Eurographics Symposium on Virtual Environments (2006)
Roger Hubbold and Ming Lin (Editors)

GraphTracker: A Topology Projection Invariant Optical Tracker

F.A. Smit[1],A. van Rhijn[1],R. van Liere[1,2]

[1]Center for Mathematics and Computer Science, CWI, Amsterdam
[2] Department of Mathematics and Computer Science, Eindhoven University of Technology

Abstract

In this paper, we describe a new optical tracking algorithm for pose estimation of interaction devices in virtual and augmented reality. Given a 3D model of the interaction device and a number of camera images, the primary difficulty in pose reconstruction is to find the correspondence between 2D image points and 3D model points. Most previous methods solved this problem by the use of stereo correspondence. Once the correspondence problem has been solved, the pose can be estimated by determining the transformation between the 3D point cloud and the model.
Our approach is based on the projective invariant topology of graph structures. The topology of a graph structure does not change under projection: in this way we solve the point correspondence problem by a subgraph matching algorithm between the detected 2D image graph and the model graph.
There are four advantages to our method. First, the correspondence problem is solved entirely in 2D and therefore no stereo correspondence is needed. Consequently, we can use any number of cameras, including a single camera. Secondly, as opposed to stereo methods, we do not need to detect the same model point in two different cameras, and therefore our method is much more robust against occlusion. Thirdly, the subgraph matching algorithm can still detect a match even when parts of the graph are occluded, for example by the users hands. This also provides more robustness against occlusion. Finally, the error made in the pose estimation is significantly reduced as the amount of cameras is increased.

Categories and Subject Descriptors (according to ACM CCS): I.4.8 [Scene Analysis.]: Image Processing and Computer Vision.Tracking; I.3.6 [Computer Graphics.]: Methodology and Techniques.Interaction Techniques;

1. Introduction

Tracking in virtual and augmented reality is the process of identification and pose estimation of an interaction device in the virtual space. The pose of an interaction device is the 6 DOF orientation and translation of the device. Several tracking methods are in existence, including: mechanical, magnetic, gyroscopic and optical. We will focus on optical tracking as it provides a cheap, wireless interface, and is less susceptible to noise compared to the other methods.

A common approach to track an input device by an optical tracker is marker based. The device is usually augmented by specific marker patterns recognizable by the tracker. Optical trackers often make use of infra-red light combined with retro-reflective markers to greatly simplify the required image processing. The markers can then be detected by simple, fast threshold and blob detection algorithms. The same approach is followed in this paper.

Once a device has been augmented by markers, the three dimensional positions of these markers are measured and stored in a database. We call this database representation of the device the model. Optical trackers are now faced with three problems. First, the detected 2D image points have to be matched to their corresponding 3D model points. We call this the point-correspondence problem. Second, the actual 3D positions of the image points have to be determined; resulting in a 3D point cloud. This is refered to as the perspective n-point problem. Finally, a transformation from the de-

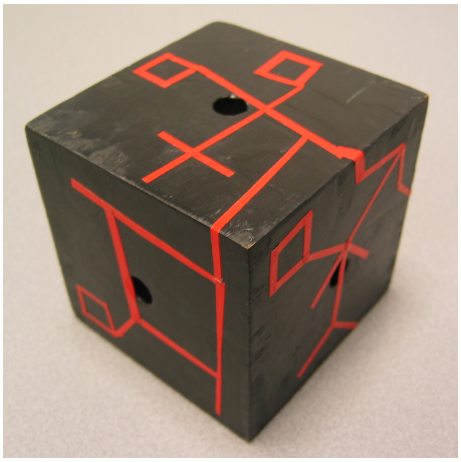

Figure 1: *A 7x7x7cm cubical input device augmented by a graph pattern of retro-reflective markers. Note that graph edges are allowed to cross over between faces of the cube and do not need to be straight lines.*

tected 3D point cloud to the corresponding 3D model points can be estimated using fitting techniques.

Many current optical tracking methods make use of stereo correspondences. All candidate 3D positions of the image points are calculated by the use of epipolar geometry in stereo images. Next, the point correspondence problem is solved by the use of an interpoint distance matching process between all possible detected positions and the model. A drawback to stereo correspondences is that every marker must be visible in two camera images to be detected. Also, since markers have no 2D identification, many false matches may occur.

