EGVE Symposium (2008)
B. Mohler and R. van Liere (Editors)

A Framework for Performance Evaluation of Model-Based Optical Trackers

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

[1]Center for Mathematics and Computer Science (CWI), Amsterdam, The Netherlands
[2]Eindhoven University of Technology, The Netherlands

Abstract

We describe a software framework to evaluate the performance of model-based optical trackers in virtual environments. The framework can be used to evaluate and compare the performance of different trackers under various conditions, to study the effects of varying intrinsic and extrinsic camera properties, and to study the effects of environmental conditions on tracker performance. The framework consists of a simulator that, given various input conditions, generates a series of images. The input conditions of the framework model important aspects, such as the interaction task, input device geometry, camera properties and occlusion.
As a concrete case, we illustrate the usage of the proposed framework for input device tracking in a near-field desktop virtual environment. We compare the performance of an in-house tracker with the ARToolkit tracker under a fixed set of conditions. We also show how the framework can be used to find the optimal camera parameters given a pre-recorded interaction task. Finally, we use the framework to determine the minimum required camera resolution for a desktop, Workbench and CAVE environment.
The framework is shown to provide an efficient and simple method to study various conditions affecting optical tracker performance. Furthermore, it can be used as a valuable development tool to aid in the construction of optical trackers.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Virtual Reality H.5.2 [Information Interfaces and Representation]: Input Devices and Strategies

1. Introduction

Tracking in virtual and augmented reality is the process of identifying the pose of an input device in the virtual space. Model-based optical tracking achieves this by using input devices augmented by markers for which the 3D features of these markers are known in advance. The set of known 3D features is called the model. Pose estimation is then performed by detecting these features in one or more two-dimensional camera images. Optical tracking is an important technology as it provides a cheap tracking solution that does not require any cables in the virtual space. Furthermore, given sufficient camera resolution, the accuracy of optical tracking is very good. However, a common and inherent problem in optical tracking is that line of sight is required: if the input device is partially occluded a pose can often not be found. Various implementations of optical trackers exist (eg. [KB99, WS07, SvRvL07, vR06]).

An important issue in optical tracking is an objective way of measuring performance. The user's performance for an interactive task often depends on the performance of the optical tracking system. Tracker accuracy puts a direct upper-bound on the level of accuracy the task can be performed with. Particularly in cases where the tracker cannot detect the input device, for example due to occlusion, interaction performance is reduced significantly. Therefore, many aspects must be taken into account when evaluating the performance of an optical tracker. These aspects include the type of interaction task that is performed; the intrinsic and extrinsic camera parameters, such as focal length, resolution, number of cameras and camera placement; environment conditions in the form of lighting and occlusion; and end-to-end latency. Furthermore, performance can be expressed in a number of different ways, such as positional accuracy, orientation accuracy, hit:miss ratio, percentage of outliers and critical accuracy, among others. Most optical tracker descriptions do not take all these aspects into account when describing the tracker performance.

In this paper, we present a framework for evaluating the performance of optical trackers in a systematic way. The presented framework allows us to quantitatively:

- Evaluate and compare the performance of different optical trackers under various conditions. This is useful for deciding which optical tracker implementation to use for a specific virtual environment, under different constraints.
- Study camera properties for various virtual environments. In this way, we can evaluate how many cameras are required to perform a specific task, what the minimum required quality of the cameras should be in terms of resolution, distortion and focal length, and where they should be placed.
- Study environment conditions for various virtual environments. This allows us to study the effects of device occlusion, which is an important aspect for optical tracking. Also, different lighting conditions can be studied, such as infrared, office or day light.

2. Related Work

Van Liere and van Rhijn [vLvR04] examined the effects of erroneous intrinsic camera parameters on the accuracy of a model-based optical tracker. They recorded a real, interactive task and subsequently ran three different optical tracking algorithms on these images, providing them with varying intrinsic camera parameters to simulate errors in the camera calibration process. They showed how these parameters affect the accuracy, robustness and latency of the tested optical tracking algorithms. The framework presented in this paper is more general in the sense that it enables us to study many more parameters than just the intrinsic camera calibration. Since we generate virtual camera images, it is possible to realistically study effects such as lighting conditions, occlusion and varying camera placements.

