Eurographics Symposium on Virtual Environments (2004)
S. Coquillart, M. Göbel (Editors)

An Experimental Comparison of Three Optical Trackers for Model Based Pose Determination in Virtual Reality

R. van Liere and A. van Rhijn

Center for Mathematics and Computer Science, Amsterdam, Netherlands

Abstract

In recent years many optical trackers have been proposed for usage in Virtual Environments. In this paper, we compare three model based optical tracking algorithms for pose determination of input devices. In particular, we study the behavior of these algorithms when applied to two-handed manipulation tasks. We experimentally show how critical parameters influence the relative accuracy, latency and robustness of each algorithm. Although the study has been performed in a specific near-field virtual environment, the results can be applied to other virtual environments such as workbenches and CAVEs.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Virtual reality; I.4.9 [Image Processing and Computer Vision]: Scene Analysis; H.5.2 [Information Interfaces and Representation]: Input devices and strategies;

1. Introduction

Optical tracking for head-tracking and interaction devices in Virtual and Augmented Reality can provide a valuable alternative over other tracking methods like magnetic, gyroscopic, and mechanical trackers. Advantages of optical tracking are that it allows for wireless 'sensors', it is less susceptible to noise, and it allows for many objects to be tracked simultaneously.

The pose determination problem is well known in the computer vision literature and many algorithms and algorithm taxonomies have been proposed. A common approach for determining the pose of interaction devices is to use stereo cameras and model-based object recognition methods. Devices are equipped with features in a pre-defined configuration (the model) and correspondence algorithms are used to recognize the device features from the stereo paired images. The pose of the device can be determined after corresponding features have been recognized.

Virtual and augmented reality researchers have defined rules of thumb that are used to design optical tracker systems. For example, Welch and Foxlin have listed requirements for a *magical device*, [WF02]. Important requirements are the accuracy, latency and robustness of trackers.

In this paper, we experimentally compare three optical tracking algorithms for determining the pose of interaction devices. Our goal is to study how, and to what effect, environmental conditions (such as lighting conditions or erroneous camera calibration) have on accuracy, performance and robustness of each algorithm. Are all algorithms equally robust under various environmental conditions? How do varying environmental conditions effect the accuracy and performance of the algorithms? The answers to these questions will allow virtual reality developers to make better choices when designing and building their systems.

For our experiments we consider two hand movement tasks that are related to object manipulation in a near field desktop environment. However, the results are applicable to other motions and to other environments.

2. Related Work

We restrict our overview to outside-in optical trackers and systems that make use of active or passive markers that are attached to the objects to be tracked. The major advantage of using markers is that these can efficiently be found in the camera images. In particular, when using infrared light in

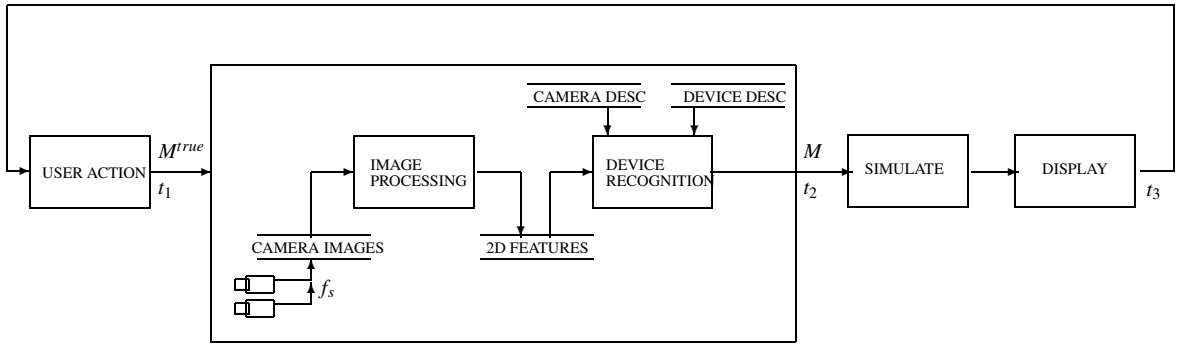

Figure 1: *Framework for model based optical tracking in VR/AR. Optical tracking consists of image processing and pose determination stages. Inputs include stereo camera images, camera descriptors, and device descriptors. Output is the pose of each device.*

combination with retro-reflective markers, image processing simply comes down to blob detection in a gray-scale image.