A common and inherent problem in optical tracking methods is that line of sight is required. There are many reasons why a marker might be occluded, such as it being covered by the users hands, insufficient lighting, or self-occlusion by the device itself. Whenever a marker is occluded there is a chance that the tracker can not find the correct correspondence anymore. Trackers based on stereo correspondences are particulary sensitive to occlusion, as they might detect false matches, and require the same marker to be visible in two cameras simultaneously.

More recently optical trackers have made use of projection invariants. Perspective projections do not preserve angles or distances; however, a projection invariant is a feature that does remain unchanged under perspective projection. Examples of projective invariants are the cross-ratio, certain polynomials, and structural topology. Using this information, the point correspondence problem can be solved entirely in 2D using a single camera image. Invariant methods have a clear advantage over stereo correspondences: there is no need to calculate and match 3D point positions using epipolar geometry so that markers need not be visible in two cameras. This provides a robust way to handle marker occlusion as the cameras can be positioned freely, ie. they do not need to be positioned closely together to cover the same area of the virtual space, nor do they need to see the same marker.

In this paper we present an optical tracking method based on the projective invariant topology of graph structures. The topology of a graph structure does not change under projection: in this way we solve the point correspondence problem by a subgraph matching algorithm between the detected 2D image graph and the model graph. A sample input device is shown in Figure 1.

We have implemented and evaluated our tracking method using the Personal Space Station (PSS), a near-field desktop VR/AR environment [MvL02]. Users perform 3D spatial interaction using tangible input devices. The setup consists of two or four cameras equipped with infra-red filters and a ring of infra-red LEDs to illuminate the scene.

The paper is organized as follows. In section 2 we discuss related work. In section 3 we give a detailed technical description of our method. Section 4 shows experimental results, followed by a discussion of the pros and cons of the method in section 5.

2. Related Work

There are several examples of stereo correspondence based trackers as described in Section 1 [Dor99, RPF01, vRM05]. As our focus is on projective invariant trackers, we will not dicuss these any further.

A cross-ratio projective invariant of four collinear, or five coplanar points was used by van Liere et al. [vLM03]. They use the invariant property to establish pattern correspondence in 2D, as opposed to direct point correspondence. Once the pattern is identified, the individual points are matched using a stereo based distance fit. The matching is simplified significantly due to the previously established pattern correspondence.

Van Rhijn [vRM04] used the angular cross-ratio of line pencils as projective invariant. Once pattern correspondence has been established, the rotational part of the device pose can be determined directly by a line-to-plane fitting routine. The translation still needs to be determined from the combination of two camera images, however the same points needs not be visible simultaneously. The method can handle partial occlusion of the line pencils without difficulty.

A topological approach is suggested by Costanza et al. [CR03]. They make use of region adjacency trees to detect individual markers. Detection is performed by a subgraph matching algorithm, which can optionally be made error tolerant. However, no device pose estimation is performed in the initial algorithm. An extension was described by Bencina et al. [BKJ05] who determined a 2D translation of markers

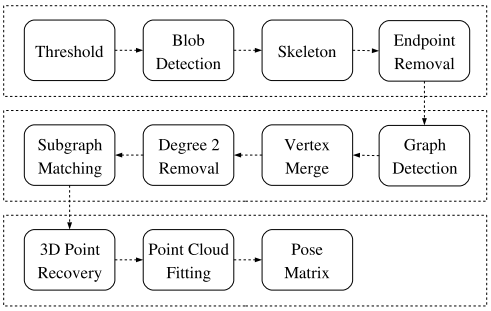

Figure 2: *The sequence of stages in the pipeline to go from a camera image to a device pose.*

on a flat transparent surface. However, their pose estimation appears sensitive to occlusion. Our method is somewhat related, especially in the subgraph matching phase. However, we use actual graphs and are able to determine a full 6 DOF pose, in addition to being less sensitive to occlusion.

A widely used framework for augmented VR is ARToolkit [KB99]. This system solves pattern correspondence by detecting a square, planar bitmap pattern. Using image processing and correlation techniques the coordinates of the four corners of the square are determined, from which a 6 DOF pose can be estimated. Drawbacks are that ARToolkit cannot handle occlusion, and only works with planar markers and four coplanar points. Fiala [Fia05] handled the occlusion problem by using an error correcting code as bitmap pattern in ARTag. However, the markers still need to be planar and the pose is estimated from four coplanar points. Our method can handle any number of points in many configurations, for example on a cylinder or a sphere.

