Visual tracking for augmented reality: No universal solution but many powerful building blocks

Mario Becker, Gabriele Bleser, Alain Pagani, Yulian Pastarmov, Didier Stricker, Florent Vial, Jens Weidenhausen, Cedric Wohlleber, Harald Wuest
Fraunhofer for Computer Graphics (IGD)
Fraunhoferstr. 5
64283 Darmstadt
Tel.: +49 (0)6151 155 124
Fax: +49 (0)6151 155 196
E-Mail: {firstname.lastname}@igd.fraunhofer.de

Abstract: In this paper, we present an overview of several visual tracking methods for industrial augmented reality applications. We show that no universal algorithm can deal with the large number of possible scenes, and that the different methods have to be seen as complementary approaches that all have their strengths and weaknesses. The main difficulty, then, consists in combining existing building blocks in the right manner so that the overall system enables stable tracking. This paper addresses each phase of the tracking, i.e. Initialization, Tracking, Re-Initialization, and proposes a first choice of appropriate algorithms. Finally, a global system is designed, tested and evaluated with help of video sequences of different real environments.

Keywords: Systems, Augmented Reality, Visual Tracking, Tracking Architecture, Framework

1 Introduction

Tracking is a key technology for augmented reality applications, but still represents an unresolved problem, especially in the context of industrial environments, for which the external conditions, such as lighting, dust or scene occlusions, are uncontrolled and unpredictable. A lot of work has been dedicated to camera tracking in the computer vision and robotic communities, and many different approaches have been proposed. The first category of algorithms uses local features [HS88] detected in the images and tracked over the video sequences [ST94]. The 3D coordinates of the features are computed either during a preparation phase [GSN02] by reconstruction, or using an existing 3D-VRML (or CAD) model of the scene [LVT03], or in real time while tracking using structure and motion approaches [KPV99]. The second category takes advantage of the shapes of the objects and scenes to be tracked [MZS03]. These solutions are particularly interesting for objects without texture [CMC03]. The last category of approaches concerns the algorithms combining several sensors, such as a camera with an inertial sensor

[YNA99], or inside-out with outside-in cameras [SUYT03]. The advantage of such multi-sensors approaches consists in the combination of each component's strength. For example, an inertial sensor will be able to measure a fast rotational motion whereas the camera image will be blurred and unusable.

Nevertheless, most of the above-referred works address particular aspects of tracking and therefore provide a specific solution. But a tracking system should offer an overall approach, and address all tracking issues at a time (Initialization, Tracking and Re-Initialization) in order to be usable. Furthermore, it has to take into consideration many different scenes and situations, and therefore be designed in a very flexible way.

In this paper, we present an approach that relies on the combination of different computer vision algorithms. A generic tracking framework enables to choose and load different tracking actions dynamically, and to combine them at starting time. In our (very first) experiences, this system architecture seems to be appropriate and provides a solution to the large spectrum of augmented reality applications. The paper is structured as follows. Section 2 presents the different tracking algorithms – the building blocks - we envisaged. Section 3 describes the overall framework and section 4 gives results of the tracking system. Section 5 concludes the paper.

2 Building blocks

2.1 Overview

A tracking system for AR has to deliver the position and the orientation in a given reference coordinate system. Therefore, a priori knowledge about the scene acting as absolute reference information is required. From a computer vision point of view, this means that the tracking methods will have to rely on a given 3D representation of the scene, which can be a VRML model [CMC03] or reference images associated to a pre-computed camera pose [LVT03, BPS05, Str01]. An additional essential aspect of our approach is that we explicitly consider each step of the tracking process. We define these steps as follows:

Initialization: The Initialization-step delivers the first position and orientation of the camera in the defined scene coordinate system. It has to work without the intervention of the user and requires therefore pattern recognition and matching with the scene model.

Tracking: The Tracking-step propagates the camera pose from frame to frame, assuming a continuous video stream. Tracking relies on temporal coherence of the image content and can exploit information such as optical flow [ST94] to recover the image features and deduce the current camera pose. When abrupt motion arises, wide baseline matching algorithms are required.

Re-Initialization: The Re-Initialization-step recovers the camera pose when the tracking fails. We made the distinction between Re-Initialization and Initialization, since at Re-Initialisation-time new information about the scene may have been collected during the tracking phase what will facilitate the pose recovery.