Virtual reality researchers have developed optical trackers using retro-reflective markers; eg. Dorfmüller [Dor99] and Ribo et al. [RPF01]. A commercial product is available from the German company Advanced Real-time Tracking, [ART]. These systems equip devices with multiple markers in a known 3D pattern and use a stereo vision system to detect and reconstruct the pose of the 3D pattern in a two step process.

The pose determination problem is well known in the computer vision literature and many algorithms and algorithm taxonomies have been proposed, eg. [FP03]. Experimental comparison studies have also been reported. Eggert et.al. , [ELF95] provide a comparative analysis of four algorithms which compute 3D rigid body transformations that aligns two sets of points. Quantitative and qualitative results are given of accuracy, robustness, stability and performance of each algorithm. Rusinliewicz et.al. [RL01] compare variations of the iterative closest point algorithm when applied to range images taken from multiple viewpoints. They evaluate what the effect of these variations is on the speed with which the correct pose is reached.

Our comparison study differs from the previous studies in two ways. First, Eggert et.al. study only those algorithms in which the point correspondence is known. In virtual environments, correspondence of points is usually not known. Rusinliewicz reports on variations of the ICP algorithm, whereas we take other algorithmic approaches into account. Their study focuses on alternative methods to improve efficiency, whereas we study the effects of noise on the behavior of the algorithms.

Second, the behavior of the motions we study are different than in the mentioned studies. Our study takes multiple objects into account (i.e. the global registration problem), whereas Eggert and Rusinliewicz study the alignment of a single object.

3. Method

3.1. General Framework

In this section we define a framework used for comparing model based optical trackers in a VR/AR environment. Its purpose is to identify critical parameters that influence tracking accuracy and performance, rather than to provide a general framework for tracking.

Consider figure 1 as a 4 stage framework. The user performs an input action with one or more input devices at time t_1. The pose of each input device is represented as a 4x4 transformation matrix; denoted as M_k^{true} for $k = 1...N_{device}$. User actions are captured by the optical tracking system and an approximation of the actual pose is determined; M_k. Next, the VR system can use M_k to update the simulation and, at time t_3, will display the virtual world. The time interval $t_3 - t_1$ is the end-to-end latency of the virtual environment. For comparing tracker algorithms, we define tracking latency as the latency introduced by the tracker; i.e. $t_2 - t_1$. Tracking accuracy is defined in terms of the difference between M_k^{true} and M_k. These definitions will be elaborated in section 5.

The optical tracking system consists of two stages; an image processing stage and device pose determination stage. In this framework we use a stereo camera pair, that provide images at a frequency f_s. Image processing is used to compute a set of 2D features for each image. The input to the pose determination stage are the 2D features, a description of intrinsic and extrinsic camera parameters, and a description of each device.

The fundamental problem in model based optical tracking is to establish the *correspondence* of data features with features in the device description. Once the correspondence

of features has been established, then the pose of the device can be computed. In this paper, we study three approaches for which algorithms can establish correspondence:

1. correspondence using 2D data features: In this case, the correspondence is performed in 2D. The basic steps involve using projective invariant pattern recognition methods to establish correspondence with 2D device features, use stereo geometry to transform the corresponding features to 3D, and compute the pose.
   In section 3.3.1 we describe an algorithm that uses projective invariant methods to establish correspondence.
2. correspondence using 3D data features: In this case, the correspondence is performed in 3D. The basic steps involve using epi-polar geometry to transform features to 3D, use a 3D pattern recognition method to establish correspondence with 3D device features, and compute the pose of the device with the 3D feature.
   In section 3.3.2 we describe a distance matrix algorithm that establishes correspondence in 3D.
3. correspondence using projecting device features: In this case, a pose of the device is projected into 2D and a fitting of the projected device features with data features is performed. The basic steps involve using the camera descriptors to project a chosen pose of the device, compute a 2D distance between the projected device features and the image data features. Perturb the chosen pose and iterate until the 2D distance approaches zero. The device features correspond with the data image features when the 2D distance is minimized.
   In section 3.3.3 we describe an Iterative Closest Point algorithm to illustrate this approach.