3. Methods

In this section we give a detailed technical description of our method. Our method is based on the detection and matching of graphs to solve the correspondence problem. An overview of the processing pipeline is given in Figure 2 and 3. The first step in the pipeline is to perform some basic image processing to acquire a skeleton of the blobs in the input image. This skeleton is sufficient to reconstruct the topology of the graph. We also keep track of the clockwise planar ordering of edges. Next, some graph simplification is performed to eliminate spurious edges, followed by the ordered subgraph isomorphism testing phase to determine correspondence. Once we have determined a correspondence between the image points and the model, a pose estimation is performed. The pose estimation algorithm first calculates the 3D positions of the image points, followed by an absolute orientation algorithm to fit the point cloud to the model and find a transformation matrix. The four major stages are each described in a seperate subsection below. We conclude this section with a brief

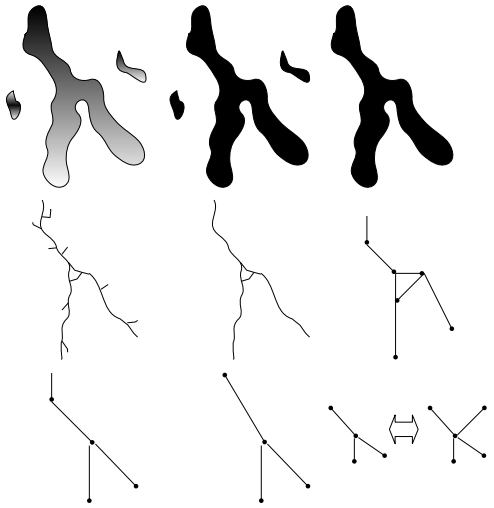

Figure 3: *A schematic visualization of the various stages in converting a camera image to a graph topology that can be matched (also see Figure 2). From top-left to bottom-right the images show a visualization of the state after: image aquisition, thresholding, blob detection, skeletization, end-point removal, graph detection, short edge removal, degree-two removal, and graph matching.*

mathematical analysis of the error made in pose estimation using multiple cameras.

3.1. Image Processing

Due to the use of retro-reflective markers and infra-red lighting, the preliminary image processing stage is straightforward. It is divided into four stages: thresholding, blob detection, skeletization, and end-point removal. All stages use simple algorithms as described by Gonzales and Woods [GW01].

First the input image is processed by an adaptive thresholding algorithm. Next, blobs are detected and merged starting at the points found by thresholding. The detected blobs are processed by a morphology based skeletization algorithm. The resulting skeleton suffers from small parasitic edges, which are removed in the final phase. The final result is a strictly 4-connected, single pixel width skeleton of the input blobs. This is the basic input to the graph topology detector.

3.2. Graph Detection

The graph detector makes some basic assumptions about the structure of graphs: any pixel that does not have exactly two neighbours in the skeleton is a vertex, and an edge exists between any vertices with a path between them consisting of pixels with exactly two neighbours. The implications are

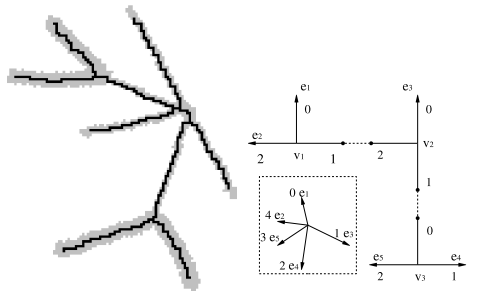

Figure 4: *(left) A detected blob in light grey with its skeleton shown in black. The top-right junction of 5 edges is split over 3 vertices. Our goal is to combine these vertices into a single vertex while maintaining edge ordering. (right) Merging vertices v_1, v_2, v_3 results in the vertex with incident edge ordering as given in the dashed inset. The numbers indicate the order of each incident edge e_i. The order of merging does not matter, for example merge(merge(v_1, v_2), v_3)) equals merge(merge(v_2, v_3), v_1))*

twofold. First, edges do not have to be straight lines; as long as two vertices are connected in some way there is an edge between them. This means the same graph can be on arbitrary convex surface shapes. Secondly, vertices of degree two cannot exists unless there is a self-loop at the vertex (ie. a degree one vertex with an extra self-loop becomes a degree two vertex).

