IPT & EGVE Workshop (2005)
R. Blach, E. Kjems (Editors)

Optical Tracking and Calibration of Tangible Interaction Devices

Arjen van Rhijn and Jurriaan D. Mulder

Center for Mathematics and Computer Science, CWI, Amsterdam, The Netherlands

Abstract

In this paper, a novel optical tracking and object calibration system is presented for the recognition and pose estimation of tangible interaction devices for virtual and augmented reality systems. The calibration system allows a user to automatically generate models of the relative positions of point-shaped markers attached to interaction devices, simply by moving them in front of the cameras. There are virtually no constraints on the shape of interaction devices. The tracking method takes the calibrated models as input, and recognizes devices by subgraph matching. Both the calibration and tracking methods can handle partial occlusion. Results show the proposed techniques are efficient, accurate, and robust.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Three-Dimensional Graphics and Realism]: Virtual reality I.4.8 [Image Processing and Computer Vision]: Stereo I.4.8 [Image Processing and Computer Vision]: Tracking

1. Introduction

It is essential that a VR system provides accurate, fast, and robust tracking to allow for smooth interaction with the environment. Optical tracking has proved to be a valuable alternative to tracking systems based on other technologies, such as magnetic, acoustic, gyroscopic, and mechanical. Advantages of optical tracking include that it is less susceptible to noise, it allows for many objects to be tracked simultaneously, and interaction devices can be lightweight and wireless. Optical tracking can be divided into vision based methods, where a scene is analyzed for known features by advanced image processing techniques, and marker based methods, where the features are artificially added to a scene in advance, and therefore image processing can be kept simple and efficient.

A common approach for marker based tracking methods is to illuminate the scene with infrared (IR) light, and to equip interaction devices with retroreflective markers. These markers reflect the IR light back into the cameras. By mounting IR filters in front of the cameras, the resulting images only contain white blobs. These can be efficiently located through simple image processing techniques. The tracking system uses a model that describes the locations of markers on an interaction device, and matches this model to the blobs in the camera images to detect the position and orientation, or *pose*, of the device. Based on these techniques, several tracking systems have been developed, both in the research community [RPF01, MvL02, Dor99] and as commercial systems, such as A.R.T. [ART] and Vicon [VIC].

An important advantage of optical tracking is that it allows for rapid prototyping of interaction devices. Since the tracking system only requires information about the locations of the markers on a device, users are able to construct their own interaction devices by equipping them with retroreflective markers and defining a model for the tracking system. However, creating such a model by hand is a tedious and error prone task, and is only possible for simple shaped objects.

In this work, we present a novel system for *model based optical tracking* and *automatic real-time object calibration*. The system supports:

- Calibration and tracking of multiple objects simultaneously;
- High framerates during both calibration and tracking;
- Handling of marker occlusion;
- Robustness against noise and spurious markers.

The system allows rigid objects of arbitrary shape to be calibrated and tracked. A user simply needs to equip the object

with retroreflective markers, and move the object in front of the cameras during the training or calibration stage. The system automatically calibrates a model of the object, which the tracking system uses to identify the object and determine its pose.

We have implemented and evaluated our optical object calibration and tracking techniques using the Personal Space Station (PSS), a near-field desktop VR/AR environment [MvL02] (see Figure 1). The PSS enables a user to interact with the environment, using tangible interaction devices. Its tracking system consists of cameras equipped with an IR filter and a ring of IR LEDs illuminating the scene. We use these cameras to calibrate and track custom objects of arbitrary shape equipped with retroreflective markers. Three example objects are shown in Figure 1.

(a) (b)

Figure 1: *(a) A prototype Personal Space Station. (b) Three custom interaction devices: a 7x7x7cm cubical object both containing 30 markers, a 7cm diameter spherical object containing 24 markers, and a toy car containing 38 markers.*

The system needs to address the following problems. First, the 3D marker locations must be determined from the 2D blobs in the camera images using *stereo correspondence*. These markers must be tracked through time using *frame-to-frame correspondence*. These steps are refered to as *marker tracking*. We propose a new robust stereo correspondence method, and show how frame-to-frame correspondence can be obtained using the same method.

Second, during *object calibration*, the system needs to determine a model of the 3D locations of the markers attached to an object. As an example, Figure 2 shows a cube and a toy car, along with the corresponding models as determined by our object calibration method. A model is described by a graph G, where a vertex represents a 3D marker, and an edge represents the (static) distance between two markers. An edge is only present if the two markers can be seen simultaneously. During calibration, the user rotates the device in front of the cameras to show the system all markers. Since not all markers may be visible at the same time due to occlusion caused by the object itself and by the user's hands, the system needs to detect reappearing markers. Furthermore,

(a) (b) (c)

Figure 2: *(a) Two example objects. The cube contains 30 markers, whereas the toy car contains 38 markers, (b) The corresponding models with the 3D marker locations, and (c) the complete model graphs, including edges.*

the system needs to determine which markers belong to a single rigid object. This allows a user to train multiple rigid objects simultaneously, and makes the system robust against other spurious markers within the tracking volume. The calibration system provides instant feedback to the user of the acquired object model.