In the past, several techniques have been proposed to study the properties of multiple camera setups and camera placements for general optical tracking. Two examples are the Pandora system by State, Welch and Ilie [SWI06] and the work by Chen [Che00] on camera placement for robust motion capturing. The Pandora system [SWI06] allows the user to set varying extrinsic and intrinsic camera parameters and projects a visualization of these parameters on a virtual scene. Every virtual camera projects a resolution grid on the scene using shadow mapping. In this way, the user can explore the virtual scene and examine, for example, which parts of the scene are visible and at which resolutions for different camera placements. Chen [Che00] proposes a quantitative metric to evaluate the quality of a multi-camera configuration in terms of resolution and occlusion. A probabilistic occlusion model is used and the virtual space is sampled to determine optimal camera placements. The focus lies on finding a robust multi-camera placement for general motion capturing systems.

These methods share the property that they take into account only the camera placements and the virtual space, but do not provide any performance measures for specific optical trackers or tasks. A typical strategy is to maximize the amount of space coverage while maintaining a pre-specified minimum resolution. Our framework is focused on model-based device tracking in virtual environments, as opposed to the larger problem of general optical tracking and space coverage. We seek to provide quantitative data for real tasks so we can compare the actual, measured performance of different optical trackers under varying conditions, which are not limited to camera placement alone.

3. Methods

In this section we provide a detailed description of our proposed framework for the performance evaluation of model-based optical trackers. First, the various components of the framework are discussed, along with some examples of typical usage scenarios. Next, a brief description is given of the implementations of two optical trackers. These optical trackers will be used in Section 4 as examples to illustrate how the presented framework can be used in practice.

3.1. Framework Description

An overview of the proposed framework is given in Figure 1. The framework consists of three major components: the simulator, the optical tracker and the analysis component. Solid arrows between varying components represent the flow of data. The simulator is responsible for generating two-dimensional image files, which are then used as input for the optical tracker. Next, the optical tracker calculates a pose based on these input images. Finally, the calculated pose is fed to the analysis component and compared to a ground truth resulting in various kinds of performance metrics.

Each major component accepts one or more input data streams generated by supporting components, which are shown in Figure 1 to be environment conditions; an occlusion model; camera parameters, both intrinsic and extrinsic; a ground truth in the form of a task; and a device model. By varying the output of the supporting components we can evaluate the performance of optical trackers under a wide range of different conditions. We will now describe each of the three main components and their inputs in more detail.

The purpose of the simulator component is to output a series of two-dimensional images representing captured camera frames. In order to achieve this, several input data are required:

- A device model that is recognized by the optical tracker.

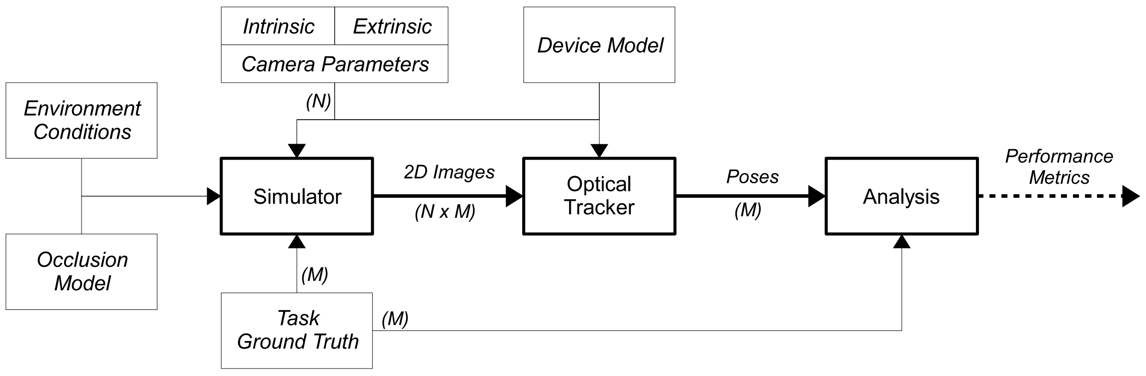

Figure 1: *An overview of the presented framework for the performance evaluation of model-based optical trackers. The three main components are the simulator, optical tracker and analysis components. Each of these components receive one or more input data streams from supporting components. The input data provided by these supporting components can be varied to evaluate tracker performance under a wide range of conditions. Data flows are indicated by solid arrows between components.*

This enables us to render a simulation of the input device, which can then subsequently be tracked by the optical tracker.