Next, we discuss the algorithm used to detect the graph topology from a given skeleton image. Suppose we have a set of vertices $V = \{v_i\}$, and a map $M(x,y) \rightarrow v_i$ that maps image coordinates to this set whenever a vertex exists at that coordinate. The vertex set initially only contains the (arbitrary) starting point. As long as the set is not empty, we take a vertex from it an process this vertex as described in the next paragraph.

Once a starting vertex has been chosen, we start at an arbitrary neighbour of that vertex and perform a recursive walk to adjacent pixels. From now on we only consider the three neighbours of a pixel different from the neighbour we reached this pixel from. First we consult the map M to see if this pixel is an existing vertex, and if so, insert an edge. Also, the pixel is set to zero in the image to indicate it has been searched. Next, we examine the neighbours of this pixel. Whenever there is exactly one neighbour, we can simply 'walk' to this neighbour and continue the recursion. When there are multiple neighbours, the pixel is added to the vertex set V and an edge is inserted. When there are zero neighbours an edge is inserted, but the vertex is not added to the vertex set. After inserting an edge we choose a new starting vertex from the set V and repeat the procedure. The process terminates once the set V is exhausted, at which time the entire topology has been constructed.

For reasons explained in Section 3.3, we impose an ordering on the incident edges of a vertex. We keep track of the pixel direction (N,E,S,W) a vertex is left from, and the direction a vertex is reached from in the recursive walk over pixels (also see Figure 4). Every edge now has an ordering attribute on both ends. We use the starting points of edges for the ordering instead of the end points, as end points might affect the ordering when occluded. Also note that this ordering of incident edges is projectively invariant up to cyclic permutations.

Even though small edges were removed from the skeleton in the image processing phase, it is still possible for the detected graph to contain very short parasitic edges. These edges have a harmful effect on our matching algorithm, and thus they are removed in a subsequent step by merging their end-points. However, care must be taken not to affect the ordering of incident edges by merging two connected vertices (see Figure 4).

Finally, all vertices of degree two, except those containing self-loops, are removed in a similar fashion as short edges. These vertices can occur due to the chosen starting point, but cannot exist in theory.

3.3. Graph Matching

After detecting one or more graphs in the image, we try to match the detected graphs as subgraphs in the model graph to solve the point correspondence problem. To this extent we use an error tolerant subgraph matching algorithm. A subgraph isomorphism is a mapping of the vertices from one graph to the other that maintains the structure of the graph. All subgraph isomorphisms must be detected to verify a unique match. The problem of finding all subgraph ismorphisms is a notoriously complex one. The decision problem is known to be NPC, and finding all possible subgraphs cannot be done subexponentially [Epp99]. We use a slightly modified version of the VF algorithm by Cordella et al. [CFSV96, CFSV04] to do this matching in worst-case exponential time. However, test cases show that in practice the algorithm is fast enough to be used in real time.

We simplified the matching further with the following extensions. First, as edges can be occluded, vertices of degree one do not provide us with a reliable position. Therefore the matcher ignores all vertices of degree one while matching. They do, however, add to the degree of their adjacent vertex. Secondly, in order to reduce the amount of isomorphisms we impose an ordering on the incident edges of a vertex. For a match to be valid this ordering has to be a cyclic permutation in the model, possibly with gaps for missing edges. In this way a star graph with one center point and five edges only has five automorphisms, as opposed to $5! = 120$.

Whenever more than one subgraph isomorphism is detected, we scan for 'fixed points' that have the same mapping in all the isomorphic mappings. In this way some points can

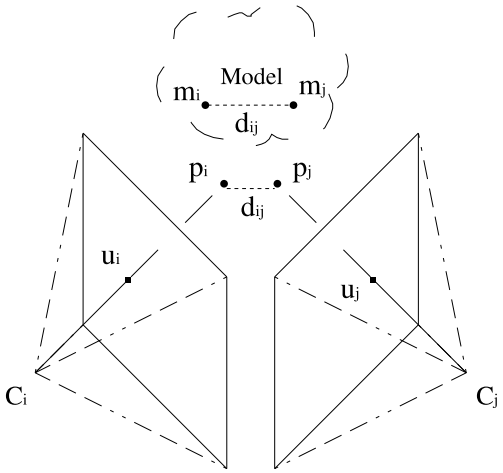

Figure 5: *Schematic view of the perspective n-point problem with two cameras. The goal is to reconstruct the p_i given the camera positions C_i, image points u_i and corresponding model points m_i*

be uniquely identified, even when multiple isomorphic mappings exist (see Figure 7). All fixed points, which will have degree greater than one, and their uniquely corresponding model points are provided as input to the pose reconstruction algorithm.