Third, after calibration, the tracking system uses these object models to determine the pose of the devices. This *model based tracking* system uses a minimum subset of markers needed to unambiguously determine the pose. This minimum number of markers can be determined during calibration.

This paper is organized as follows. In Section 2 we review related work. Section 3 details the marker tracking stage. In Sections 4 and 5 we discuss the object calibration and model based tracking stages. Section 6 presents results of all stages of our system in practice. Section 7 provides a discussion of the pros and cons of our proposed calibration and tracking methods. Finally, in Section 8 conclusions are given.

2. Related work

We categorize related work in marker tracking, object calibration and model based object tracking.

2.1. Marker tracking

Marker tracking consists of two stages: determining 3D marker locations of the blobs in the camera images using stereo correspondence, and tracking these markers in time using frame-to-frame correspondence. Both problems are closely related, and can be seen as cases of the general feature correspondence problem.

Much research has been done on stereo correspondence. Many researchers have addressed the problem of dense stereo correspondence [SS02], dealing with large numbers

of features. A common approach is to match features based on the similarity of the surrounding pixels [PMG85, Pil97, BT98]. Pilu [Pil97] uses an elegant and simple algorithm to incorporate the uniqueness and similarity constraints into the matching process. He uses the correspondence method described by Scott and Longuet-Higgins [SLH91], and defines a similarity metric to match markers, based on the distance between the 2D feature locations in the camera images, and the correlation of the surrounding pixels. However, in the case of IR images, metrics as the correlation of surrounding pixels or marker characteristics are meaningless. We adapt this method for stereo correspondence of calibrated IR camera images, by including the epipolar constraint and defining a correlation function of surrounding markers in the rectified images. Using [SLH91] as a method to find a correspondence between two sets of points, we solve both stereo and frame-to-frame correspondence using the same method, by defining appropriate matching functions.

2.2. Object calibration

Obtaining a model of an object by moving it in front of the cameras is closely related to motion segmentation in long image sequences. A common approach is to track markers using a Kalman filter, and to group markers together if they have similar kinematic parameters. Zhang and Faugeras [ZF92] track 3D line markers and estimate their motion using an extended Kalman filter, grouping together markers with similar motion. Occlusion is handled by predicting the location of disappearing features using the Kalman filter. This assumes short-term occlusions. Smith [Smi95] uses a similar approach, but uses 2D point markers.

Mills [MN00] and Hornung [HSDK05] use an alternative approach. They maintain a graph, where each node represents a marker, and an edge represents a rigid relation between markers. As markers are moved, edges are updated, and when the distance between markers varies too much, the edge is deleted. The approaches differ in how occlusion is handled. However, both approaches suffer from problems which limit their applicability, which will be discussed in Section 4. Our approach shares ideas with the work of Mills, but is better suited to handle significant occlusion.

2.3. Model-based tracking

Model-based optical tracking methods for point markers can be divided into two categories: *pattern based* methods, and *point-cloud based* methods. The first category subdivides a model into small unique patterns, and tries to match a complete pattern with blobs in the camera images. Examples of such methods are described in [vLvR04]. Pattern-based methods are generally efficient in terms of computational and storage requirements, but fail to track an object when patterns are only partially visible.

Point-cloud based methods consider all points of a model as one whole, and match a subset of the model in the data. A well-known technique is geometric hashing [LSW88], which is based on a preprocessing stage to generate fast lookup tables. For each combination of three model points, a coordinate system is defined in which all remaining points are expressed. Next, all points in this system serve as addresses into a hashtable, which stores a pointer back to the model and the reference frame. During recognition, a combination of three data points is taken and a reference frame is determined. All other data points are expressed in this frame, and are used to address the hashtable to generate votes for a pair <model, reference frame>. If a model receives enough votes, it is marked as identified. Since the repeated transformation of data points into a reference frame can be computationally expensive, the efficiency of this method depends on the amount of candidates that need to be examined.

We propose a new point-cloud based tracking method, which is both fast and robust. The method is based on subgraph matching, and requires far less storage and preprocessing than geometric hashing, without having the drawbacks common to pattern based methods.

3. Marker tracking

To track markers in 3D, two subproblems need to be solved. First, stereo correspondence is used to determine the 3D marker locations of the 2D blobs. Second, frame-to-frame correspondence is used to track 3D markers through time. In the following section, we will discuss these techniques in more detail.

3.1. Stereo correspondence

The first step in our tracking system is to find the blobs in the camera images, corresponding to the markers attached to an interaction device. The location of these blobs can be found by simple image processing techniques. To determine the 3D locations of the 2D blobs, pairs of blobs in two (or more) camera images have to be found such that each pair corresponds to one device marker. This is known as the stereo correspondence problem. When correspondence has been established, the 3D location of each marker can be determined using triangulation.