- Intrinsic and extrinsic parameters for N virtual cameras. For each of the input cameras an image is rendered as if the device model was captured with such a camera. In this way, we can modify camera placements and the number of cameras, along with camera properties such as focal length and resolution.
- Simulated environment conditions used for rendering. By altering environment conditions we can modify the rendered images in various ways. Some examples of environment conditions are lighting models, such as infrared, office or day light; background images, which can complicate the required image processing; and different types of generated image noise.
- An occlusion model where the input device is partially occluded for different cameras. Occlusion is an important aspect for optical trackers, as line-of-sight is always required to determine a pose. Examples of occlusion models are the user's hands on the input device, or even people's bodies in larger, CAVE-like virtual environments. Simulated occluders can be rendered in addition to the input device, making it more difficult to detect.
- Ground truth data of where exactly the input device is located. A ground truth position and orientation allows us to render the device in that exact location, resulting in an animation sequence of M ground truth device poses. A number of different ways are possible to generate this ground truth location data. A synthetic signal can be used to evaluate different positions and orientations in the environment; however, it is also possible to record and replay a real-life interaction task. Several different types of interaction tasks can be recorded in this manner, such as positioning or pointing. Finally, different sampling schemes can be used to simulate varying camera update rates.

The output of the simulator component is a series of NxM images: for each of the M ground truth poses, N camera images are rendered. Rendering is performed using standard OpenGL functionality; a 3D polygon mesh of the occlusion model is rendered, along with a texture mapped cube representing the input device. The six textures for the rendered input device are obtained through digital photographs of a real input device. The extrinsic and intrinsic parameters for the virtual camera are modelled by the OpenGL modelview and projection matrix, respectively. The camera focal length is represented by entries $(0,0)$ and $(1,1)$ of a 3x3 projection matrix, with the principal point at entries $(0,2)$ and $(1,2)$. Focal length values are computed in such a way as to match those of actual 35mm equipment. The virtual camera's principal point is set to $((w+1)/2,(h+1)/2)$, where w and h represent the camera's horizontal and vertical resolution, matching the principal point used by OpenGL for rendering. The modelview matrix is a standard 4x4 transformation matrix for camera orientation and position. These values are similar to the calibration values of a real camera. Given this virtual camera setup and a rendered simulation image, the tracker component can subsequently reconstruct a 3D ray in the simulated interaction space from the virtual camera through a given a 2D image point.

The optical tracker component is responsible for calculating a device pose based on the images rendered by the simulator component. In order to achieve this, the tracker receives additional input consisting of the same device model description that was provided to the simulator, and also with the same intrinsic and extrinsic camera parameters. This is equivalent to the situation where calibrated, real cameras provide captured images. Alternative approaches may restrict this input, for example, in the case of evaluating markerless tracking, structure from motion approaches or camera calibration. The optical tracker component itself can be implemented by many different types of optical trackers that

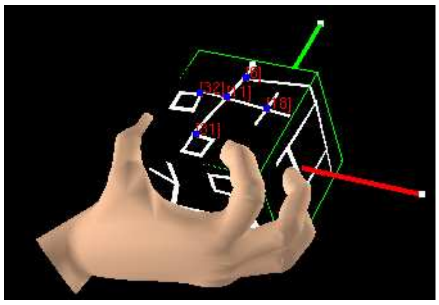
Figure 2: *The simulated input device for GraphTracker. The graph markers are rendered in white to simulate infra-red lighting. The 3D hand occlusion model is also shown.*

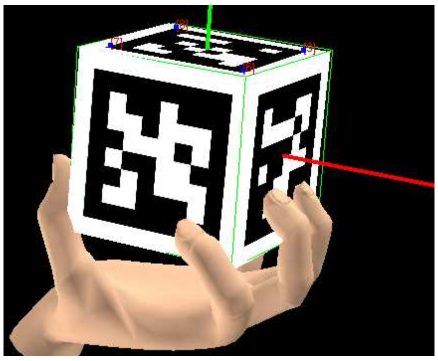
Figure 3: *The cubical input device for the optical tracker based on ARToolkitPlus. The 3D hand occlusion model is also shown.*

we wish to evaluate under certain conditions. The output of the optical tracker component is a device pose, which can be compared to the ground truth pose.