2.2 Initialization

Several initialization methods are possible:

Initializing with markers: Markers are often used in augmented reality applications. They are well defined and can be easily found in images [KB99]. Ideally, just one marker has to be placed in the scene in order to get the first camera pose of the tracking session. The markers are used as anchors or absolute references, and allow so to determine the position and orientation of the VRML/CAD-model in the scene. This information is then used and maintained during the tracking phase (see paragraph 2.3 and 2.4)

Initializing with key frames: The basic idea of the initialization with key frames is to match in real-time the current video image with pre-defined reference images of the scene, and so to transfer 3D information about the scene to the current frame. The matching method can be implemented in various ways. We base our approach on local SIFT features [Low04] matched between the live video frame and the pre-calibrated reference images. At starting time, the initialization procedure performs feature extraction from the reference frame and back-projects all points onto the 3D model in order to obtain their 3D coordinates. The back-projection is done by sending rays from the related camera position through the image plane and computing the intersection points with the 3D model. Another possibility of back-projection is to render the CAD/VRML model on the graphics card. Then the depth-buffer (Z-buffer) can be read in order to retrieve the depth of the 2D image points and so the 3D coordinates. The result of both methods is a set of 2D/3D corresponding points associated to image features describing their local appearances. This information enables to compute the camera pose for the current image frame and provides image features to the tracking procedure. Implementation details about this approach can be found in [BPS05].

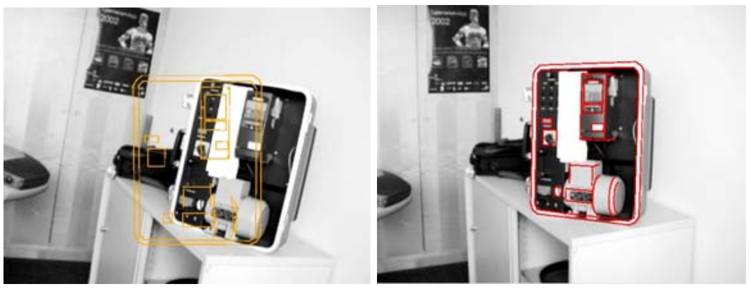

(a) (b)

Figure 1: Initialization with a line model: (a) initial coarse position/orientation (b) automatic registration

Initializing with a geometry model: The last approach we developed applies only the object/scene geometry model. The VRML model is converted into a line or contour model that is automatically matched in the image. The initialization procedure works as follows. An initial position/orientation of the model is defined. The user moves until the model approximately overlays the object in the image as illustrated in Fig. 1 (a). With help of the local search, the line model is accurately fitted into the image (Fig. 1 (b)). More details about the registration algorithm are given in section 2.4.

All three approaches require user interaction and scene preparation. In the case of the first two methods, the necessary engineering cannot be automated and represents time consuming work. In the case of the markers, they have to be placed in the scene and carefully measured. The initialization with key frames assumes a set of calibrated images for each object/scene view. Furthermore, the key frame approach relies on the assumption that the scene geometry and appearance is almost static. Strong illumination differences as it may occur for outdoor environments, partial changes in the geometry, or deterioration of the object surfaces will limit the initialization.

2.3 Tracking with local features

Tracking procedure: To obtain reliable 2D/3D correspondences from natural local features, different types of feature detectors [Low04, Bau00], descriptors [Bau00, MS03, KS04, Low04], alignment [ZGN04] and update methods [MTB04, MS03] can be applied. While the selection of the tracking actions has to be adapted to the requirements of the given application, the overall workflow of the tracking procedure remains the same. This one is represented in Fig. 2 and is implemented as a pipeline of abstract actions instantiated at running time (see section 3).