Optical tracking algorithms are very sensitive to the reliability of their inputs. We use the framework to identify three sources of errors that can influence the inputs to the algorithms:

- lighting conditions: processing of camera images is very sensitive to the lighting conditions. This can lead to inaccurate positions of the 2D features and will result in a high-frequency jitter in the device pose.
- camera calibration: inaccurate camera parameters will influence many aspects of the algorithms. For example, erroneous camera parameters influences stereo geometry computations and epi-polar geometry computations, inaccurate distortion parameters influence the geometry of the 2D features, etc.
- device description: inaccurate device descriptions will influence the computations involved to establish the correspondence between device and data features.

3.2. Devices and Device Descriptions

For our experimental study, we have chosen simple wooden cubes for input devices and point sets as the basis for device descriptors. Devices are constructed by placing small retro-reflective markers on each surface of the wooden cube. Infrared light from LEDs mounted on the camera is reflected by the markers back into the lens such that blobs of white pixels can be detected in the image. The center of gravity of each blob is the 2D position of the blob. We denote blobs as b_i with $i = 1...N_{blobs}$, in which N_{blobs} is the number of blobs in the image.

A device description is defined by the 3D positions of the markers on the device; denoted as m_i with $i = 1...N_{device}$, in which N_{device} are the number of markers on a device. Marker coordinates are given with respect to a common reference frame of the device.

Patterns are defined as a group of markers. Devices can be constructed from a pattern of 4 collinear markers, or from one or more patterns of 5 coplanar markers.

Figure 2 shows an example of two-handed interaction in the Personal Space Station, a near-field desktop virtual environment, [MvL02]. A molecule, tethered on a wooden cube, is held in one hand. The second hand uses a pen to position a clipping plane around the solvent surface of the molecule. The cube device consists of 6 patterns of 5 coplanar markers. The pen device consists of 4 collinear markers.

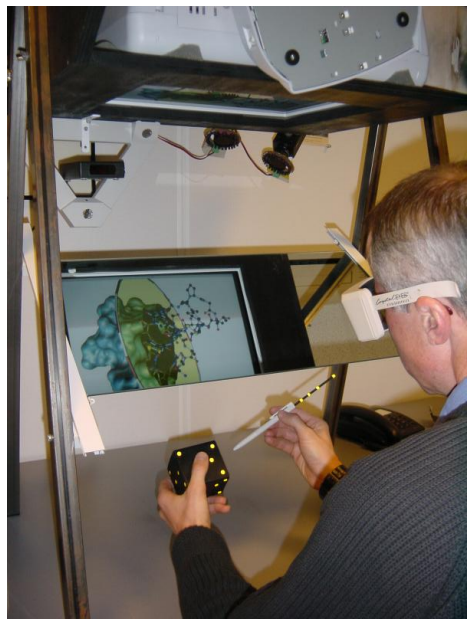

Figure 2: *Two-handed interaction in a near-field virtual environment. In this example a pen and cube equipped are used to explore a molecule. A 7x7x7 cm wooden cube device with 30 circular markers arranged in 6 patterns of 5 coplanar markers, and a pen device of 4 collinear markers. Markers have a diameter of 1/2 cm.*

In the sequel, we will assume that the device descriptions are sufficiently accurate.

3.3. Algorithms

3.3.1. Projective Invariant Patterns

In a recent paper, we describe an optical tracker algorithm that uses invariant properties of 2D point patterns to determine the pose of interaction devices, [vLM03].

A well known projective invariant relation is the cross ratio. A cross ratio of 4 collinear points (labeled as A, B, C, D) is the real number defined by: $\lambda = \frac{|AC| \ / \ |BC|}{|AD| \ / \ |BD|}$ where $|AC|$ is the length of the 2D line segment from A to C. A remarkable property of the cross ratio is that it is invariant under perspective projections; i.e. although the relative distances between the projected points on the line changes, the cross ratio remains constant.

The algorithm to find a 5 point device pattern in the 2D data points consists of three steps. First, the cross ratio for all 5 point combinations of the 2D data points is checked with the cross ratio of the device; i.e. $\lambda_{2D} = \lambda_{device}$. This step results in zero or more candidate patterns that correspond to the pattern in the device description. In the second step, 3D geometry tests are used to select the best candidate. This is realized by pairwise checking each candidate obtained in the image with all candidates obtained from an other image. Stereo geometry is used to transform the points of each candidate pair into 3D and the candidate pair which best corresponds to the 3D geometric information in the device description is selected as the best pattern. The 3D fit value of a candidate pattern with a data pattern is defined as the difference of 3D distances between points:

$$F = \sum_{i=1}^{5} \| p_{i,i+1} - d_{i,i+1} \| \tag{1}$$

In the third step, the 3D points of the best fit used to determine the pose of the device.