3.4. Pose Reconstruction

Once the correspondence between 2D image points and 3D model points is known, the 6 DOF device pose can be reconstructed. Our approach is very closely related to the method suggested by Quan [QL99]. We solve a system of polynomial equations by partial algebraic elimination and singular value decomposition, followed by Horn's [Hor87] absolute orientation determination algorithm using quaternions. Quan's method does not directly support multiple cameras, but the extension is straightforward. We briefly review the method now.

Given camera positions C_i, 3D image points u_i on the focal plane, corresponding 3D model points m_i, and the camera calibration matrices, the task is to calculate the 3D positions p_i and the transformation matrix M that maps p_i to m_i (see Figure 5). Since all p_i reside in a different frame as the m_i, we can only use interpoint relations in the same frame. Each point p_i lies on the 3D line through its corresponding image point u_i and the camera location C_i. For each pair of such lines we can write the line equation in parametric form and solve for the parameters (t_i, t_j) where the distance between those points equals $d_{ij} = ||m_i - m_j||^2$:

$$||(C_i + t_i(u_i - C_i)) - (C_j + t_j(u_j - C_j))||^2 = d_{ij} \quad (1)$$

Simplifying this equation and setting $D_i = u_i - C_i$ and $C_{ij} = C_i - C_j$ results in a polynomial in two unknowns, (t_i, t_j), with the following coefficient matrix:

$$\begin{pmatrix} -d_{ij} + C_{ij} \cdot C_{ij} & -2(D_j \cdot C_{ij}) & D_j \cdot D_j \\ 2(D_i \cdot C_{ij}) & -2(D_i \cdot D_j) & 0 \\ D_i \cdot D_i & 0 & 0 \end{pmatrix} \quad (2)$$

Every pair of points defines an equation of this form. However, solving such a system of non-linear equations algebraically proves extremely difficult. A direct solution can be obtained using Gröbner bases, however the number of terms in this solution is extremely large and it is sensitive to numerical errors.

We can simplify the coefficient matrix greatly by normalizing the direction vectors of the parametric lines ($D_i \cdot D_i = D_j \cdot D_j = 1$) and translating all cameras to the same point at the origin ($C_{ij} = \bar{0}$). Once these simplifications have been made, the polynomial system reduces to the same form as defined by Quan [QL99]:

$$P_{ij} = t_i^2 + t_j^2 - 2(D_i \cdot D_j) - d_{ij} = 0 \quad (3)$$

Because of the simplifying camera translations, the new distance calculation becomes:

$$d_{ij} = ||(m_i - C_i) - (m_j - C_j)||^2 = ||m_i - m_j - C_{ij}||^2 \quad (4)$$

Given N input points, there are $\binom{N}{2}$ constraining equations. Using three equations P_{ij}, P_{ik}, P_{jk} in t_i, t_j, t_k we can construct a 4th degree polynomial in t_i^2 by variable elimination. If we fix the first parameter to t_i, we can construct $\binom{N-1}{2}$ such polynomials in t_i^2. For $N > 4$ this system is an over defined linear system in $(1, t_i^2, (t_i^2)^2, (t_i^2)^3, (t_i^2)^4)$, which can be solved in a least-squares fashion by using a singular value decomposition on a $\binom{N-1}{2} \times 5$ matrix. In the case of $N = 4$ a slight modification has to be made, as the linear system is under determined in this case, but a unique solution can still be found (see [QL99] for details). This means we need to detect a minimum of four points over all the cameras to reconstruct the 3D point cloud.

Note that after calculating t_i we cannot simply substitute its value in the original polynomial equations. As the obtained solution is a least squares estimate, there might not exist a solution to the polynomial equation using this variable. Recall that the polynomial equation represents a constraint on the distance between two lines. By fixing a point on one of these lines, there is no guarantee that there even exists a point on the other line for which the distance constraint holds. One could minimize the difference in distance, however to be precise all points need to be taken into account.