The purpose of the analysis component is to provide various performance metrics based on the pose reported by the optical tracker component and the known ground truth. Therefore, the input to the analysis component consists of the same ground truth animation sequence that was provided to the simulator component and the sequence of poses generated by the optical tracker. The output of the analysis component can be a wide range of performance metrics, such as position and orientation accuracy, for different animation frames or cameras, either per frame or averaged over the entire sequence, hit/miss ratios or processing time. Additionally, the animation sequence may contain weights to describe specific frames where accuracy is deemed to be much more important than at other frames, for example at interaction locations near a target. The output of this analysis can be compared for different settings to determine which setting is best suited for a specific task.

3.2. Tracker Implementations

In order to test the presented framework, we have made use of two different optical trackers in Section 4. In this section we will briefly describe the technical details of these two optical trackers; the reader is assumed to be familiar with optical tracking techniques. Both trackers are structured in such a way that they consist of two components: a 2D-to-3D point correspondence component and a 3D pose reconstruction component. The correspondence component is responsible for detecting 2D image features and mapping them to 3D device model features. The pose reconstruction component uses this correspondence information to calculate a device pose.

The first optical tracker is an implementation of GraphTracker as presented by Smit et al. [SvRvL07]. The correspondence component operates by detecting projection invariant graph structures in a 2D image. A sample input device is shown in Figure 2. The vertices of these detected graphs are considered to be image features and are mapped to 3D model points. Since feature correspondence is established in a projection invariant manner, only a single camera is required, but more cameras are allowed. In fact, correspondence is established for every input camera image, and the combination of these correspondences is used as input to the pose reconstruction component.

The pose reconstruction component first determines a device pose for each camera individually using the efficient perspective-n-point algorithm [MNLF07], modified to use Horn's absolute orientation method [Hor87]. For each pose found in this way, the reprojection error is determined and the pose with the smallest error is used as a starting pose to an iterative procedure using all cameras. This iterative procedure is an extension for multiple cameras by Chang et al. [CC04] to the iterative pose reconstruction by Lu et al. [LHM00]. Further enhancements would be the use of Micheals-Boult absolute orientation determination [Mic99] and a modification of the iterative procedure to support the robust planar pose algorithm [SP06]; however, these enhancements are currently not implemented.

The second optical tracker is a modified version of ARToolkitPlus by Wagner and Schmalstieg [WS07], which in turn is based on ARToolkit by Kato and Billinghurst [KB99]. ARToolkitPlus is capable of detecting different, square markers in a single camera image. We have constructed a cubical input device with different markers on each of the six cube sides as shown in Figure 3. Next, ARToolkitPlus is used to establish feature correspondence between the four 2D corners of each square marker and the 3D positions of these corners as given by the device model. For each individual camera image, the set of 2D to 3D feature correspondences is provided to the same pose reconstruction component that was described previously in the case of GraphTracker.

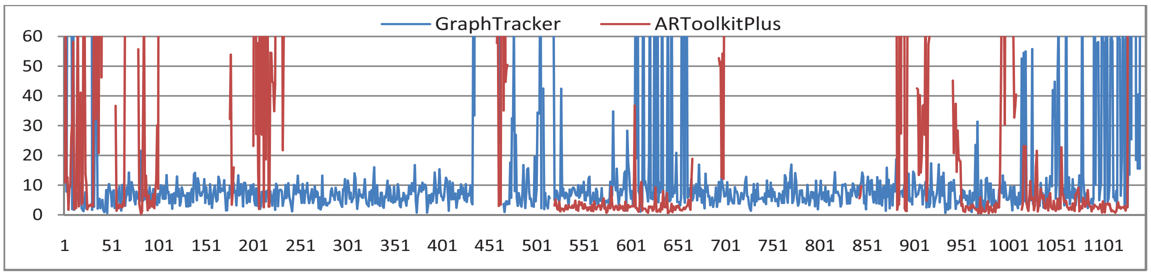

Figure 4: *Positional accuracy in millimeters for all animation frames. Gaps indicate a pose could not be found.*

	Hit%	Position Err.	Orientation Err.
ARToolkitPlus	45%	3.4 mm	3.2 deg
GraphTracker	97%	7.0 mm	10.8 deg

Table 1: *Summarized results for the first experiment. The table shows the percentage of hits and the median positon and orientation errors for the entire task. GraphTracker detects a pose more often; however, ARToolkitPlus is often more accurate when it detects a pose.*

4. Results

In this section we show three sample uses of our framework to evaluate different aspects of optical tracker performance. In the first experiment, the accuracy of two different optical trackers is compared, given a fixed environment. In the second experiment, we try to determine the optimal camera placement by varying the extrinsic camera parameters. Finally, in the third experiment, we evaluate the effect of camera resolution and distance by varying the intrinsic camera parameters as well. These experiments provide answers to interesting sample problems for optical trackers; however, the framework presented is not limited to these experiments alone.