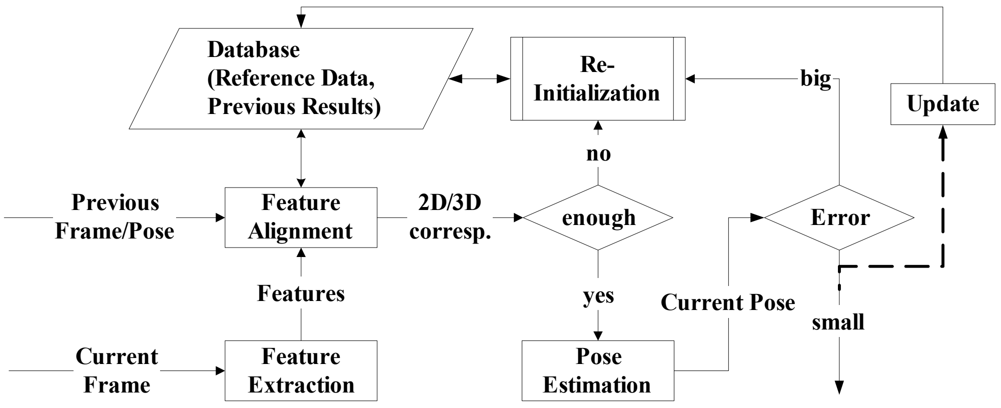


Figure 2: Overall architecture of the tracking procedure

The tracking procedure always handles two successive frames, the current camera frame and the previous one. Additionally, it makes use of a database containing reference data (3D model of the scene, calibrated reference images, confident 3D features) and previous calculation results (previous pose, updated features). If the tracking fails due to too few matches or to a badly defined pose, the Re-Initialization is invoked automatically. A successfully calculated pose is always employed to update the database, e.g. by back-projecting new features onto the 3D model to obtain 3D coordinates, thus handling occlusions or light changes. An important point is the reading and writing access of the feature alignment module to the database for either aligning against or updating reference data. Thus, the tracking is not only done in an iterative manner but also against confident reference features in order to avoid drift. Several implementation of this approach have been achieved and are presented in the next two paragraphs.

Abrupt camera motion: Tracking of a head mounted display in the context of a mobile AR-application is very challenging. Rapid camera movements in connection with high motion velocity generate large feature displacements, whereat no meaningful motion model can be assumed to give a good prediction of the current camera pose. In this case, it is essential to use robust feature detectors and descriptors that are invariant to Euclidian [Low04] or even affine transformations [MS02]. As the described boundary conditions are very similar to those defined for tracking-initialization, we apply again robust SIFT features matched between successive frames and – in order to prevent drift - additionally matched against confident features of previous frames. Those are dynamically cached within the database, every time the pose is well defined. As feature matching generally produces outliers, the initial set of 2D/3D correspondences is filtered using the well-known RANSAC (Random Sampling Consensus) algorithm. Details can be found in [BPS05].
While this approach is very robust, the computation of the SIFT features, the matching and the outlier rejection lead to high computational costs. A high-quality pose prediction (e.g. from an inertial sensor) allows not only to deal with rapid camera movements with less robust and faster methods, but also enables to transform the feature descriptors with respect to the predicted pose and to center the search area at the expected position. This approach has been implemented for planar patches and provided very stable results. Pre-defined image patches defined as 3D planar surfaces are warped by a homography with help of the predicted pose onto the current image. The patch position is then refined with simple block matching (such as L1- or L2-Norm). As the image textures are never updated, the tracking does not drift at all.

Continuous camera motion: If the camera motions are continuous and the temporal coherence is fulfilled, in particular for stationary systems, small feature windows can be tracked directly. In this case, we use the well-known Kanade-Lucas tracker enhanced by several extensions [ZGN04] like building an image pyramid for coarse-to-fine tracking, estimating affine transformation parameters to detect outliers and prevent drift. As most outliers are already detected at feature level, we use the complete set of 2D/3D correspondences for pose estimation rather than applying a further filtering. This approach is very fast at the expense of robustness against rapid camera movements.

2.4 Tracking with line models

For industrial applications, poorly textured objects with sharp edges have to be handled. Contour-based tracking approaches turned out to be suitable for such scenarios [CMC03].

Camera pose estimation by registering a 3D line model onto the image gradient can be done with or without explicitly extracting the image line features. In the first case, the camera pose computation is achieved by minimizing the distances between the lines of the projected 3D model and the corresponding image line features. A widely used technique to avoid the computationally expensive extraction of image line features is to perform a search for gradient maxima in the image only for several sample points of a line in a perpendicular direction to that line. In our work, we focus on a tracking approach for which the camera pose is estimated by minimizing the distance between sample points on a projected line of the given 3D model and the corresponding maxima of the image gradient.

Multiple hypotheses: The challenge of the tracking system is to find the maximum of the gradient that corresponds to the sample point of a line, as there can be many candidate matches for one sample point (Fig. 3). We apply here the edge tracking method of [VLF04] that uses multiple hypotheses for assigning a 2D feature to a 3D sample point of a line model together with the Tukey Estimator [RL87] as a robust estimator function.