The complexity of this algorithm is order $O(N^5)$ since $\begin{pmatrix} N \\ 5 \end{pmatrix}$ point combinations in a N point space must be tested.

Dealing with noise Since the cross ratio is sensitive to noise in the 2D point positions, an off-line training session is used to determine deviations in the cross ratio. Instead of storing the cross ratio as λ_{device}, an interval $[\lambda^{min}_{device}, \lambda^{max}_{device}]$ of all computed cross ratios during the training session is stored. During tracking, a correspondence will occur when $\lambda_{2D} \in [\lambda^{min}_{device}, \lambda^{max}_{device}]$.

3.3.2. 3D Distance Matrix

We have developed a pose determination algorithm that is based on 3D distances between points. The algorithm is a special case of the general Euclidean Distance Matrix completion problem [Lau01], and can be seen as a generalization of the method proposed by Dorfmüller, [Dor99].

An $N \times N$ euclidean distance matrix is a symmetric matrix, $D = [d_{ij}]$ that stores all 3D distances between N points; i.e. $d_{ij} = ||x_i - x_j||$ for $i, j = 1, 2, 3, ...N$.

For each pattern on the device, a distance matrix is constructed by measuring the 3D distance between each marker pair of the pattern. For a pattern of 5 points, this results in a 5×5 distance matrix,

$$P = [p_{ij}] = \begin{pmatrix} 0 & p_{12} & p_{13} & p_{14} & p_{15} \\ p_{12} & 0 & p_{23} & p_{24} & p_{25} \\ p_{13} & p_{23} & 0 & p_{34} & p_{35} \\ p_{14} & p_{24} & p_{34} & 0 & p_{45} \\ p_{15} & p_{25} & p_{35} & p_{45} & 0 \end{pmatrix} \tag{2}$$

During tracking, epi-polar geometry is used to compute M 3D points from 2D data points in a pair of stereo images. An $M \times M$ data distance matrix $D = [d_{ij}]$ is computed from the M 3D data points. The algorithm to find a pattern in the 3D data points consists of two steps. The first step finds all the 5 sub-matrices in the data distance matrix D which fits in the pattern distance matrix P. i.e. find rows k and columns l such that $p_{ij} = d_{kl}$ for $i, j = 1...5$. As a second step, a least squares fitting method is used to find which sub-matrix of D best fits pattern distance matrix P. The fit value F of a sub-matrix with the point pattern is defined as:

$$F = \sum_{i,j=1}^{5} \| p_{ij} - d_{kl} \| \tag{3}$$

The complexity of this algorithm is order $O(M^2 \log(M))$, since each element in the data distance matrix must be tested.

Dealing with noise Without noise, it would be sufficient to test if $p_{ij} = d_{kl}$. However, with noisy data, the test must be extended to include all possible distances within an interval around d_{kl}. The length of this interval can be trained or a predefined threshold value can be chosen. Hence, the test will be $p_{ij} \in [d_{kl} - T_{3d}, d_{kl} + T_{3d}]$ in which T_{3d} is the threshold value.

3.3.3. Iterative Closest Point

The iterative closest point (ICP) algorithm was introduced in 1992 by Besl and McKay, [BM92]. It is a general purpose, representation-independent method for registering 3D free form shapes. The ICP algorithm is designed to match a data shape to a model shape, under the assumption that the data shape is formed by a (possibly noisy) sampling of the model. Given a particular pose, a distance function is used to determine how well the data shape fits the model. An optimization procedure is used to minimize the distance function.

A projection of the device model is used in order to compare the blob point set with the device description. The projection depends on the translation/orientation of the device pose and the camera's intrinsic/extrinsic parameters. The

projection of a device model point onto the camera image is given by

$$p_i = C\,M\,m_i \tag{4}$$

in which C is the camera transformation matrix, M is the device transformation matrix and p_i is the projected model point of m_i.

Normals in the device description are used for determining the visibility of a model point and are used to cull points on surfaces that are not facing the camera.

The distance metric D is defined as the average distance of every blob b_j to the closest projected model point p_i;

$$D = \frac{1}{N_{blobs}} \sum_j min\{d(b_j, p_i) | i = 1...N_{model}\} \tag{5}$$

in which $d(b_j, p_i)$ is the euclidean distance between blob b_i and projected model point p_i.