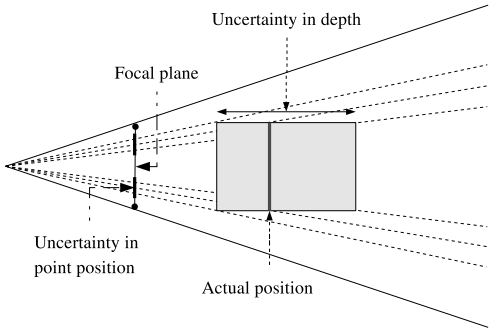

Figure 6: *Expected error perpendicular to the camera plane.*

Therefore, we simply solve all of the t_i seperately using the method described above.

At this point we have effectively solved the least squares perspective n-point problem for multiple cameras in closed form. The next step is to determine a transformation matrix between the determined 3D point cloud and the 3D model. To accomplish this we use the closed form absolute orientation method from Horn [Hor87]. As this algorithm can be left unmodified we will not describe it any further here.

3.5. Error Analysis

Next, we will describe a mathematical model of the errors made in pose estimation under the presence of noise, using multiple cameras. We can distinguish between three sources of errors: inaccurate vertex positions in the camera images, errors in the device description, and errors in the camera calibration. For our purposes we only consider the first source of errors, and assume both the device as well as the camera errors to be negligible.

As depth information cannot be obtained by using a single projected point, we assume a camera with two detected points as shown in Figure 6. If we assume the error in the projected image points to follow a Gaussian distribution with expectation ε_b, the resulting expected errors ε_{XY} in the XY-, and ε_Z in the Z-direction are given by:

$$\varepsilon_{XY} = \frac{d}{f}\varepsilon_b \qquad \varepsilon_Z = \frac{d^2\varepsilon_b}{lf - d\varepsilon_b} \tag{5}$$

where d is the distance from the points to the camera's center of projection, f the focal distance, and l the distance between the points. Note that ε_Z is larger than ε_{XY}, and thus the largest error is made in estimating depth information. Extending equation 5 to N points results in

$$\varepsilon_{XY}^N = \frac{1}{\sqrt{N}}\varepsilon_{XY} \qquad \varepsilon_Z^N = \frac{1}{\sqrt{N}}\varepsilon_Z \tag{6}$$

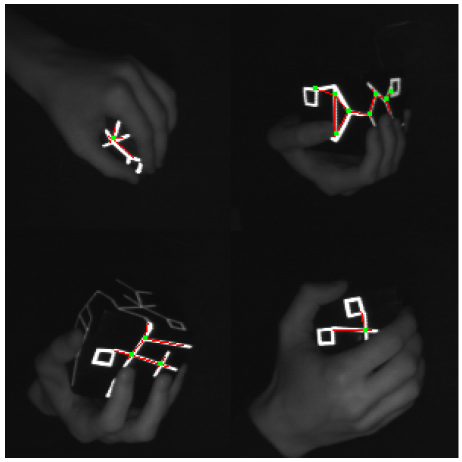

Figure 7: *When parts of the graph are occluded, some fixed points can still be detected. An interesting example is the bottom-right image; the detected subgraph matches the model in two ways. The point connecting the two self-loops can be uniquely identified by noting a fixed point. However, the points respresenting the loops themselves cannot be uniquely identified as the two points can be interchanged freely.*

Equation 6 shows that the error decreases in all directions when more points are visible to the camera. We can extend these equations to the case of multiple cameras. To do this, the expectations have to be transformed to a common reference frame. Writing N_c for the number of cameras, and ε_i for the transformed expectation of errors in the i-th camera gives

$$\varepsilon = \frac{1}{N_c}\sqrt{\sum_i \varepsilon_i} \tag{7}$$

for each of the three directions with respect to the reference frame. Equation 7 shows that the expected error is reduced by increasing the amount of cameras. Our initial claim to use multiple cameras in our method to reduce the amount of error in pose estimation is supported by this result.

4. Results

Our implementation uses a cubical input device augmented by retro-reflective markers, as shown in Figure 1. Several points can be uniquely identified in the pressence of occlusion (Figure 7), allowing the device pose to be reconstructed. Using multiple cameras increased the number of detected points as expected.