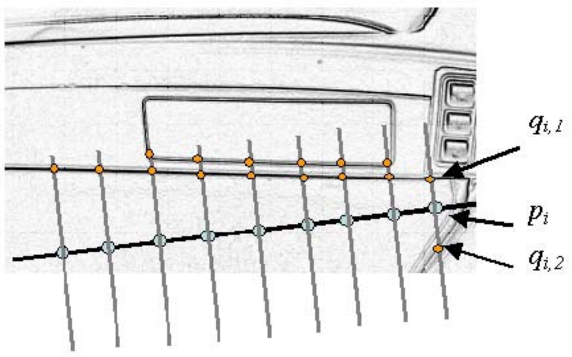


Figure 3: Search segment and multiple hypotheses

The given 3D line model is projected into the image plane with the camera pose of the previous frame. For each projected edge E of the 3D model, equally spaced sample points p_i are determined. At every sample point p_i, a perpendicular search for the points $q_{i,j}$ is performed on both sides of the projected edge. The points $q_{i,j}$ are possible candidates for points on the desired edge.

Adaptive sample points: To improve the robustness of the tracker, we implemented an adaptive system using only gradient maxima that most likely correspond to the considered line sample point. In order to know which gradient maximum is the one on the wanted edge in the image, a state of the visual properties of a sample point on an edge has to be maintained. Therefore, a temporal low pass filter is applied on the visual properties of every sample point of an edge. As the visual perspective, the lighting conditions and the background in an image sequence are not constant over time, the appearance of an object edge can change considerably. Therefore, it is not sufficient to describe the visual state with a single Gaussian distribution. By using a mixture of Gaussians it is possible to describe a single sample point with several visual properties.

Choice and visibility of the sample points: The number of sample points on a projected line is chosen in such a way that the sample points are evenly distributed in the image. The test, if a sample point of a line feature is visible, is performed in hardware using a GL extension. A VRML model of the tracked object is needed for this visibility test.

2.5 Re-Initialization

The goal of the re-initialization is to recover the current pose, when the tracking failed. Here we take advantage of the information collected during the tracking. The current implementation checks the pose quality on base of the re-projection error in the image and the covariance matrices of the camera pose and stores robust features (SIFT). At re-initialization time the features of the current frame are matched with the features cached preliminarily. The problem can then be seen as a wide baseline matching problem intensively investigated in the computer vision community [Bau00, LVT03, Low04].

3 Arranging the building blocks: A generic framework "VisLib2"

Since the objects to be tracked depend on the scenarios, it should be possible to easily (re)-configure the tracking system even at running time. For this purpose, we developed a modern and flexible C++ based library that offers a high level of abstraction. The fundamental concept of our implementation relies on an explicit and systematic separation of data and algorithms. Data are stored in a shared memory region called DataSet that offers access to data objects by name over well-defined keys. Algorithms are encapsulated in Actions with an unified interface. This interface is reduced to four simple functions, one for setting the data keys, one for applying the action to the data and two for reading from and writing to an XML file all internal parameters and keys needed for execution. All available actions are registered in the so-called TypeFactory that allows instantiation (by name) at run-time. The instantiation of actions and data enables completely dynamic system configuration and extension. New actions are developed as additional DLLs and - with the TypeFactory mechanism – they are simply added to the algorithm pool at system start. Actions can be combined to ActionPipes, which perform the sequential execution of all added actions in a loop. As ActionPipes are actions themselves, whole action combinations, e.g. the realization of a tracking system, can be saved and loaded via xml once they have been composed. As a nice analogy, one can say that VisLib2 configurations are read from XML files like C programs are built up from source code. For visual assembling, loading, saving and executing of VisLib2 ActionPipes, a user interface named PicMod has been developed. As the pipes only support sequential execution of actions, the integration of more complex control structures based on complete graphs is under development.