Figure 3 illustrates three iterations of the ICP algorithm for one camera and one cubic device. The first iteration (left) shows the distance between the data image points (drawn in red) and the projected device points as black lines. Subsequent iterations (middle and right) show – after translation and orientation of the device – that the distances between the data image points and the projected device points is minimized. Note that in this formulation of the distance function no correspondence information is taken into account; i.e. more than one 2D point can correspond to the same device feature.

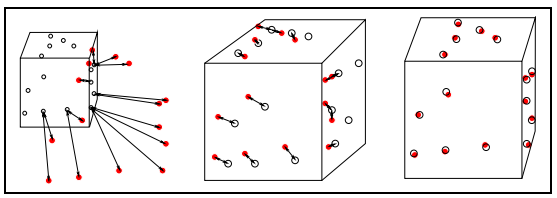

Figure 3: *Three iterations of the ICP algorithm: a device is positioned and oriented so that the distance between the image points (in red) and the model (as circles) is minimized.*

The ICP method can be formulated as an optimization problem in which the distance function produces an erratic multi-dimensional landscape with many local minima. A 6 DOF device results in a 6 dimensional space. Our implementation uses the simplex method as the optimization procedure to find a global minimum of the landscape, [NM65].

The ICP method can easily be extended for multiple devices. Each device has its own device transformation matrix (M in equation 4). The rest of the problem formulation remains unchanged; only the number of points used by the distance function increases. and the number of dimensions in the multi-dimensional landscape increases.

Dealing with noise Without noise, the global minimum of this landscape would be zero. However, with noisy data, the minimum value of the distance function is not a-priori known. A threshold value T_{2d} is used as an upper bound to verify if a minimum can be considered to be the global minimum. In our system we have chosen a threshold value of $T_{2d} = 0.9$ pixels. Geometrically, this may be interpreted as that the average distance between blobs and projected model points may not exceed T_{2d} pixel.

The choice of T_{2d} is related to the required accuracy of the found pose. A small T_{2d} will result in an accurate pose, but it may require a substantial amount of time to find the global minimum.

4. Test environment

Synthetic data sets were generated in a two step process. In the first step, a two handed interactive session was recorded and the pose of each device was stored on disk. This sequence of poses is used as the *reference data set*.

Our experiments were conducted in the Personal Space Station (PSS) [MvL02], is a near-field desktop environment (see figure 2). The PSS uses a mirror-based display: the user is seated in front of the mirror which reflects the stereoscopic images of the virtual world as displayed by the monitor. The visual space and interaction space are co-located: the user can reach under the mirror into the virtual world without obscuring the image or colliding with the monitor.

Two progressive scan CCD-cameras with wide-angle lenses (focal length of 3.6 mm) cameras are mounted on the chassis, providing an outside-to-inside view of the interaction volume. The cameras can take 60 images per second. The approximately 50x50x50 cm interaction volume is illuminated by rings of IR LEDs mounted around the camera lenses. Retro-reflective markers are pasted onto the interaction devices. IR light is reflected by the markers into the lens. Zhang's method for camera calibration and parameter estimation is been used, [Zha00].

A two handed interactive session was recorded of a user exploring a molecule (see figure 2). The recorded data set consists of 2200 time steps. Figure 4 plots the trajectories of the device frames during the interactive session. The left plot shows the trajectories in 3D, the right plot shows the projections of the trajectories on the XY, XZ, YZ planes of the interaction volume.

4.1. Creation of Synthetic Data Sets

In section 3.1 we identified three sources of errors that effect the inputs to the algorithms. To simulate errors arising from lighting conditions, we project the reference data set using the reference camera descriptors and add zero mean

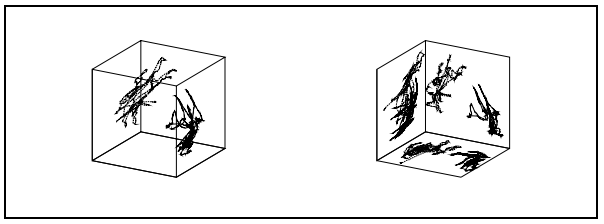

Figure 4: *Trajectories of two devices in 3D (left) and their projections (right).*

Gaussian noise with rms σ to each coordinate of the projected points. Three, experiementally derived, values for σ were chosen: 0.05, 0.20, 0.50.