We want to stress the fact that the presented results are not intended to provide an answer to the question of which optical tracker performs strictly better than the other. Instead, these results should be looked upon as examples of the usage of the presented framework to determine and compare the performance of optical trackers over a broad spectrum of conditions. Each optical tracker has its own pros and cons, and the decision of which optical tracker is better suited for a particular task at hand should be based on the combination of relevant metrics and properties, rather than on a single statistic. This will be discussed further in Section 5.

4.1. Experiment 1: Tracker comparison in a fixed environment

The first experiment is a comparison between the two optical trackers, GraphTracker and ARToolkitPlus, described in Section 3.2. For this experiment, we will vary the implementation of the optical tracker component and, thus, the device model. All other components are kept constant. In both cases the virtual input device consists of a cube with edges of 7cm. The question asked for this experiment is, "Which optical tracker should be used for this particular environment and task?"

For the camera parameters we use a setup typically used for the PSS [MvL02], a near-field virtual environment. We use a dual camera setup above the virtual working space, pointing at the origin at a distance of approximately 55cm. The focal length of the cameras is set to 28mm wide angle in terms of 35mm full-frame photographic equipment. Environment conditions are set for infrared lighting. To simulate this, the device markers are rendered in black and white. To simplify processing, the background is kept strictly black and no image noise is added. The occlusion model consists of a fixed 3D model of a human hand that is attached to the cubical input device, simulating the user holding the input device. This was shown in Figure 3. The hand model remains static with respect to the cube's orientation. The task, or ground truth, is a pre-recorded animation sequence of a user executing a real-life task in a near-field virtual environment with a similar cubical input device. This task was recorded using a magnetic tracker so occlusion has no effect in establishing the ground truth. The animation sequence consists of 1146 frames sampled at regular intervals. The combination of all these conditions provides a reasonable simulation of a real near-field virtual environment.

Figure 4 shows the positional accuracy in millimeters for each of the 1146 animation frames and for both optical trackers. Summarized results are given in Table 1. For this fixed camera setup, the percentage of hits for GraphTracker is relatively high at 97% despite the occlusion, whereas the hit percentage for ARToolkitPlus is only 45% due to occlusion. However, whenever a pose is detected by ARToolkitPlus, the reported pose is generally more accurate than the pose reported by GraphTracker. These results suggest that GraphTracker is the better choice for this particular task and environment; however, this may no longer be the case when we are free to reposition the cameras. Furthermore, if we are willing to sacrifice hit:miss ratio in favour of enhanced accuracy, then ARToolkitPlus would be the tracker of choice.

4.2. Experiment 2: Determining optimal camera placement

The second experiment uses a similar setup as the first experiment, only this time the extrinsic camera parameters and the number of cameras will be varied. All other conditions, such as the occlusion model and task, are kept constant as before. By varying the extrinsic camera parameters we can determine an optimal camera placement for this particular task and setup. The question here is, "Which camera placement is best for the execution of this particular task?"

To answer this question we have randomly generated 200 sets of camera placements. This was done by taking the camera distance to the origin of the near-field camera setup of experiment 1, and randomly placing cameras on the quarter sphere on top and in front of the workspace defined by this distance. In this way, we generated four different batches of 50 camera sets. The first batch consists of stereo pairs of two cameras. For every set the two cameras were constrained to have a distance between 5 and 15cm. The second batch consists of two cameras per set as well; however, the distance constraint was changed so the cameras must be further than 30cm apart. The third batch consists of sets of two stereo-pairs at a distance greater than 30cm. Finally, the fourth batch consists of four cameras, all at a distance larger than 30cm from each other.