The implementation uses relatively unoptimized image processing algorithms, and runs entirely on the CPU. Even so, the entire reconstruction, from image processing to pose

	Accuracy (in mm)		Detected Points	
Camera	Average	RMSE	Mono	Stereo
1	-26.4	15.4	4.94	-
2	-25.1	15.3	5.29	-
3	-12.7	10.8	5.75	-
4	-20.5	24.3	5.75	-
1/2	0.232	0.90	10.23	4.89
3/4	-0.0754	2.01	11.50	1.39
1/2/3/4	0.0324	0.57	21.73	4.01

Table 1: *Measurement-to-plane summarized results for 1/2/4 cameras. The average distance to the XZ-plane and the RMSE are given in the first column. The average number of detected points per camera individually (mono) and equal corresponding points (stereo) are given in the last column.*

estimation with multiple cameras, takes between 10 and 20ms per frame. Of this time more than half is spent in the unoptimized image processing phase. These results show that our method is well suited to run in real-time with multiple cameras.

4.1. Accuracy

Determining the absolute accuracy of a tracker over the entire working volume is a tedious and very time-consuming process. A grid of sufficient resolution covering the tracking volume has to be determined. Next, the device has to be positioned accurately at each grid position, and the resulting tracker measurement has to be determined.

We have used a different, simplified approach. We slide the cube over a plane of constant height in the tracking volume and log the positional result. Thus, the measured cube positions should all lie on the XZ-plane. Now we determine the average distance to the XZ-plane, which gives an indication of the systematic error made. The standard deviation of this set of distances (RMSE) gives a good indication of the random error. Although this procedure does not provide us with an absolute accuracy, we do get relative accuracy with respect to the plane.

The results of the XZ-plane measurements for one, two and four cameras are shown in Figure 8 and Table 1, first column. For a different, random interaction session the average number of detected points per camera individually, and the average number of detected stereo corresponding points are listed in the last column of Table 1.

A number of observations can be made from these results:

- Both the systematic and the random error decrease as the number of cameras increases. A pair of cameras is more accurate than either of the single cameras, and four cameras are yet slightly more accurate than either pair of cameras.
- In the case of cameras 1 and 2, the XZ plane is almost parallel to the camera planes. In this case the error in position is mostly determined by the depth estimation. Hence, the error is dominated by ε_Z as given in Equation 5. For cameras 3 and 4 the XZ plane is at a near 45 degree angle, and thus the systematic error is decreased as can be seen in Table 1. However, the random error is increased as the cameras are positioned further away. The combination of all cameras is even more accurate as the device is viewed from more directions now.
- The total number of detected points increases as the number of cameras increases. Hence, a pose can be determined a larger percentage of the time with more cameras. The theoretical accuracy is increased as well, since more points are being used in the calculations. Stereo correspondence can often not be found, while individual cameras do see enough points combined for pose reconstruction.

5. Discussion

To allow for robustness against occlusion, it is important that a detected, occluded subgraph can be matched to a unique subgraph in the model. However, it is not yet clear what kind of graph topology is best suited to accomplish this. We found through direct experimentation that model graphs consisting of several ordered non-isomorphic components are desirable. Also, each of the components, and the entire model itself, should not be ordered automorphic (except for the trivial identity mapping).

A different approach would be to extend the graph matching algorithm to detect missing vertices and edges due to common cases of occlusion. However, developing a matching algorithm that incorporate this kind of meta-knowledge and still runs in real-time might be infeasible. We can also impose the requirement that all model graphs must be planar or locally planar, which allows the use of much faster graph matching algorithms [Epp99]. This might, however, affect the types of (convex) surfaces we can use for devices.

Another issue is the optimal placement of cameras. We expect a camera setup with three cameras, one on each principal axis, will be optimal with respect to the error made in pose estimation (see Section 3.5), however this has still to be verified formally. Also, by varying camera placements, occlusion by the users hands might be avoided entirely as one side of the input device might always be completely visible.

Future work will therefore consist primarily of finding specific model graphs, which are provably robust against occlusion. Also, developing a more robust graph matching algorithm with knowledge of occlusion cases might be desirable.