Our approach to this problem is based on a more general method to determine the correspondence between two sets of points, which was introduced by Scott and Longuet-Higgins [SLH91]. Their method starts by defining a proximity matrix G of the two sets of points, where each element G_{ij} is a Gaussian-weighted similarity measure between points I_i and I_j. Next, a Singular Value Decomposition (SVD) $G = TDU^T$ is performed, and D is converted to a new matrix E by replacing all singular values with one. This matrix is used to compute a new matrix $P = TEU^T$. As shown in [SLH91], a one-to-one mapping between points I_i and I_j is found if P_{ij} is both the greatest element in its row and in its column.

Pilu [Pil97] uses the method of Scott and Longuet-Higgins to solve the stereo correspondence problem for uncalibrated cameras. He defines a similarity measure for

matching points (i, j) based on the correlation of the surrounding pixels and the distance of the 2D locations of the points in the camera images.

In our case, a correlation metric of surrounding pixels cannot be used, as we are dealing with IR images and round blobs with identical properties. Moreover, using the 2D distance of blob locations requires the disparity between blobs in both images to be small, or at least a known constant. However, we found that the disparity of a marker moving in the workspace of our desktop near-field virtual environment varies greatly with the position of the object.

We define a metric that exploits the epipolar and similarity constraints (the uniqueness constraint is automatically included in the method of Scott and Longuet-Higgins). Basically, for two points to match, they should lie on the same epipolar line, and their neighbouring points should be distributed similarly.

The epipolar constraint is included into the proximity matrix G_{ij} by

$$g_{ep}(i,j) = e^{-|P_i.y - P_j.y|^2/2\sigma_{ep}^2} \tag{1}$$

where P_i denotes the rectified image coordinates of point I_i, and σ_{ep} is a tuning parameter which should reflect the expected error in epipolar geometry. The similarity constraint is included by defining a region R of size S around the rectified points P_i and P_j. All points in the regions R_i and R_j are translated so that P_i and P_j are in O. Next, we calculate the mean of minimum distances as

$$d_{md}(S_1,S_2) = \frac{1}{N}\sum_{P\in S_1} \parallel P_i - \mathcal{C}_{cp}(P_i,S_2) \parallel \tag{2}$$

where S_1 denotes the smallest set of points, N denotes the size of set S_1, and S_2 is the larger set of points. The function $\mathcal{C}_{cp}$ is the closest point operator defined as

$$\mathcal{C}_{cp}(a,\zeta) = \arg\min_{x\in\zeta} \parallel x - a \parallel \tag{3}$$

The total proximity matrix is then given by

$$G_{ij} = g_{ep}(i,j)e^{-d_{md}(R_i,R_j)^2/2\sigma_{md}^2} \tag{4}$$

where σ_{md} should reflect the expected similarity error.

3.2. Frame-to-frame correspondence

Frame-to-frame correspondence can be obtained by application of the correspondence method of Scott and Longuet-Higgins [SLH91] with a different proximity metric. We use the Euclidian distance between 3D marker locations of the frames at time t and $t - \Delta t$:

$$G_{ij} = e^{-\|p_i(t) - p_j(t-\Delta t)\|^2/2\sigma_f^2} \tag{5}$$

where $p_i(t)$ is the 3D location of markers I_i at time t, and σ_f a parameter defining the expected error. To improve the robustness of the frame-to-frame correspondence in case of multiple objects moving independently and in case of fast movements, we include a simple linear prediction of each marker, such that σ_f can be chosen small.

4. Object calibration

Our calibration method for rigid interaction devices shares ideas with the method for motion segmentation in long image sequences by Mills [MN00]. However, it differs in the way occlusion is handled and how the model is maintained, making it more robust to long-term occlusion.

The basic idea is to use frame-to-frame correspondence to track markers through time, assigning each an age and a unique ID, and to maintain a graph $G = (V,E)$ where

- V is a set of vertices, where each vertex corresponds to a marker ID.
- $E \subseteq V \times V$ is the set of edges, where an edge E_{ij} represents the average Euclidian distance between vertices V_i and V_j. An edge E_{ij} is only present if during calibration the distance between the markers associated with vertices V_i and V_j remains static. In this case, the markers associated with V_i and V_j are said to have a *rigid relation*.

Initially, all visible vertices are connected. As markers are moved around, the Euclidian distance between each marker pair is examined and compared to the distance associated with the corresponding edge in the graph G. If the difference in distances exceeds a certain threshold, the edge is deleted. In order to deal with noise and measurement errors, we maintain a running average distance between markers over the last N frames, and compare this distance with the edge distance. This has the effect of making edges somewhat elastic.

Problems arise when markers enter the scene for which no frame-to-frame correspondence can be established, i.e. markers of age zero. A marker with age zero can be a new marker not yet part of the model, or a previously occluded marker that reappears. The system needs to distinguish between both cases, and in the case of a reappearing marker assign the original ID and age to this marker. New markers are assigned new IDs, added to G, and connected to all other visible markers.

4.1. Detecting reappearing markers

To detect reappearing markers, new markers need to be compared to occluded markers. A marker is new if has no frame-to-frame correspondence, i.e. its age is zero. Other (older) visible markers are refered to as identified points.

