graham D. Finlayson, Mark S. Drew, and Cheng Lu. Intrinsic images by entropy minimization. In *Proc. European Conference on Computer Vision*, pages 582–595, 2004.

[FH04a] Hui Fang and John C. Hart. Textureshop: texture synthesis as a photograph editing tool. *ACM Transactions on Graphics (Proc. of ACM SIGGRAPH 2004)*, 23(3):354–359, 2004.

[FH04b] Pedro F. Felzenszwalb and Daniel P. Huttenlocher. Efficient graph-based image segmentation. *International Journal of Computer Vision*, 59(2):167–181, 2004.

[FH04c] Vojtěch Franc and Václav Hlaváč. Statistical pattern recognition toolbox for matlab. Technical Report CTU–CMP–2004–08, Center for Machine Perception, Czech Technical University, 2004.

[FH06] Hui Fang and John C. Hart. Rototexture: Automated tools for texturing raw video. *IEEE Transactions on Visualization and Computer Graphics*, 12(6):1580–1589, 2006.

[FP02] David A. Forsyth and Jean Ponce. *Computer Vision: A Modern Approach*. Prentice Hall, 2002.

[GGW+98] Brian K. Guenter, Cindy Grimm, Daniel Wood, Henrique S. Malvar, and Frederic H. Pighin. Making Faces. In *Proc. SIGGRAPH*, pages 55–66, 1998.

[GKB03] Igor Guskov, Sergey Klibanov, and Benjamin Bryant. Trackable surfaces. In *SCA '03: Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer animation*, pages 251–257. Eurographics Association, 2003.

[GL96] G. H. Golub and C. F. Van Loan. *Matrix Computations*. The John Hopkins University Press, 1996.

[GMN+98] B. Galvin, B. McCane, K. Novins, D. Mason, and S. Mills. Recovering Motion Fields: An Evaluation of Eight Optical Flow Algorithms. In *Proceedings of the British Machine Vision Conference* , 1998.

[GMP+04] M. Gruber, C. Michel, S. Pabst, Markus Wacker, Michael Keckeisen, and S. Kimmerle. tcCloth - An interactive cloth modeling and animation system. *Proc. Graphiktag*, pages 361–372, 2004.

[Gus02] Igor Guskov. Efficient tracking of regular patterns on non-rigid geometry. In *International Conf. on Pattern Recognition (2)*, pages 1057–1060, 2002.

[GW02] Rafael C. Gonzalez and Richard E. Woods. *Digital Image Processing*. Prentice Hall, 2002.

[HAR+06] Nils Hasler, Mark Asbach, Bodo Rosenhahn, Jens-Rainer Ohm, and Hans-Peter Seidel. Physically based tracking of cloth. In *Proc. Vision, Modeling and Visualization (VMV)*, pages 49–56, 2006.

[HE00] D. H. House and D. E. Breen (Eds.). *Cloth Modeling and Animation*. AK Peters, Ltd., Natick, MA, 2000.

[HJO+01] Aaron Hertzmann, Charles E. Jacobs, Nuria Oliver, Brian Curless, and David Salesin. Image analogies. In *Proceedings of ACM SIGGRAPH 2001*, pages 327–340, 2001.

[HRA+07] Nils Hasler, Bodo Rosenhahn, Mark Asbach, Jens-Rainer Ohm, and Hans-Peter Seidel. An analysis-by-synthesis approach to tracking of textiles. In *Proc. of the International Workshop on Motion and Video Computing*, February 2007.

[HRS07] Nils Hasler, Bodo Rosenhahn, and Hans-Peter Seidel. Reverse engineering garments. In *Proc. Mirage 2005 - Computer Vision/Computer Graphics Collaboration Techniques and Applications*, pages 200–211, 2007.

[HS92] Robert M. Haralick and Linda G. Shapiro. *Computer and Robot Vision Volume I*. Addison Wesley, 1992.

[HTF01] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer, 2001.

[HZ00] Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2000.

[HZ05] Feng Han and Song-Chun Zhu. Cloth representation by shape from shading with shading primitives. In *CVPR '05: Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 1*, pages 1203–1210, 2005.

[HZ07] Feng Han and Song-Chun Zhu. A two-level generative model for cloth representation and shape from shading. *IEEE Transactions on Pattern Analysis and Machine Intelligence (To appear)*, 29, 2007.

[IAM04] Ivo Ihrke, Lukas Ahrenberg, and Marcus Magnor. External camera calibration for synchronized multi-video systems. *Journal of WSCG (International Conf. in Central Europe on Computer Graphics, Visualization and Computer Vision)*, 12(1-3):537–544, January 2004.

[Ihr07] Ivo Ihrke. *Reconstruction and Rendering of Time-Varying Natural Phenomena (to appear)*. PhD thesis, Universität des Saarlandes, 2007.