To simulate errors arising from camera calibration, we perturb the intrinsic and extrinsic parameters of the reference camera descriptors. A new data set is generated for each perturbation. For the intrinsic parameters we perturb the focal length with 1%, 2.5% and 5% of the focal length of the reference cameras. For extrinsic parameters we rotate the cameras around their 'line of sight axis' (the Z-axis) by 1 and 2.5 degrees. Also, we translate the cameras in the XY plane by 1 and 2.5 cm.

This results in $3x3\ x\ 2\ x\ 2\ =\ 36$ data sets.

The experiments were performed on a single PentiumIV PC @ 1.6 Ghz.

5. Experimental Comparison

5.1. Accuracy

In these experiments, the relative accuracy of each algorithm was analyzed. This is by realized by comparing the difference between device poses of the reference data set with the poses computed by the algorithm.

We define the translation accuracy as the norm of the difference between the translation vector of the computed pose and the translation vector of the reference pose; i.e.

$$T_i^{err} = ||T_i^{ref} - T_i^{alg}|| \tag{6}$$

where T_i^{ref} is the translational error of the reference data set at frame i. The orientation accuracy is defined as the norm of the difference between the quaternion of computed pose and the quaternion of the reference pose:

$$Q_i^{err} = ||Q_i^{ref} - Q_i^{alg}|| \tag{7}$$

where Q_i^{ref} is the quaternion of the reference data set at frame i. It can be argued that an orientation accuracy metric based on angles would be more appropriate. We use quaternion differences since others have used this metric as well.

The root mean square (RMS) error of the translation and orientation accuracy is defined as $RMS_t = \sqrt{(\sum_1^N T_i^{err})/N}$ and $RMS_\omega = \sqrt{(\sum_1^N Q_i^{err})/N}$, with N the number of poses in the sequence.

Figure 5 show the translation and orientation accuracy plots of the distance matrix and ICP algorithms when zero mean Gaussian noise was added to the reference data set. Plots for σ = 0.05 (blue plot), 0.20 (green plot) and 0.50 (red plot) are shown. The plots show what is to be expected: adding increasing levels of Gaussian noise to the 2D points result in more noise in the device pose. Note however, that there is more noise in the distance matrix plots than in the ICP plots.

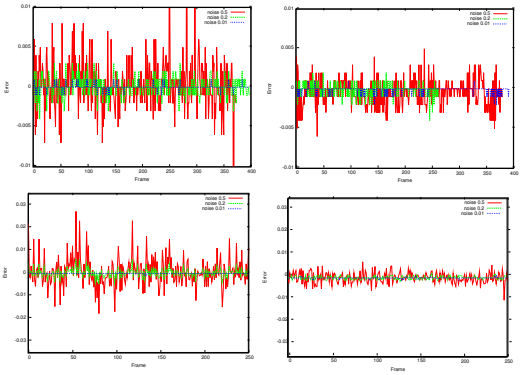

Figure 5: *Relative translation (top row) and orientation (bottom row) pose accuracy for the distance matrix (left col) and ICP (right col) trackers. Three noise levels are shown; σ = 0.05 (blue plot), 0.20 (green plot), 0.50 (red plot). pixels.*

Table 1 tabulates the RMS_t and RMS_ω for three noise levels.

2D noise	Dist Matrix	Proj Inv	ICP
0.05	3e-5 / 5e-3	3e-5 / 5e-3	2e-5 / 3e-4
0.20	4e-4 / 8e-3	1e-4 / 7e-3	8e-5 / 7e-3
0.50	5e-4 / 2e-2	3e-4 / 2e-2	1e-4 / 9e-3

Table 1: *RMS positional/orientation error for noise levels σ = 0.05, 0.20, 0.50*

It can be seen from figure 5 and table 1 that the accuracy of devices poses computed with the distance matrix and projective invariant trackers are more sensitive to 2D noise than the ICP trackers. This observation can be attributed to the nature of the algorithms; the distance matrix and projective invariant algorithms reconstruct a pose from corresponding point patterns in data. Noisy data will give noisy poses. On the other hand, ICP iteratively projects a pose and uses a distance fitting function until a predefined threshold distance is reached. The projection of the pose is fit into 2D data points.

If the device description is accurate then this will result in a more accurate pose.

5.2. Latency

In these experiments, the tracker latency of each algorithm was analyzed. In section 3.1, we defined latency as $t_3 - t_1$, the time needed by the tracker to do the image processing and pose determination.