Hornung [HSDK05] detects reappearing markers by comparing the distances between new markers and identified markers, to the distances between occluded markers which have an edge to identified markers. When all distances match, the new marker is considered to be a reappearing marker. However, this approach fails for occluded markers which do not (yet) have an edge with the identified markers. Figure 3 illustrates the problem: when training the cube, a side may reappear with markers that have no edges with the markers on the side of the cube that is visible.

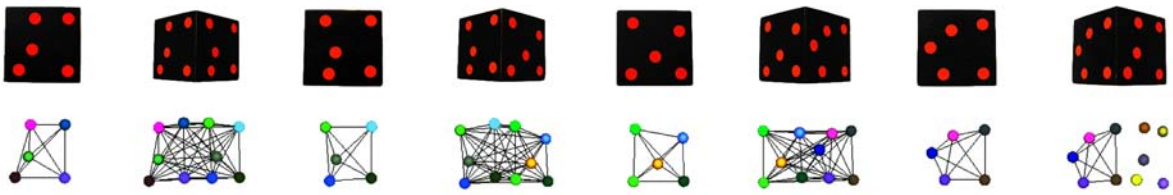


Figure 3: *A cube being rotated during calibration. The calibration system determines the relation between sides* $(1,2)$*,* $(2,3)$*, and* $(3,4)$*. When side* 4 *is visible and side* 1 *reappears, there is no direct relation between sides* 4 *and* 1*, and so matching the (reappearing) markers of side* 1 *directly to the other visible markers will fail.*

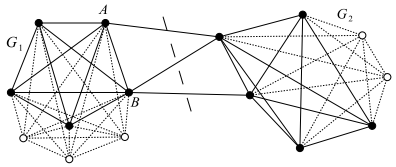


Figure 4: *A sample graph. Mills' triangle-based clustering would produce one rigid body, whereas our pyramid-based clustering produces subgraphs* G_1 *and* G_2*. Since rotations around axis AB have no effect on the graph, prediction of occluded markers (denoted by white dots) will be inaccurate if the complete graph is classified as rigid.*

A better approach to detect reappearing markers is to directly predict the location of occluded markers, as followed by Mills [MN00]. Mills maintains one graph G holding all data, and clusters this graph into rigid substructures. Rigid substructures form cliques in G. However, as clique finding in graphs is computationally expensive, Mills proposes a triangle-based clustering, where markers are only assigned to the same cluster if they are both part of a triangle with a shared edge. The clusters are then associated with objects, and a rigid body transform is computed to predict the location of occluded vertices. However, this method can falsely qualify structures as rigid. Consider for example the graph of Figure 4. As all triangles in this graph share edges with at least one other triangle, Mills' clustering method would produce one rigid body, while in reality the graph contains two rigid object graphs G_1 and G_2. Any rotation of G_1 around axis AB does not remove any edges in the graph, and therefore the complete graph is incorrectly classified as rigid. Since edges are somewhat elastic, this situation may occur quite frequently in practice.

Our approach maintains a graph for each detected object G_O, rather than one graph G for all data. This ensures that markers assigned to different objects are not reconnected at a later stage by new markers appearing, increasing both the accuracy and efficiency of occlusion prediction. Initially, all visible markers are assigned to one object. When markers start moving apart, edges are removed and the graph gets clustered into multiple objects. To cluster graphs correctly into rigid clusters while maintaining realtime performance, we propose a clustering based on connected pyramids or 4-cliques. A pyramid is a rigid substructure consisting of 4 vertices, where each pair of vertices has an edge. Two pyramids are connected if and only if they share a triangle. The clustering is defined by the connected components of the pyramid graph, which can be efficiently computed by running a depth-first search from each node. Although connected pyramids do not necessarily form a clique, it is evident that markers within a cluster are part of the same rigid structure.

A pyramid-based clustering can be efficiently computed by determining all triangles in a graph, and connecting triangles if and only if they both share an edge and if there is an edge between the adjacent vertices (i.e. if they form a pyramid). As opposed to a triangle-based clustering, our pyramid-based clustering identifies the subgraphs G_1 and G_2 in Figure 4 correctly.

Next to each object graph G_O, a model of all object markers is maintained in a normalized coordinate system. Marker locations are averaged over all frames to reduce inaccuracies due to noise and outliers. The locations can be determined by calculating the rigid body transform that maps the identified markers to the corresponding model markers in a least-squares manner [Hor87]. This transform is used to predict the locations of occluded markers. When a marker is found for which no frame-to-frame correspondence could be established, its location is compared to the predicted occluded marker locations.

4.2. Object calibration summary

The complete object calibration procedure is summarized in the following steps:

1. *Marker tracking:* Blobs are detected in all camera images, and the corresponding 3D locations are determined using stereo correspondence, as described in Section 3. Markers are associated with the markers in the previous frame by frame-to-frame correspondence, so that each marker has a unique ID that stays constant during the time it is visible.
2. *Edge updating:* For each object graph G_O, the edges between visible markers within the model are updated.