[Int01] Intel Corporation. *Open Source Computer Vision Library*, 2001.

[KB84] D. H. U. Kochanek and R. H. Bartels. Interpolating splines with local tension, continuity, and bias control. In *Proc. SIGGRAPH '84*, pages 33–41, 1984.

[KCVS98] Leif Kobbelt, Swen Campagna, Jens Vorsatz, and Hans-Peter Seidel. Interactive multi-resolution modeling on arbitrary meshes. In *SIGGRAPH '98: Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 105–114. ACM Press, 1998.

[KFW04] Michael Keckeisen, Matthias Feurer, and Markus Wacker. Tailor Tools for Interactive Design of Clothing in Virtual Environments. In *Proceedings of ACM Symposium on Virtual Reality Software and Technology (VRST)*, pages 182–185, 2004.

[KWT88] M. Kass, A. P. Witkin, and Demetri Terzopoulos. Snakes: Active contour models. *International Journal of Computer Vision*, 1(4):321–331, January 1988.

[Lau94] A. Laurentini. The visual hull concept for silhouette-based image understanding. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(2):150–162, 1994.

[LF04] Anthony Lobay and David A. Forsyth. Recovering Shape and Irradiance Maps from Rich Dense Texton Fields. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (1)*, pages 400–406, 2004.

[Lin05] Wen-Chieh Lin. *A Lattice-based MRF Model for Dynamic Near-regular Texture Tracking and Manipulation*. PhD thesis, Carnegie Mellon University, 2005.

[LK81a] B. D. Lucas and T. Kanade. An Iterative Image Registration Technique with an Application to Stereo Vision. In *Proceedings of the Seventh International Joint Conference on Artifical Intelligence* , pages 674–679, 1981.

[LK81b] B. D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *Proc. Seventh International Joint Conference on Artificial Intelligence*, pages 674–679, 1981.

[LL05] Wen-Chieh Lin and Yanxi Liu. NRT-based Texture Replacement in Real Videos. In *SIGGRAPH '05: ACM SIGGRAPH 2005 Sketches*, page 133, 2005.

[LL06] Wen-Chieh Lin and Yanxi Liu. Tracking dynamic near-regular textures under occlusion and rapid movements. In *Proc. European Conference on Computer Vision*, May 2006.

[LL07] Wen-Chieh Lin and Yanxi Liu. A Lattice-Based MRF Model for Dynamic Near-Regular Texture Tracking. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 29(5):777–792, 2007.

[LLH04] Yanxi Liu, Wen-Chieh Lin, and James Hays. Near-regular texture analysis and manipulation. *ACM Transactions on Graphics (Proc. of ACM SIGGRAPH 2004)*, 23(3):368–376, 2004.

[LLW06] Anat Levin, Dani Lischinski, and Yair Weiss. A closed form solution to natural image matting. In *CVPR '06: Proc. of the 2006 IEEE Conf. on Computer Vision and Pattern Recognition*, pages 61–68, 2006.

[Low04] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[LSS05] Yin Li, Jian Sun, and Heung-Yeung Shum. Video object cut and paste. *ACM Trans. on Graphics (Proc. of ACM SIGGRAPH 2005)*, 24(3):595–600, 2005.

[LSTS04] Yin Li, Jian Sun, Chi-Keung Tang, and Heung-Yeung Shum. Lazy snapping. *ACM Trans. on Graphics (Proc. of ACM SIGGRAPH 2004)*, 23(3):303–308, 2004.

[LTF+05] Ce Liu, Antonio Torralba, William T. Freeman, Fredo Durand, and Edward H. Adelson. Motion magnification. In *ACM Trans. on Graphics (Proc. of ACM SIGGRAPH 2005)*, pages 519–526, 2005.

[Met96] D. Metaxas. *Physics-Based Deformable Models Applications to Computer Vision, Graphics and Medical Imaging*. Kluwer, November 1996.

[MOC+98] Raymond A. Morano, Cengizhan Ozturk, Robert Conn, Stephen Dubin, Stanley Zietz, and Jonathan Nissanov. Structured Light Using Pseudorandom Codes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(3):322–327, 1998.

[MS97] S. Malassiotis and M. G. Strintzis. Model-Based Joint Motion and Structure Estimation from Stereo Images. *Computer Vision and Image Understanding*, 65(1):79–94, 1997.

[MTCK+04] N. Magnenat-Thalmann, F. Cordier, Michael Keckeisen, S. Kimmerle, R. Klein, and J. Meseth. Simulation of Clothes for Real-time Applications. In *Proc. of Eurographics, Tutorial 1*, 2004.

[MTVTW05] N. Magnenat-Thalmann, P. Volino, B. Thomaszewski, and Markus Wacker. Key techniques for interactive virtual garment simulation. In *Proc. of Eurographics, Tutorial 4*, 2005.