Table 2 tabulates the average frame rate of each method, subject to varying noise levels and for the reference data set. The distance matrix and projection invariant methods are clearly faster than the ICP method. Also, it can be seen that performance of the distance matrix and projection invariant method is not sensitive to noise. The ICP method suffers from noise, since more optimizations are needed to find the global minimum.

2D noise	3D DistMat	ProjInv	ICP
0.00	100	50	16
0.05	100	50	10
0.20	100	50	12
0.50	100	50	3

Table 2: *Frame rate for reference data set and different noise levels.*

Figure 6 plots the accumulative latency for each algorithm in the case of no additional noise. The slopes of the distance matrix and projective invariant plots are practically constant The slope of ICP plot is often constant, but has a few large discontinuities. The reason for this is that the initial pose needed by the ICP method was not determined correctly, resulting in many optimizations before the minimum is approximated.

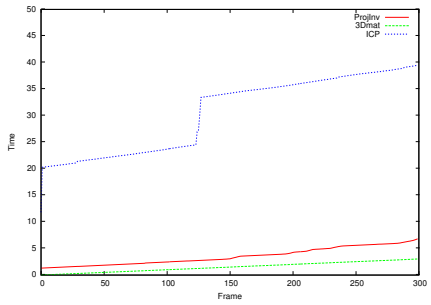

Figure 6: *Accumulative time for 3D matrix (red), projective invariant (green), and ICP (blue) trackers. No noise.*

5.3. Robustness

In these experiments, the robustness of the methods are analyzed. We define a tracker to be robust if the pose of all devices can be determined. If a device cannot be found – due to occlusion, or because the pattern cannot be found in the data – then the method is not considered robust. For each data set we count the number of times that a device is not found.

Table 3 tabulates the robustness of each method, subject to various noise levels and camera perturbations. For the case of additional noise on the 2D points, it can be seen that the ICP method is more robust than the distance matrix and projection invariant methods.

The ICP method is not robust when the focal length is mis-calibrated by more than 1% of the reference focal length. The distance matrix and projection invariant methods are not robust when focal length is mis-calibrated by more than 2.5%.

All methods are robust to small extrinsic camera parameters errors. However, when camera orientations are mis-calibrated by more than 1 degree then all methods fail. Similarly, all methods fail when camera positions are mis-calibrated.

		Dist Matrix	Proj Inv	ICP
reference		2 / 0	2 / 0	3 / 1
2D noise	0.05	2 / 0	2 / 0	3 / 0
	0.20	2 / 0	7 / 1	2 / 0
	0.50	43 / 9	77 / 24	3 / 25
intrinsic	1%	2 / 0	2 / 0	3 / 1
focal leng	5%	44 / 0	75 / 55	300 / 300
extrinsic	1 °	2 / 0	2 / 0	2 / 13
rotation	2.5 °	272 / 300	2 / 0	300 / 300
extrinsic	1 cm	2 / 0	2 / 0	3 / 12
translation	2 cm	2 / 0	2 / 0	3 / 288

Table 3: *Robustness for varying noise levels and intrinsic/extrinsic camera parameters.*

6. Discussion

In the previous sections we have compared three model based optical tracking algorithms. We summarize the results by discussing the three main comparison criteria:

- Accuracy
  The experimental data shows the relative accuracy of the ICP is better than the projective invariant and 3D distance matrix methods.

The reason for this can be explained as follows. The projective invariant and 3D distance matrix methods use the image data to perform model correspondence and then perform pose computation using the computed 3D data points. Noisy features will result in noisy 3D data points and a noisy pose. On the other hand, the ICP is an iterative method that computes the best fit of the model onto the feature data. If the device description is accurate and the threshold criteria is small, the optimization will find a minimum and a more accurate pose will be found.