3. *Invalid marker removal:* In certain circumstances, the blob detector may find blobs that do not correspond to valid markers. If these false blobs are present in multiple camera images, this may result in false 3D marker positions. However, these markers are only visible for a short period of time, and are therefore easily detected and removed from the object graphs.
4. *Graph clustering:* Each object graph G_O is clustered into new object graphs as necessary, by performing the pyramid based graph clustering technique.
5. *Occluded marker prediction:* The rigid body transform of each object is calculated and the locations of its occluded markers are predicted. Next, all markers for which no frame-to-frame correspondence could be established are compared with these occluded markers, and if they are within a certain radius of each other, the marker is recognized as a reappearing marker. Its ID and age are updated with those of the occluded marker. Note that only markers that are considered reliable are used to compute the rigid body transform.
6. *New marker insertion:* New markers that have not been recognized as reappearing markers are inserted into all object graphs.
7. *Object collapsing:* Since new markers are inserted into all object graphs, markers end up in multiple objects. When objects are moved, these duplicate markers are assigned to small invalid objects, which are a subset of the markers of a valid object. This step removes such objects.

5. Model-based object tracking

The input to the tracking system is a calibrated model graph $G_m = (V,E)$, where V is a set of vertices denoting marker locations, and E is a set of edges, where an edge E_{ij} is present if and only if vertices V_i and V_j can be visible at the same time. The tracking system needs to identify a subset of this graph in the image points. This is closely related to the double subgraph isomorphism problem. A subgraph G_1 is isomorphic to another subgraph G_2 if there is a one-to-one correspondence between their vertices and there is an edge between two vertices of G_1 if and only if there is an edge between the corresponding vertices in G_2. As this problem is known to be *NP*-complete [MA99], we simplify the problem by defining a minimum size S_{min} of a subgraph of G_m which needs to be present in a data graph G_d, with the constraint that this subgraph is a clique. Parameter S_{min} defines how many markers are needed to unambiguously identify a model graph, and each set of S_{min} markers that can be visible simultaneously is fully connected per definition. Note that S_{min} can be determined during object calibration.

The first step is to preprocess the model graph G_m. A hashtable is constructed which indexes each distance between vertices V_i and V_j for which there exists an edge E_{ij}. The table stores pointers back to the model and V_i and V_j. The tracking method proceeds as follows. First, a data point p is chosen, and the distance $d = \| p - p_j \|$ between p and all other data points p_j is indexed into the hashtable. Each vertex V_k in G_m maintains a list of possible matching data and model point pairs, $< p_j, V_l >$. Consequently, a set of candidates is created for each V_k, for which $\| p - p_j \| = \| V_k - V_l \|$. If p matches V_k, there must be a combination of points $< p_j, V_l >$ which is fully connected and for which the remaining distances are correct, since we defined that each set of S_{min} matching points must form a clique. Next, each combination of 3 points of each candidate is checked for the rest of the distances with the model. If these match, we have found a subgraph isomorphism of size 4. Next, the rigid body transform matching the model to the data graph is found by least-squares [Hor87]. All data points are then transformed to the model coordinate system, and compared to the model points. If at least S_{min} matching points are found, the tracking system can mark the model as identified and stop the search for this model. However, to increase robustness, we examine all candidates and select the one that matches the image points best. This best fit is found by minimizing

$$F = \frac{1}{N} \sum_{i=1}^{N} \| p_j - \mathbf{M} d_i \| \tag{6}$$

where $\mathbf{M}$ is the transform that maps the data points to the model points, p_j denotes a recognized model point, and d_i is its corresponding data point. Note that the tracking method implicitly exploits the fact that an edge is only present if the two markers can be simultaneously visible, thus greatly reducing the number of candidates.

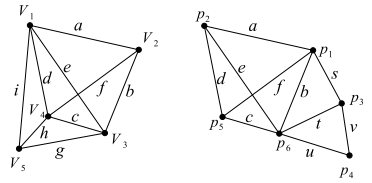


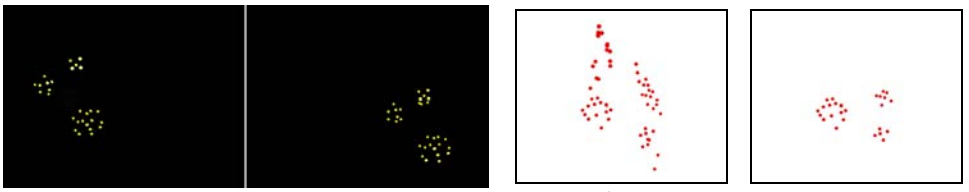
Figure 5: *(Left) An example model graph G_m with vertices V_i and distances $a, \ldots, i$. (Right) A data graph G_d with points p_i, which has a double subgraph isomorphism with G_m*

As an example, consider the model and data graphs of Figure 5. This situation could occur when model point V_5 is occluded, V_2 and V_5 cannot be visible simultaneously, and data points p_3 and p_4 represent spurious markers. The hashtable of G_m, omitting model pointers, is given by

$$\mathbf{H} = \begin{array}{lll} a \to (V_1,V_2) & d \to (V_1,V_4) & g \to (V_3,V_5) \\ b \to (V_2,V_3) & e \to (V_1,V_3) & h \to (V_4,V_5) \\ c \to (V_3,V_4) & f \to (V_2,V_4) & i \to (V_1,V_5) \end{array}$$