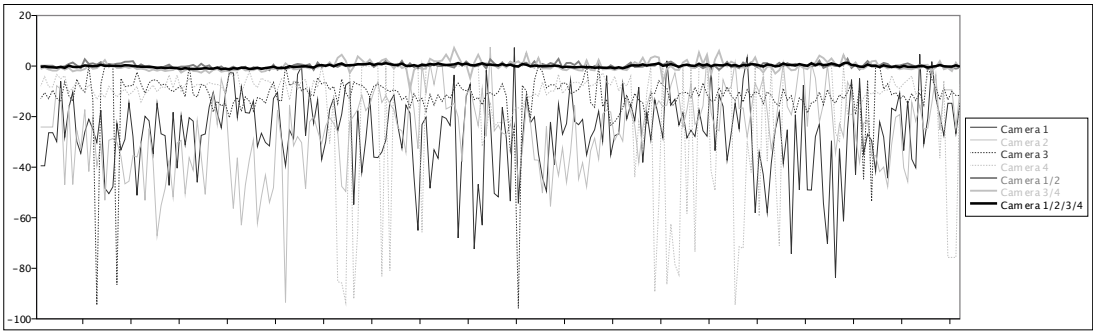

Figure 8: *Tracking accuracy with respect to the XZ-plane. The vertical axis shows the distance to the XZ-plane in millimeters. The horizontal axis represents a sequence of about 200 frames. When a camera could not detect a pose the values are ommited.*

6. Conclusion

We have proposed a projective invariant optical tracker based on graph topology. As the correspondence problem is solved entirely in 2D, any number of cameras can be used without restrictions. This, in combination with error tolerant subgraph matching, provided more robustness against occlusion than classic methods. Even though theoretically complex algorithms are used, the solution is fast enough to estimate a pose from multiple cameras in real-time. Also, by using multiple cameras the error made in pose estimation is reduced significantly.

References

[BKJ05] BENCINA R., KALTENBRUNNER M., JORDÀ S.: Improved topological fiducial tracking in the reactivision system. In *Proceedings of the IEEE Procams 2005* (2005).

[CFSV96] CORDELLA L. P., FOGGIA P., SANSONE C., VENTO M.: An efficient algorithm for the inexact matching of arg graphs using a contextual transformational model. In *ICPR '96: Proceedings of the International Conference on Pattern Recognition (ICPR '96) Volume III-Volume 7276* (1996), p. 180.

[CFSV04] CORDELLA L. P., FOGGIA P., SANSONE C., VENTO M.: A (sub)graph isomorphism algorithm for matching large graphs. *IEEE Trans. Pattern Anal. Mach. Intell. 26*, 10 (2004), 1367–1372.

[CR03] COSTANZA E., ROBINSON J.: A region adjacency tree approach to the detection and design of fiducials. In *VVG* (2003), pp. 63–69.

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

[Epp99] EPPSTEIN D.: Subgraph isomorphism in planar graphs and related problems. *J. Graph Algorithms & Applications 3*, 3 (1999), 1–27.

[Fia05] FIALA M.: ARTag, a fiducial marker system using digital techniques. In *CVPR (2)* (2005), pp. 590–596.

[GW01] GONZALEZ R. C., WOODS R. E.: *Digital Image Processing*. Addison-Wesley, 2001.

[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.

[KB99] KATO H., BILLINGHURST M.: Marker tracking and hmd calibration for a video-based augmented reality conferencing system. In *IWAR '99: Proceedings of the 2nd IEEE and ACM International Workshop on Augmented Reality* (1999), p. 85.

[MvL02] MULDER J., VAN LIERE R.: The Personal Space Station: Bringing interaction within reach. In *VRIC 2002 Conference Proceedings* (2002), pp. 73–81.

[QL99] QUAN L., LAN Z.: Linear n-point camera pose determination. *IEEE Trans. Pattern Anal. Mach. Intell. 21*, 8 (1999), 774–780.

[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* (2001), pp. 1932–1936.

[vLM03] VAN LIERE R., MULDER J.: Optical tracking using projective invariant marker pattern properties. In *Proceedings of the IEEE Virtual Reality Conference 2003* (2003), pp. 191–198.

[vRM04] VAN RHIJN A., MULDER J.: Optical tracking using line pencil fiducials. In *Proceedings of the Eurographics Symposium on Virtual Environments 2004* (2004), pp. 35–44.

[vRM05] VAN RHIJN A., MULDER J.: Optical tracking and calibration of tangible interaction devices. In *Proceedings of the Immersive Projection Technology and Virtual Environments Workshop 2005* (2005).