Next, for all camera placements, the 1146 frame task was executed. For every camera placement we registered the average position and orientation accuracy for the task, as well as the hit:miss ratio. These results are shown in Figure 5 for the batches of two cameras. In Table 2 all data is summarized by calculating a subsequent average over specific batches of camera sets for the position and orientation accuracy. A number of observations can be made from this data. For two cameras in stereo-setup (N<50 in Figure 5), ARToolkitPlus achieves very poor hit:miss ratios due to occlusion. GraphTracker achieves somewhat better hit:miss ratios; however, the pose accuracy is generally worse than that of ARToolkit in the case it does detect a pose. For two independent cameras (N>50 in Figure 5) the his:miss ratios improve for both ARToolkitPlus and GraphTracker, but the pose accuracy is often reduced, especially for GraphTracker and the position accuracy of ARToolkitPlus. This leads us to believe that stereo camera setups are generally more accurate due to the difficulty of robust pose reconstruction in a single camera, but result in worse hit:miss ratios due to the smaller coverage of space. In the case of four cameras in the form of two stereo-pairs (see Table 2), ARToolkitPlus still shows some poor camera placements with respect to hit:miss ratio. However, for four independent cameras the occlusion problem is reduced significantly due to the large coverage of space. In this case both ARToolkit and GraphTracker are capable of finding a pose in most circumstances, and ARToolkitPlus achieves a much higher accuracy in doing so. It can also be seen that four cameras are generally more accurate than two cameras. These results suggest that if we are free to choose a specific camera placement, then ARToolkit would be tracker of choice for this particular task and environment.

4.3. Experiment 3: Determining minimum camera resolution requirements

In the third experiment we vary the intrinsic camera parameters by changing the resolution; furthermore, the extrinsic parameters will be modified by placing the cameras at different distances from the origin. These camera distances are representative for different types of virtual environments. The other conditions remain constant. This experiment may help in answering the question, "What is the minimum required camera resolution for optical tracking in this virtual environment?"

To determine camera positions for this experiment, we used the results of the second experiment for optimal camera placement of four cameras and chose the camera position set where the positional error was smallest. It turned out that these positions also provided good orientation accuracy, as well as a good hit:miss ratio. Next, we scaled these positions in such a way that we obtained three sets of cameras for distances of 0.75m, 1.5m and 3.0m from the origin. These distances are representative for near-field, Workbench- and CAVE-like virtual environments. Due to increased camera distance we increased the focal length of the cameras to a standard 50mm in terms of 35mm full-frame photography equipment. We then executed the 1146 frame task for each of these camera sets at camera resolutions of 512x384, 640x480, 800x600, 1024x768 and 1280x960.

The results of these measurements are given in Table 3. This table shows the hit percentage, average task accuracy in millimeters and the average orientation accuracy in degrees. It can be seen that the performance of GraphTracker is reduced significantly for larger viewing distances. The performance of ARToolkitPlus decreases with distance as well, but not as quickly as for GraphTracker. Generally, an increase of resolution results in increased accuracy and better hit:miss ratios. In order to use these trackers in a CAVE-like environment, with cameras placed at 3m distance, a minimum camera resolution of 800x600 is required for ARToolkitPlus (preferably 1024x768) and for GraphTracker a resolution of 1280x960 is still not sufficient. This shows that when designing optical trackers for such environments special considerations have to be taken into account. For environments like the Workbench, with camera distances of 1.5m, both trackers can be used; however, GraphTracker requires higher resolutions. Both trackers show comparable behaviour in near-field environments, where higher camera resolution is apparently an inefficient use of resources.