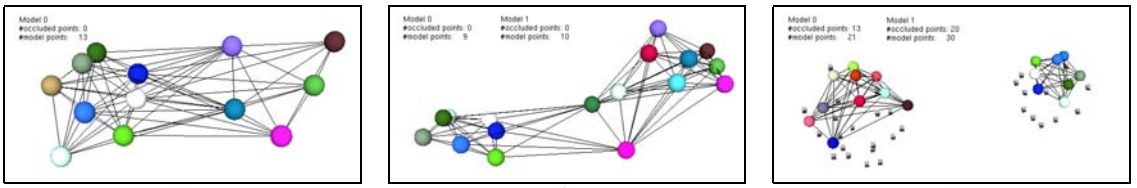
The distance matrix of the data graph G_d is given by

$$\mathbf{M_d} = \begin{pmatrix} 0 & a & s & 0 & f & b \\ a & 0 & i & 0 & d & e \\ s & i & 0 & v & 0 & t \\ 0 & 0 & v & 0 & 0 & u \\ f & d & 0 & 0 & 0 & c \\ b & e & t & u & c & 0 \end{pmatrix}$$



(a) (b) (c)

Figure 6: *Stereo correspondence. (a) Camera images. (b) Resulting 3D marker locations of stereo matching by epipolar constraint only. (c) Resulting 3D marker locations of our SVD based stereo correspondence.*



(a) (b) (c)

Figure 7: *Three frames during the simultaneous calibration of the cubical and spherical objects of Figure 1. (a) Initially, all points are regarded as one object. (b) After some movement, edges are removed and the system correctly identifies two objects. (c) During calibration, occluded point locations are predicted and drawn with grey cubes.*

The tracking method first hashes all distances $\| p_1 - p_i \|$, $i = 2, \ldots, 6$, i.e. the first row of $\mathbf{M_d}$, into $\mathbf{H}$ to generate a list of candidate matches

$$\begin{array}{lllll} V_1 \rightarrow & < p_2, V_2 > & & & \\ V_2 \rightarrow & < p_2, V_1 > & < p_5, V_4 > & < p_6, V_3 > & \\ V_3 \rightarrow & < p_6, V_2 > & & & \\ V_4 \rightarrow & < p_5, V_2 > & & & \end{array}$$

Next, all vertices V_i with at least 3 matches are taken as a candidate for point p_1, which in this case is only V_2. All combinations of 3 points are examined and the system checks the remaining distances

$$\begin{aligned} \| V_1 - V_4 \| &= \| p_2 - p_5 \| \\ \| V_1 - V_3 \| &= \| p_2 - p_6 \| \\ \| V_3 - V_4 \| &= \| p_6 - p_5 \| \end{aligned}$$

Since all distances match, a double subgraph isomorphism of size 4 has been found as $< V_1, p_2 >, < V_2, p_1 >, < V_3, p_6 >, < V_4, p_5 >$. The system can determine a rigid body transform to find other matching data points, and accepts the match if the fit is good enough and S_{min} points are found.

6. Results

We have implemented and evaluated our marker tracking, object calibration, and model based tracking techniques in the PSS, our near-field desktop VR system. In the following subsections, we will present some results on stereo correspondence compared to an approach using only the epipolar constraint, and show results of the object calibration and tracking methods in terms of robustness and performance.

6.1. Stereo correspondence

A straightforward method for stereo correspondence is to match all pairs of points in two camera images that are within a certain epipolar distance of each other. However, this method generates many false matches when multiple points are close to the same epipolar line. Figure 6 illustrates the difference between this approach and our SVD based matching approach. Figure 6(a) shows the blobs in the camera images, while Figures 6(b) and (c) depict the output of stereo correspondence by epipolar matching, and stereo correspondence by SVD matching, respectively. The SVD based correspondence successfully identifies the correct 3D marker locations, whereas the epipolar matching generates many additional false matches.

6.2. Object calibration

Figure 7 depicts three frames of a calibration sequence of 2200 frames, where a spherical object of diameter 7cm and a cubical object of 7x7x7cm are calibrated simultaneously (see Figure 1). These objects consists of 24 and 30 markers, respectively. The figure shows that initially, all points are regarded as a single object. In the second frame, after some movement, connections between new points and previously identified points are created, and connections between points not rigidly attached are removed. At this point, the calibration system correctly identified two objects. Note that a triangle-based clustering of this graph would result in only one object. In the last frame, after a calibration sequence of only 36 seconds, the system correctly identified two objects and created a model of all markers attached to the objects. Creating these models by hand is obviously a very difficult and time consuming task.