4 Evaluation

We defined several scenarios in the field of industrial maintenance and cultural heritage, each one of them having different boundary conditions and requirements due to varying target objects, scenes, camera motions and lighting changes. For example, we use as non-planar test objects a poorly textured BMW armrest, a machine command and the engine of a BMW. A (well textured) poster serves as a planar test object and a whole room as a larger test scene. From those target scenes, we captured video sequences (320x240) with a Firewire camera performing rapid camera movements as well as continuous ones and switching on and off the lights. As an outdoor scenario, a video of the ruins of Sagalassos has been recorded with a camcorder being fixed to a tripod, thus performing only rotational motions. Then, we defined different instances of the tracking framework (MA-MD) by combining appropriate tracking actions according to the requirements of the scenarios:

Method	Initialization	Tracking	Re-Initialization
MA	Local SIFT features	Local SIFT features	Local SIFT features
MB	Marker	Local KLT features	Marker
MC	Marker	Line model	Marker
MD	Local SIFT features	Texture Patches	Local SIFT features

Table 1: Instances of the tracking framework

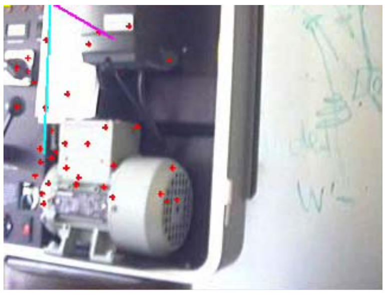
Results of the different methods are given in Fig. 4. All methods have been tested on a 1.7 GHz processor with the accordant parameters (e.g. search ranges, number of features) being optimized for yielding sufficiently stable results. We did not correct for radial distortion. Scenario (a)-(c) shows the machine command with obvious light changes and occlusions. It has been calibrated using method MA (SIFT). The BMW engine (d)-(f) has been processed with the same approach. Images (g)-(i) (engine command, armrest, room) show results from the tracking with line models (MC). The poster (k)-(l) has been calibrated successfully with method MD (texture patches). Again, these images show obvious light changes and large motions. Frames (m)-(o) show the outdoor environment and could be handled with the same approach due to pure rotational camera movements. The analysis below sheds light on the strengths and weaknesses of the different methods. Precision, robustness and speed are chosen as criteria for the validation of their capability with respect to the given scenarios.

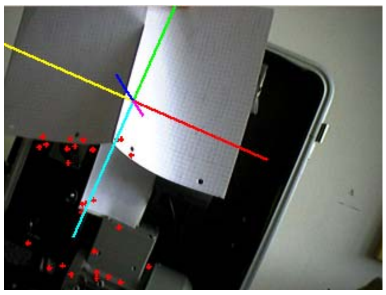
Method MA (SIFT) is very robust against rapid camera movements, light changes (Fig. 4 (b)) and partial occlusions (Fig. 4 (a),(c)). In particular, this method is very stable with highly textured as well as with poorly textured objects, respectively the command machine (Fig. 4 (a)-(c)) and the BMW engine compartment (Fig. 4 (d)-(f)). Poorly textured objects with clear contours are well suited to the line model method. The BMW armrest (Fig. 4 (h)), the room (i) and also the machine command (g) fit this requirement and are well tracked over long sequences.

As the computation of the Euclidian invariant SIFT features is computationally very expensive, method MA (12 Hz) is appropriate for challenging scenarios with rapid camera movements and uncontrolled lighting, whereat the target object is small and exhibits no strong contours. Approach MC reaches a higher speed (20 Hz) and is the most appropriate one to use if we consider objects with dominant contours or larger test scenes.

If the target object is planar and highly textured like the poster, we do not have to spend much time computing invariant features, and reach very good results with method MD (Fig. 4 (j)-(l)). Rapid camera movements are also handled, as our implementation reaches a high framerate (25 Hz) and the texture patches are always transformed by a homography into a convenient view. Even light changes (Fig. 4 (l)) and a large change in depth (Fig. 4 (j),(k)) are managed. As the homography warping is also valid for purely rotational camera motions combined with non-planar target scenes, we obtained good results applying this method to the video of the outdoor environment (Fig. 4 (m)-(o)) by assuming the whole scene lies on a single plane. If the target scene is sufficiently textured and the camera motions are expected to be continuous without many rapid movements, method MB is the most appropriate one, as it is very fast (30 Hz) and delivers a smooth camera trajectory. A continuous image sequence of the engine command could be calibrated very precisely using this approach.