[OCDD01] Byong Mok Oh, Max Chen, Julie Dorsey, and Frédo Durand. Image-based modeling and photo editing. In *Proceedings of ACM SIGGRAPH 2001*, pages 433–442, 2001.

[PFTV92] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling. *Numerical Recipes in C*. Cambridge University Press, 1992.

[PH91] A. Pentland and B. Horowitz. Recovery of Nonrigid Motion and Structure. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(7):730–742, 1991.

[PH03] D. Pritchard and Wolfgang Heidrich. Cloth Motion Capture. *Computer Graphics Forum (special issue Proc. Eurographics 2003)*, 22(3):263–272, 2003.

[PLF05a] J. Pilet, V. Lepetit, and P. Fua. Augmenting deformable objects in real-time. In *International Symposium on Mixed and Augmented Reality*, 2005.

[PLF05b] J. Pilet, V. Lepetit, and P. Fua. Real-time non-rigid surface detection. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, pages 822–828, 2005.

[Pri03] D. Pritchard. Cloth parameters and motion capture. Master's thesis, University of British Columbia, 2003.

[PSB05] K. Patwardhan, G. Sapiro, and M. Bertalmio. Video inpainting of occluding and occluded objects. In *Proc. International Conf. on Image Processing (ICIP) (2)*, pages 69–72, 2005.

[PSB07] K. A. Patwardhan, G. Sapiro, and M. Bertalmio. Video inpainting under constrained camera motion. *IEEE Transactions On Image Processing*, 16(2):545–553, Feb 2007.

[PSGM03] Jordi Pagès, Joaquim Salvi, Rafael García, and Carles Matabosch. Overview of Coded Light Projection Techniques for Automatic 3D Pro-

filing. In *Proc. IEEE International Conf. on Robotics and Automation (ICRA)*, pages 133–138, 2003.

[RKB04] Carsten Rother, Vladimir Kolmogorov, and Andrew Blake. ”GrabCut”: interactive foreground extraction using iterated graph cuts. *ACM Trans. on Graphics (Proc. of ACM SIGGRAPH 2004)*, 23(3):309–314, 2004.

[SA85] S. Suzuki and K. Abe. Topological structural analysis of digital images by border following. *Graphical Models and Image Processing (CVGIP)*, 30(1):32–46, 1985.

[SJTS04] Jian Sun, Jiaya Jia, Chi-Keung Tang, and Heung-Yeung Shum. Poisson matting. *ACM Transactions on Graphics (Proc. of ACM SIGGRAPH 2004)*, 23(3):315–321, 2004.

[SM04] Volker Scholz and Marcus A. Magnor. Cloth Motion from Optical Flow. In *Proc. Vision, Modeling and Visualization 2004*, pages 117–124, Stanford, USA, November 2004.

[SM06a] Volker Scholz and Marcus Magnor. Garment texture editing in monocular video sequences based on color-coded printing patterns. Research Report MPI-I-2006-5-003, Max-Planck-Institut für Informatik, Stuhlsatzenhausweg 85, 66123 Saarbrücken, Germany, April 2006.

[SM06b] Volker Scholz and Marcus Magnor. Texture replacement of garments in monocular video sequences. In *Rendering Techniques 2006 (Proc. Eurographics Symposium on Rendering EGSR)*, pages 305–312, 2006.

[SSK+05] Volker Scholz, Timo Stich, Michael Keckeisen, Markus Wacker, and Marcus Magnor. Garment motion capture using color-coded patterns. *Computer Graphics Forum (Proc. Eurographics EG ’05)*, 24(3):439–448, 2005.

[SSY+04] Heung-Yeung Shum, Jian Sun, Shuntaro Yamazaki, Yin Li, and Chi-Keung Tang. Pop-up light field: An interactive image-based modeling and rendering system. *ACM Transactions on Graphics*, 23(2):143–162, 2004.

[ST06] Peter Sand and Seth J. Teller. Particle video: Long-range motion estimation using point trajectories. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (2)*, pages 2195–2202, 2006.

[Tau95] Gabriel Taubin. A signal processing approach to fair surface design. In *Proc. SIGGRAPH '95*, pages 351–358, 1995.

[TB02] Lorenzo Torresani and Christoph Bregler. Space-time tracking. In *Proc. European Conference on Computer Vision (1)*, pages 801–812, 2002.

[Ter84] Demetri Terzopoulos. *Multiresolution computation of visible surface representations*. PhD thesis, MIT, Cambridge, MA, 1984.

[Ter88] Demetri Terzopoulos. The computation of visible-surface representations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 10(4):417–438, 1988.

[TFA05] Marshall F. Tappen, William T. Freeman, and Edward H. Adelson. Recovering intrinsic images from a single image. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(9):1459–1472, 2005.