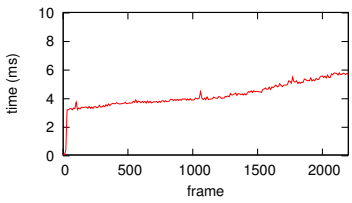


Figure 8: *Computational time of the simultaneous calibration of a spherical and cubical object of 30 and 24 markers.*

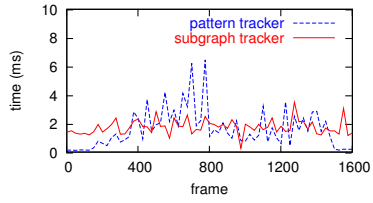


Figure 9: *Computational time of our subgraph tracker vs. a pattern based tracker. The dataset contains the movements of two cubes being manipulated simultaneously.*

Tracker	*Cube 1*			*Cube 2*		
Method	*hits*	*misses*	*rel.*	*hits*	*misses*	*rel.*
Subgraph	1672	55	96.8%	1619	108	93.7%
Pattern	1639	88	94.9%	1550	177	89.8%

Table 1: *Hits and misses for a datasequence of two cubic interaction devices, for our subgraph tracker vs. a pattern tracker*

Figure 8 gives the total computation time the calibration procedure requires for each frame for the same calibration sequence as Figure 7. The computation time slowly increases while the objects are being moved, as more points appear and the models get more complex. Most of the time is spent in the graph clustering procedure. The figure shows that the time required to update the models is well below 10 ms, meaning we can calibrate two objects, with a total of 54 markers, with a framerate of over 100 Hz.

6.3. Tracking

We compare the performance of our new subgraph tracker with a pattern based tracker based on matching distances, which is described in [vLvR04]. Since we need to define the location of the patterns on the interaction device manually, we used two cubic object of sizes 7x7x7cm and 5x5x5cm, instead of the spherical object. For each cube, we defined 6 patterns of 5 points for the pattern tracker, and calibrated the object for our subgraph tracker. Next, a dataset was recorded, where both cubes were manipulated simultaneously with both slow and fast, more erratic movements. For both trackers, we determined the number of frames the cubes could not be found (misses) versus the number of frames the cubes were identified (hits), and the computational time required by the tracking method, excluding blob detection and stereo correspondence. Both trackers examine all candidate matches, and select the one with the lowest fit metric as defined by Equation 6. This results in identical accuracy of both trackers.

Table 1 shows the performance of both trackers in terms of hits and misses. The table demonstrates that a pattern based tracker does not deal with multiple partially visible patterns, while the subgraph tracker has enough information in these situations to correctly identify the device and its pose. Both trackers perform well, as closer inspection of the results reveals that most misses are due to either failed blob detection, or that one cube occludes the other.

Figure 9 gives the computational performance for both methods on the same dataset. The figure shows that both methods are competitive and able to track both cubes with more than 100 Hz.

7. Discussion

We have described a method for the automatic calibration of object models of arbitrary shape, and the use of these models in a subgraph tracking system. Objects are equipped with retroreflective markers, and used as interaction devices in our virtual environment. We now discuss some advantages and disadvantages of the calibration and tracking techniques.

7.1. Object calibration

The object calibration method can handle virtually any shape, as long as at least 3 non-colinear points are visible and recognized in order to establish a relation with new points. The method can calibrate multiple objects with a moderate amount of markers simultaneously. Since objects that are identified as seperate clusters are never reconnected, the graph clustering method never needs to handle more points than are on a device, making the complexity practically linear in the number of devices. The worst case performance of the clustering method occurs when the graph is fully connected. A fully connected graph has $\binom{N}{3} = O(N^3)$ triangles, where each triangle forms a pyramid with $3(N-3)/2$ other triangles. Therefore, the worst case computational complexity is $O(N^4)$, which we may be able to improve by updating the pyramid graph incrementally. In our experience, three objects of 30 markers can be calibrated with high framerates.

The objects calibration method assumes that motions are slow and smooth, and that the 3D data is reasonably reliable. This means that mistakes in marker tracking or blob detection (e.g. multiple blobs that become one when they are aligned during rotation), may result in calibration errors. In order to support faster and more erratic motions, some simple strategies can be applied. First, we can use a Kalman filter to estimate 3D marker locations, so that outliers and

jittering can be reduced. The filter can also be used to predict marker locations more accurately, resulting in more reliable frame-to-frame correspondence. Second, the robustness of the blob detection can be increased by incorporating marker quality metrics. For instance, the roundness of a blob can be checked, so that two markers forming one blob in a camera image can be rejected. Third, in case frame-to-frame correspondence is completely lost, the tracking method can be applied using the model acquired so far, until a known part of the model is found again. This situation occurs when less than 3 non-colinear markers remain visible. This would also enable a user to completely remove an object from the workspace during calibration, and insert it at a later time, or even to completely stop the calibration procedure to resume it at a later time.

Figure 10: *A partially occluded cube. A pattern based tracking approach would fail to recognize the object, as both sides are only partially visible. Our subgraph tracker correctly identifies the object and determines its pose.*

7.2. Model-based tracking

The tracking method treats the markers on an interaction device as one point-cloud, and does not require markers to be grouped into patterns of fixed size. This makes the system more flexible compared to pattern based approaches. For instance, in the situation illustrated in Figure 10, two sides of a cube are partially occluded. A pattern based approach would fail to recognize the object, whereas our subgraph tracker correctly identifies the cube and its pose.