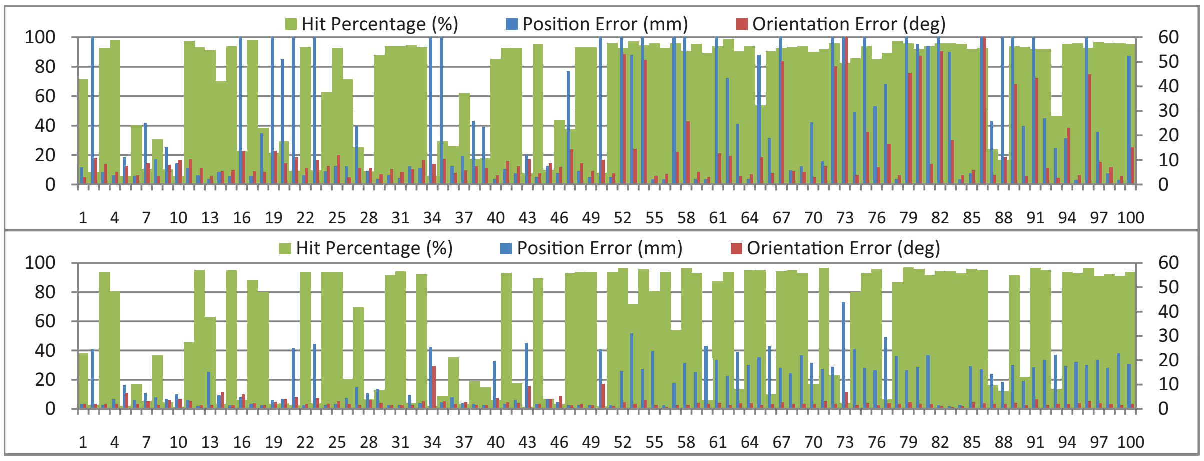

Figure 5: *Top: GraphTracker for sets of two cameras. Bottom: ARToolkitPlus for sets of two cameras. The first 50 sets consist of stereo pairs, whereas the second 50 sets consist of two independent cameras.*

	ARToolkitPlus				GraphTracker			
Placement Strategy	**Stereo**		**Free**		**Stereo**		**Free**	
Number of Cameras	**2**	**4**	**2**	**4**	**2**	**4**	**2**	**4**
Hit Percentage (%)	41%	76%	74%	95%	54%	90%	89%	96%
Average Position Error (mm)	6.7	2.9	17.6	6.8	21.1	4.2	59.8	7.7
Average Orientation Error (deg)	3.4	1.8	2.1	1.4	7.6	5.1	18.7	6.1

Table 2: *Summarized results for the 2nd experiment. The rows show, respectively, the average percentage of hits, position accuracy in mm and orientation accuracy in degrees, over all camera sets. The colums represent two and four cameras, both in stereo pairs and in a free setup.*

5. Discussion

In Section 4 we have shown a sample evaluation of two different optical trackers under varying conditions. By using the presented framework, it was possible to acquire performance metrics for different situations, such as varying camera placements, that would be difficult to acquire rapidly in practice. However, care must be taken not to judge tracker performance too rapidly based on a single experiment.

The accuracy for GraphTracker was shown to be generally worse than that of ARToolkitPlus. This may be due to the fact that the tested device model for GraphTracker has been constructed from a real input prop, using digital photographs as textures for the simulated device. Next, an automated model training step [SvRvL07] was performed using the presented framework to acquire the 3D model description. Therefore, the model description for GraphTracker is likely to contain small errors, whereas the model description for ARToolkitPlus was constructed in a purely synthetic, error-free manner. This illustrates that all factors must be considered when comparing trackers.

In the first experiment, GraphTracker showed a much better his:miss ratio for the given environment than ARToolkitPlus. If we are free to change the number of cameras and their placements, it can be seen from the second experiment that ARToolkit can be made to perform equally well in terms of hit:miss ratio for certain camera placements. However, these specific camera placements may only be valid for the particular task animation that was executed. For example, a right-handed user holding the input device in such a way that the left side is always completely visible probably benefits most from a camera placement with cameras on the left side. Now, if this user is replaced by a left-handed user, the same setup is likely to perform very poorly in terms of hit:miss ratio. The framework can be used to test many different types of tasks and determine a robust camera placement.

In the third experiment it could be seen that GraphTracker performed relatively poor for larger camera distances. This is due to the fact that GraphTracker has been specifically designed for near-field environments, while ARToolkitPlus was designed as a general marker tracking system. It may be possible to improve the image processing of GraphTracker to allow for larger camera distances. This shows that the framework can also be used as a valuable development tool: poor

	ARToolkitPlus			GraphTracker		
	0.75m	**1.50m**	**3.00m**	**0.75m**	**1.50m**	**3.00m**
512x384	96 / 1.11 / 0.88	93 / 3.29 / 3.76	0 / - / -	96 / 1.97 / 4.24	43 / 1220 / 103	0 / - / -
640x480	96 / 0.86 / 0.69	95 / 1.94 / 1.61	58 / 22.28 / 25.0	97 / 1.43 / 3.83	66 / 23.0 / 15.5	0 / - / -
800x600	97 / 0.68 / 0.54	95 / 1.52 / 1.30	86 / 9.20 / 12.95	97 / 1.34 / 3.60	94 / 5.97 / 9.09	16 / 2725 / 94.8
1024x768	98 / 0.52 / 0.42	96 / 1.15 / 0.97	93 / 3.51 / 4.0	98 / 1.26 / 3.25	97 / 2.86 / 4.84	38 / 2531 / 97.9
1280x960	98 / 0.41 / 0.33	96 / 0.89 / 0.76	95 / 1.98 / 1.71	98 / 1.14 / 2.90	98 / 1.53 / 3.91	67 / 23.75 / 16.8