- Latency
  The data shows that the projective invariant and 3D distance matrix methods have a lower latency than the ICP method. The complexity of the first two methods is related to the number of 2D features; i.e. the projective invariant is $O(N^5)$ whereas the 3D distance matrix is $O(N^2\ log(N))$ in which N is the number of 2D features. On the other hand, the ICP is an optimization method; i.e. for two devices, ICP will search a 12 dimensional space for a global minimum. It may take a very long time to find a global minimum in such a high-dimensional space, However, the probability that the optimization will rapidly converge increases when an initial pose is chosen near to the global minimum. The slope of the plots in figure 6 show that ICP is competitive with the other methods when an initial pose can be chosen near the global minimum.
- Robustness
  The robustness of the model-based trackers when processing noise corrupted data is characterized by two factors: the probability of detecting a model feature in the data, and the number of false detections. These two measures are intimately correlated; increasing the probability of detection (by increasing threshold intervals) will inherently increase the number of false detections. False detections should be avoided at all costs, since they will result in an inaccurate pose. The robustness of each tracker is thus closely related to the threshold values.
  For the same reason as discussed in the accuracy item, the ICP tracker is less sensitive to noisy 2D data than the distance matrix and projective invariant tracker.
  Incorrect camera calibration greatly influences the robustness of all methods. The ICP method is sensitive to camera mis-calibration, since it relies on an accurate projection of a chosen pose. Even slightly mis-calibrated cameras have grave effects on its robustness.
  Camera mis-calibration will result in inaccurate epi-polar geometry computations. The distance matrix method uses epi-polar geometry to transform 2D points into 3D. Inaccurate epi-polar geometry computations will result in corrupted 3D points. This is particularly apparent for small perturbation in camera rotations.

Occlusion is an inherent problem in optical tracking. The probability of occlusion increases when the number of devices increases; e.g. in the case of two-handed tasks. A valid approach to address occlusion is adding more cameras. All algorithms discussed in section 3.3 can be extended to incorporate multiple cameras. The projective invariant and 3D distance matrix algorithms could determine the pose of a device if the model features are recognized in two of the views. The distance function in the ICP tracker can be extended to incorporate all features simultaneously.

A disadvantage of using point patterns in the projective invariant and 3D distance tracker is that the pattern recognition requires a complete pattern. If one point in the pattern is occluded, then the pattern cannot be detected. In addition, the distance matrix tracker requires that the pattern must be visible in both images, since epi-polar geometry is used to transform points into 3D. The ICP tracker does not suffer from these disadvantages.

It should also be recognized that the synthetic data sets have been generated from one reference set. It may well be that the results will differ if the reference data set has different characteristics. A similar argument can be made about the lighting conditions. The synthetic data sets were generated by adding zero mean Gaussian noise to the reference data set. An alternative would be to add a bias factor to represent more realistic lighting problems.

7. Conclusions

In this paper, we have compared three model based optical tracking algorithms for pose determination. We have studied the behavior of these algorithms when applied to two-handed manipulation tasks. We have experimentally show how critical parameters influence the relative accuracy, latency and robustness of each algorithms.

Each algorithm has its advantages and disadvantages and it will depend on the specific virtual environment which is to be preferred. The accuracy of the ICP method is less sensitive to environmental noise, but too slow when an initial pose does not exist. The distance matrix and projective invariant methods are very fast, but are sensitive to noise. We recommend a hybrid approach which combines the advantages of both methods: use the distance matrix method to determine an initial pose and use this as input for the ICP method.

Acknowledgment

This work has been partially funded by the Dutch BSIK project 03019: Virtual Laboratory for e-science (VL-e).

References

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

[BM92] BESL P., MCKAY N.: A method for registration of 3d shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence 14*, 2 (1992), 239–256.

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

[ELF95] EGGERT D., LORUSSO A., FISHER R.: A comparison of four algorithms for estimating 3-d rigid transformations. In *British Machine Vision Conference* (1995), pp. 237–246.

[FP03] FORSYTH D., PONCE J.: *Computer Vision A Modern Approach*. Prentice=Hall, 2003.

[Lau01] LAURENT M.: *Matrix completion problems*, vol. III. Kluwer, 2001, pp. 221–229.

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

[NM65] NELDER J., MEAD R.: A simplex method for function minimization. *Computer Journal 7* (1965), 308–311.

[RL01] RUSINLIEWICZ S., LEVOY M.: Efficient variants of the icp algorithm. In *3rd International Conference on 3D Digital Imaging and Modeling (3DIM 2001)* (2001), pp. 145–152.

[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.

[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.

[WF02] WELCH G., FOXLIN E.: Motion tracking: No silver bullet, but a respectable arsenal. *IEEE Computer Graphics and Applications 22*, 6 (November/December 2002), 24–38.

[Zha00] ZHANG Z.: A flexible new technique for camera calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence 22*, 11 (2000), 1330–1334.