Results show the method is competitive with a pattern based approach in terms of computational efficiency, and more robust against occlusion. Although it would be possible to use a pattern based tracker with a calibrated device model by generating all patterns from the model, this would require a prohibitive amount of patterns. For instance, an object with 30 markers would require $\binom{30}{5} = 142506$ patterns. Although this number can be decreased by using visibility information, the number of patterns required is clearly too large for realtime tracking.

There is a tradeoff between tracking speed and the allowed noise levels in marker distance. Higher noise levels mean that more distances are indexed into the same location in the hashtable, and generate more candidates. In our tracking system, noise levels were set so that the distance error could not exceed 4 mm. In this case, we are able to track 2 objects of each 30 markers with over 100 Hz. With a much larger number of markers, performance degrades.

The accuracy of the tracking method is identical to pattern based approaches that optimize the pose estimate by performing a fit on all data, minimizing Equation 6. Accuracy depends on the size of the object, where a smaller object results in more jitter in the pose, and the quality of the blob detection, which is related to the distance of the object to the cameras and lighting conditions.

8. Conclusion

In this paper, we have presented a novel optical tracking system, capable of the automatic calibration and tracking of objects of arbitrary shape. The system is marker based, allowing a user to equip an object with retroreflective fiducials, and train the system to recognize the object by moving it in front of the cameras. The tracking system is based on subgraph matching, finding a subset of the model graph in the data.

Results show the system handles significant occlusion, is robust against noise and outliers in the data, and maintains framerates over 100 Hz when calibrating and tracking two objects of 30 markers simultaneously.

Future work will include investigating techniques to support fast and erratic motions during object calibration. This can be achieved by incorporating the tracking method into the calibration system in case of loss of frame-to-frame correspondence, better blob detection, and noise reduction strategies.

References

[ART] http://www.ar-tracking.de/.

[BT98] BIRCHFIELD S., TOMASI C.: Depth discontinuities by pixel-to-pixel stereo. *ICCV* (1998), 1073–1080.

[Dor99] DORFMÜLLER K.: Robust tracking for augmented reality using retroreflective markers. *Computers and Graphics 23*, 6 (1999), 795–800.

[Hor87] HORN B.: Closed-form solution of absolute orientation using unit quaternions. *Journal of the Optical Society of America, A 4*, 4 (1987), 629–642.

[HSDK05] HORNUNG A., SAR-DESSAI S., KOBBELT L.: Self-calibrating optical motion tracking for articulated bodies. In *Proceedings of the IEEE Virtual Reality Conference* (2005), pp. 75–82.

[LSW88] LAMDAN Y., SCHWARTZ J. T., WOLFSON H. J.: On recognition of 3-d objects from 2-d images. In *Proceedings of IEEE Int. Conf. on Robotics and Automation* (1988), pp. 1407–1413.

[MA99] M.J. ATALLAH E.: *Algorithms and Theory of Computation Handbook*. CRC Press LLC, 1999.

[MN00] MILLS S., NOVINS K.: Motion segmentation in long image sequences. In *Proceedings of the British Machine Vision Conference (BMVC2000)* (2000), pp. 162–171.

[MvL02] MULDER J., VAN LIERE R.: The personal space station: Bringing interaction within reach. In *Proceedings of the Virtual Reality International Conference, VRIC 2002* (2002), Richer S., Richard P.,, Taravel B., (Eds.), pp. 73–81.

[Pil97] PILU M.: A direct method for stereo correspondence based on singular value decomposition. *IEEE International Conference of Computer Vision and Pattern Recognition* (1997), 261–266.

[PMG85] POLLARD S., MAYHEW J., G.P.FRISBY: PMF: A stereo correspondence algorithm using a disparity gradient limit. *Perception 14* (1985), 449–470.

[RPF01] RIBO M., PINZ A., FUHRMANN A.: A new optical tracking system for virtual and augmented reality applications. In *Proceedings of the IEEE Instrumentation and Measurement Technical Conference* (Budapest, Hungary, 2001), vol. 3, pp. 1932–1936.

[SLH91] SCOTT G., LONGUET-HIGGINS H.: An algorithm for associating the features of two patterns. In *Proc. Royal Society London* (1991), vol. B244, pp. 21–26.

[Smi95] SMITH S.: ASSET-2: Real-time motion segmentation and shape tracking. In *Proc. 5th Int. Conf. on Computer Vision* (1995), pp. 237–244.

[SS02] SCHARSTEIN D., SZELISKI R.: A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *Int. J. Comput. Vision 47*, 1-3 (2002), 7–42.

[VIC] http://www.vicon.com/.

[vLvR04] VAN LIERE R., VAN RHIJN A.: An experimental comparison of three optical trackers for model based pose determination in virtual reality. *Eurographics Symposium on Virtual Environments* (June 2004), 25–34.

[ZF92] ZHANG Z., FAUGERAS O.: Three-dimensional motion computation and object segmentation in a long sequence of stereo frames. *Int. J. Comput. Vision 7*, 3 (1992), 211–241.