Table 3: *Results for the 3rd experiment on camera resolution and distance. The rows represent the different resolutions, while for each tracker the columns represent the different camera distances to the origin. Every table cell shows the percentage of hits, the average position error in mm and the average orientation error in degrees, respectively. When the percentage of hits equals zero, accuracy information is not available.*

tracker performance can be detected for specific conditions, after which the tracker software can be updated to resolve these issues.

Most optical trackers have their own pros and cons, and the decision of which tracker to use should be based on a number of required factors, some of which may not have been tested. For example, GraphTracker is designed in such a way that it can handle any convex shaped input device, whereas ARToolkitPlus is restricted to planar surfaces. On the other hand, GraphTracker currently requires infrared lighting, while ARToolkitPlus is capable of operating in normal daylight. Before deciding on a specific tracker, one should carefully consider the requirements and the available environment.

The presented framework does not take end-to-end latency into account. While it is possible to measure execution times for the optical tracker component, the framework is unable to measure real, observed latency. In order to do so a number of additional factors need to be simulated, such as the cameras' CCD fill rates, data transmission times, application/simulation update rates and scene graph rendering rates. To be accurate, even the display refresh rate needs to be modeled, along with the difference in update rates between various application components. Since this information varies greatly among different applications, and in some cases may not be available, we chose not to provide any latency modeling in the presented framework.

6. Conclusion

We described a framework to evaluate optical tracker performance under various conditions. Three examples were shown where the framework was used to compare the performance of two different trackers under similar conditions, to determine optimal camera placements and to determine the minimum required camera resolution in various environments. The presented framework provided an efficient and simple method to study various conditions affecting optical tracker performance. Furthermore, it proved to be a valuable development aid for the construction and improvement of future optical trackers.

References

[CC04] CHANG W.-Y., CHEN C.-S.: Pose estimation for multiple camera systems. In *ICPR* (2004), pp. 262–265.

[Che00] CHEN X.: *Camera Placement Considering Occlusion for Robust Motion Capture*. Tech. rep., CGL Stanford, 2000.

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

[KB99] KATO H., BILLINGHURST M.: Marker tracking and HMD calibration for a video-based augmented reality conferencing system. In *Proc. of the IWAR* (Oct. 1999).

[LHM00] LU C.-P., HAGER G. D., MJOLSNESS E.: Fast and globally convergent pose estimation from video images. *IEEE Trans. PAMI 22*, 6 (2000), 610–622.

[Mic99] MICHEALS R.: A new closed-form approach to the absolute orientation problem. Masters thesis, Lehigh University, 1999.

[MNLF07] MORENO-NOGUER F., LEPETIT V., FUA P.: Accurate non-iterative O(n) solution to the PnP problem. In *Proc. of the ICCV* (2007).

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

[SP06] SCHWEIGHOFER G., PINZ A.: Robust pose estimation from a planar target. *IEEE Trans. PAMI 28*, 12 (2006), 2024–2030.

[SvRvL07] SMIT F. A., VAN RHIJN A., VAN LIERE R.: Graphtracker: A topology projection invariant optical tracker. *Computers & Graphics 31*, 1 (2007), 26–38.

[SWI06] STATE A., WELCH G., ILIE A.: An interactive camera placement and visibility simulator for image-based VR applications. In *Proc. of the SPIE* (Feb. 2006), vol. 6055, pp. 640–651.

[vLvR04] VAN LIERE R., VAN RHIJN A.: An experimental comparison of three optical trackers for model based pose determination in virtual reality. In *Proc. of EGVE* (2004).

[vR06] VAN RHIJN A.: *Configurable Input Devices for 3D Interaction using Optical Tracking*. PhD thesis, Eindhoven University of Technology, 2006.

[WS07] WAGNER D., SCHMALSTIEG D.: ARToolkitPlus for pose tracking on mobile devices. In *Proc. of the CVWW* (Feb. 2007).