(a)


(b)


(c)

(d)

(e)

(f)

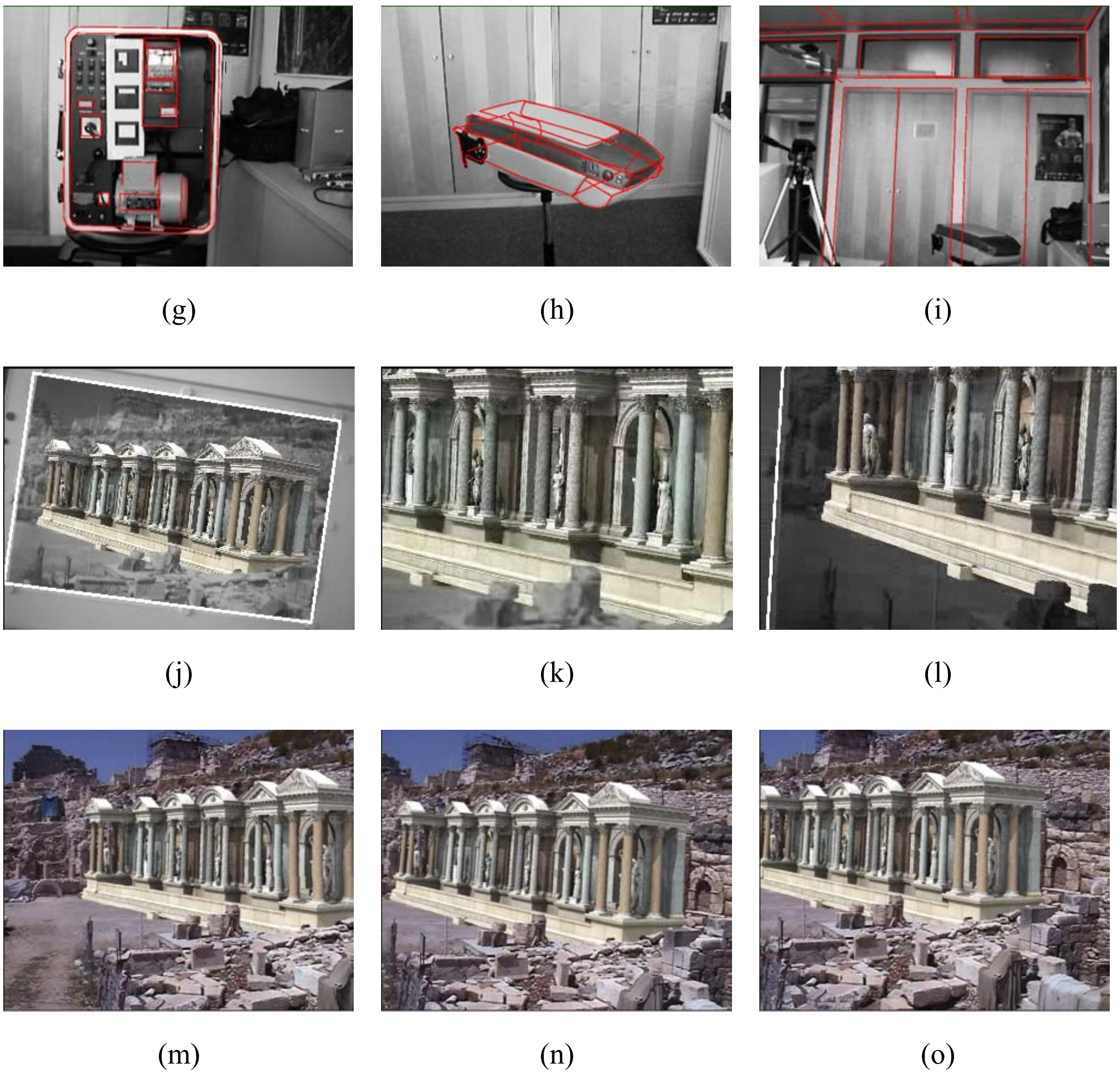


Figure 4: Different scenarios have been calibrated with appropriate instances of our tracking framework: For optical verification of the current pose, the video frames are augmented with either the axis of the world coordinate system or the CAD model of the target object or scene.

5 Conclusion

We have presented an overview of different tracking approaches and stressed their strengths and weaknesses in the context of visual tracking for augmented reality. We showed how to combine them as building blocks of a general framework. In particular, we proposed an improved line model tracking algorithm with multi-hypothesis matching, and introduced adaptive learning of the edge properties. On the one hand, this approach enables a better identification of the most likely edge, and on the other hand, avoids local minima by excluding wrong hypotheses. This method is used for initialization as well as for tracking. The different feature-based methods provides stratifying results for fast and robust tracking. The matching with pre-defined reference features removes drift and thus enables tracking of very long sequences with a high accuracy.