[TH04] Lorenzo Torresani and Aaron Hertzmann. Automatic non-rigid 3d modeling from video. In *Proc. European Conference on Computer Vision (2)*, pages 299–312, 2004.

[TLR01] Yanghai Tsin, Yanxi Liu, and Visvanathan Ramesh. Texture replacement in real images. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (2)*, pages 539–544, 2001.

[TM91] Demetri Terzopoulos and D. Metaxas. Dynamic 3D Models with Local and Global Deformations: Deformable Superquadrics. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(7):703–714, 1991.

[TPBF87] Demetri Terzopoulos, John Platt, Alan Barr, and Kurt Fleischer. Elastically deformable models. In *SIGGRAPH '87: Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, pages 205–214, 1987.

[TYAB01] Lorenzo Torresani, Danny B. Yang, Eugene J. Alexander, and Christoph Bregler. Tracking and Modeling Non-Rigid Objects with Rank Constraints. In *IEEE Conf. on Computer Vision and Pattern Recognition (1)*, pages 493–500, 2001.

[VBR+99] S. Vedula, S. Baker, P. Rander, R. Collins, and T. Kanade. Three-Dimensional Scene Flow. In *Proceedings of the 7th International Conference on Computer Vision* , pages 722–729, 1999.

[VSA03] V. Vezhnevets, V. Sazonov, and A. Andreeva. A Survey on Pixel-Based Skin Color Detection Techniques. In *Proc. Graphicon 2003*, pages 85–92, 2003.

[Wah90] G. Wahba. *Spline Models for Observational Data*. SIAM, 1990.

[WBC+05] Jue Wang, Pravin Bhat, R. Alex Colburn, Maneesh Agrawala, and Michael F. Cohen. Interactive video cutout. *ACM Trans. on Graphics (Proc. of ACM SIGGRAPH 2005)*, 24(3):585–594, 2005.

[WC05] Jue Wang and Michael F. Cohen. An iterative optimization approach for unified image segmentation and matting. In *ICCV '05: Proc. of the Tenth IEEE International Conf. on Computer Vision*, pages 936–943, 2005.

[WCF07] Ryan White, Keenan Crane, and David A. Forsyth. Capturing and animating occluded cloth. *ACM Trans. on Graphics (Proc. of ACM SIGGRAPH 2007) (to appear)*, 2007.

[WF06] R. White and D.A. Forsyth. Retexturing single views using texture and shading. In *European Conference on Computer Vision*, volume LNCS 3954, pages 70–81. Springer, 2006.

[WFV06] Ryan White, David A. Forsyth, and Jai Vasanth. Capturing real folds in cloth. Technical report, EECS Department, University of California, Berkeley, 2006.

[Whi05] Ryan White. Capturing cloth. Master's thesis, University of California, Berkeley, 2005.

[Wik07] Wikipedia. George Méliès. `http://en.wikipedia.org`, 2007.

[WL00] Li-Yi Wei and Marc Levoy. Fast texture synthesis using tree-structured vector quantization. In *Proc. of ACM SIGGRAPH 2000*, pages 479–488, 2000.

[WLF05] Ryan White, Anthony Lobay, and David A. Forsyth. Cloth capture. In *SIGGRAPH '05: ACM SIGGRAPH 2005 Sketches*, 2005.

[Woo80] Robert J. Woodham. Photometric method for determining surface orientation from multiple images. *Optical Engineering*, 19(1):139–144, 1980.

[WSI04] Yonatan Wexler, Eli Shechtman, and Michal Irani. Space-time video completion. *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, 01:120–127, 2004.

[WSI07] Yonatan Wexler, Eli Shechtman, and Michal Irani. Space-time completion of video. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(3):463–476, 2007.

[WTXC04] Jue Wang, Bo Thiesson, Yingqing Xu, and Michael F. Cohen. Image and video segmentation by anisotropic kernel mean shift. In *Proc. European Conference on Computer Vision (2)*, pages 238–249, 2004.

[ZCS02] Li Zhang, Brian Curless, and Steven M. Seitz. Rapid Shape Acquisition Using Color Structured Light and Multi-pass Dynamic Programming. In *The 1st IEEE International Symposium on 3D Data Processing, Visualization, and Transmission*, pages 24–36, June 2002.

[ZFGH05] Steve Zelinka, Hui Fang, Michael Garland, and John C. Hart. Interactive material replacement in photographs. In *Proc. Graphics Interface*, pages 227–232, 2005.

[Zha99] Zhengyou Zhang. Flexible Camera Calibration by Viewing a Plane from Unknown Orientations. In *Proc. IEEE International Conference on Computer Vision*, pages 666–673, 1999.

[ZTCS99] Ruo Zhang, Ping-Sing Tsai, James Edwin Cryer, and Mubarak Shah. Shape from shading: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(8):690–706, 1999.