None of the presented combinations leads to the best results in all the tested scenes. Nevertheless we could always find at least one sensible combination of tracking actions for every target application that optimizes precision and speed yielding the necessary robustness.

Future work will concentrate on the management and tracking of dynamic scenes. Furthermore, additional methods for automatic tracking initialization have to be found in order to avoid any kind of user interaction or scene engineering.

6 References

[Bau00] Baumberg, A.: Reliable Feature Matching Across Widely Separated Views. Proc. CVPR, Volume(1), 2000

[BPS05] Bleser, G., Pastamorv, Y., Stricker, D.: Real-time 3d camera tracking for industrial augmented reality applications. Proc. WSCG, pp. 47-53, Plzen, 2005.

[CMC03] Comport, A.I., Marchand, E., and Chaumette, F.: A real-time tracker for markerless augmented reality. Proc. ISMAR, pp. 36-45, Tokyo, Japan, 2003

[GSN02] Genc, Y., Riedel, S., Souvannavong, F., Navab, N.: Markerless tracking for augmented reality: A learning-based approach. Proc. ISMAR, 2002

[HS88] Harris, C., Stephens, M.J.: A combined corner and edge detector. Proc. Fourth Alvey Vision Conference, pp. 147-151, Manchester, 1988

[KB99] Hirokazu K. and Mark B.: Marker tracking and hmd calibration for a video-based augmented reality conferencing system. Proc. IWAR, pp. 85-94, October 1999

[KPV99] Koch, R., Pollefeys, M., Van Gool, L.: Realistic 3D Scene Modeling from Uncalibrated Image Sequences. Invited contribution to special session on Image Analysis and Synthesis, 1999

[KS04] Ke, Y., Sukthankar, R.: PCA-SIFT: A More Distinctive Representation for Local Image Descriptors. Proc. CVPR, 2004

[Low04] Lowe, D.G.: Distinctive Image Features from Scale-Invariant Keypoints. International Journal of Computer Vision, 2004

[LVT03] Lepetit, V., Vacchetti, L., Thalmann, D. and Fua, P.: Fully Automated and Stable Registration for Augmented Reality Applications. Proc. ISMAR, 2003

[MS02] Mikolajczyk, K., Schmid, C.: An affine invariant interest point detector. Proc. ECCV, 2002

[MS03] Mikolajczyk, K., Schmid, C.: A performance evaluation of local descriptors. Proc. CVPR, pp. 257-264, 2003

[MTB04] Matthews, I., Takahiro, I., Baker, S.: The Template Update Problem. IEEE Transactions on Pattern Analysis and Machine Intelligence, pp. 810-815, 2004

[MZS03] Mikolajczyk, K., Zisserman, A., Schmid, C.: Shape recognition with edge based features. Proc. British Machine Vision Conference, 2003

[RL87] Rousseeuw, P., Leroy, A.: Robust Regression and Outlier Detection, Wiley, 1987

[Str01] Stricker, D.: Tracking with Reference Images: A Real-Time and Markerless Tracking Solution for Out-Door Augmented Reality Applications. Proc. of VAST, 2001

[ST94] Shi, J., Tomasi, C.: Good features to track. Proc. CVPR, pp. 593-600, Seattle, Washington, 1994

[SUYT03] Satoh, K., Uchiyama, S., Yamamoto, H., Tamura, H.: Robust Vision-Based Registration Utilizing Bird's-Eye View with User's View. Proc. ISMAR, pp. 46-55, 2003

[VLF04] L. Vacchetti, V. Lepetit, and P. Fua. Combining edge and texture information for real-time accurate 3d camera tracking. Proc. ISMAR, 2004.

[YNA99] You, S. Neumann, U., Azuma, R.: Hybrid Inertial and Vision Tracking for Augmented Reality Registration," Proc. IEEE Virtual Reality*,* IEEE CS Press, pp. 260-267, Los Alamitos, California, 1999

[ZGN04] Zinßer, T., Gräßl, C., Niemann, H.: Efficient Feature Tracking for Long Video Sequences. Proc. DAGM Symposium, 2004