Fully Automated and Stable Registration for Augmented Reality Applications *

Vincent Lepetit † Luca Vacchetti † Daniel Thalmann ‡ Pascal Fua †
† CVlab ‡ VRlab
Swiss Federal Institute of Technology
1015 Lausanne, Switzerland
{vincent.lepetit,luca.vacchetti,daniel.thalmann,pascal.fua}@epfl.ch

Abstract

We present a fully automated approach to camera registration for Augmented Reality systems. It relies on purely passive vision techniques to solve the initialization and real-time tracking problems, given a rough CAD model of parts of the real scene. It does not require a controlled environment, for example placing markers. It handles arbitrarily complex models, occlusions, large camera displacements and drastic aspect changes.

This is made possible by two major contributions: The first one is a fast recognition method that detects the known part of the scene, registers the camera with respect to it, and initializes a real-time tracker, which is the second contribution. Our tracker eliminates drift and jitter by merging the information from preceding frames in a traditional recursive tracking fashion with that of a very limited number of key-frames created off-line. In the rare instances where it fails, for example because of large occlusion, it detects the failure and reinvokes the initialization procedure.

We present experimental results on several different kinds of objects and scenes.

1. Introduction

Using the well known AR toolkit [14] is a good way to obtain good registration data because it is easy to use, automatic and robust. This level of performance, however, requires the use of markers, which is often cumbersome and sometimes even impossible. By contrast, we propose a registration method based on natural features that, given a rough partial CAD model of the scene, achieves a comparable level of automation while being accurate enough to provide the visual comfort required by AR applications.

This has been made possible by our two main contributions. The first one is a fast and automatic initialization method that detects the known part of the scene and registers the camera with respect to it. During a learning stage, a database of feature points in the scene is build. Given an image of the scene, the camera can then be registered online by matching the feature points present in the image against the database. This approach is inspired by recent work on object recognition [15, 1] and wide-baseline matching, but our method goes much further towards reducing the computational burden, thus making it suitable for AR applications.

The second contribution is a real-time camera tracking that can handle large camera displacements, drastic aspect changes and partial occlusions and that is drift- and jitter-free, without imposing any restriction on the target object's complexity. It achieves this level of performance by fusing the information from preceding frames in a conventional recursive tracking fashion with the one provided by a very limited number of key-frames created during an off-line stage. In the rare instances where it fails, for example because of large occlusion, it can detect the failure and reinvokes the initialization procedure. Figure 1 depicts a typical behaviour of our tracker on a simple application. In the Results section, we will show the capability of our system in more complex situations.

We believe our system to go well beyond the current AR state-of-the-art: While automated initialization is a crucial problem for practical applications, it is rarely addressed in the context of natural feature based tracking. Most existing trackers are initialized by hand, or require the camera to be very close to a specified position. Furthermore, when they lose track they typically have to be reinitialized in the same fashion. Similarly, while the off-line camera registration problem can be considered as essentially solved, robust online tracking remains an open issue. Many of the real-time algorithms described in the literature still lack robustness, can lose a partially occluded target object, tend to drift or are prone to jitter, which makes them unsuitable for real-world Augmented Reality applications.

Our system requires a small user-supplied set of key-

*This work was supported in part by the Swiss Federal Office for Education and Science.

Figure 1. Typical behaviour of our system. a: a keyframe of the tracked object, used during the learning stage; b: an image is given to the system, it is matched with one of the keyframes (here the one shown in a) to automatically initialize the tracker; c and d: feature points are tracked to register the camera, even under partial occlusions; e: the tracker fails because of a complete occlusion of the tracked object, the failure is detected and the initialization procedure is reinvoked; f: this procedure recognizes the object when it reappears; g: the tracker can handle aspect changes; h, i and j: virtual objects can be added with respect to the tracked object, and occlusion are properly handled. By watching the video at http://cvlab.epfl.ch/research/augm/, one can see that the virtual objects are remarkably stable.

frames and a rough CAD model of part of the real scene for both initialization and tracking. In theory, this can be seen as a drawback but it is not really one in practice: A CAD model is also required by most AR applications that can benefit from a 3D tracker and can easily be created using either automated techniques or commercially available products such as Image Modeler by RealViz$^{(TM)}$, or Boujou by 2D3$^{(TM)}$. These commercial products can also be used to very quickly create a few key-frames. The keyframes make the tracker robust, and allow automatic initialization.

Unlike previous techniques that limit the range of object shapes that can be handled, we impose no such constraint and put no restriction on the object type or its complexity as long as enough feature points are present. We have been able to track objects as different as a machine tool, a corridor and a human face.

In the remainder of the paper, we first discuss related work. Section 3 explicits some important details about keyframes used in our system. Section 4 describes our initialization method, and Section 5 describes the tracking stage.

2. Related Work

2.1 Automatic Initialization

Few works address specifically the problem of tracker initialization in the Augmented Reality area, but this problem is closely related to object recognition. In this research domain, two approaches can be distinguished, either a view-based representation or a 3D representation is used.

View based methods represent objects by a set of images [18, 19] or image properties such as histograms [24], and have the advantage of allowing fast recognition, even for complex objects. Nevertheless, it can require huge amounts of memory, and it is difficult to handle occlusions and cluttered backgrounds. Considering the object 3D model allows more flexibility. In this case, the object recognition and the camera registration are simultaneously performed: 2D geometric features such as corners or edges are extracted from the image and matched against the object 3D features, and geometric constraints are applied to remove spurious correspondences. These methods are restricted to relatively simple objects, and can be relatively expensive [13, 5] in terms of computation time.

Recently, methods that combine both approaches have been developed. The object to be recognized is represented by a set of key points characterized by their local appearances. The object 3D model may be known [1], or not [15]. Feature points are extracted from the images, and characterized in order to be matched against the set of keypoints. Then, the spurious matches are removed by applying geometric constraints, for example by robustly estimating the epipolar geometry between the different object views or the object pose when the 3D model is available.

Ideally, the feature point extraction and characterization should be insensitive to viewpoint and illumination changes. Scale-invariant feature extraction can be obtained

using the Harris detector [10] at several Gaussian derivative scales, or considering local optima of pyramidal difference-of-Gaussian filters in scale-space [15]. Mikolajczyck et al. [16] have also defined an affine invariant point detector to handle larger viewpoint changes, but it relies on an iterative estimation that would be too slow in our context. Various local descriptors have been proposed: Schmidt et al. [21] use a vector made of orientation-invariant measures that are functions of relatively high order image derivatives. Baumberg [2] uses a variant of the Fourier-Mellin transformation to achieve rotation invariance. He also gives an algorithm to remove stretch and skew and obtain an affine invariant characterization. Allezard et al. [1] represent the key point neighbourhood by a hierarchical sampling, and rotation invariance is obtained by starting the circular sampling with respect to the gradient direction.

These different works obtain impressive results, but they are still too slow to be used in an Augmented Reality context, because the involved computation typically takes several seconds. Our method (Section 4) goes much further towards reducing the computational burden to about one hundred milliseconds, thus making it suitable for AR applications.

2.2 Real-Time Camera Tracking

Some model-based approaches to camera tracking look for 3D poses that correctly re-project the features such as edges, line segments, or points, of either a planar or fully 3D model into a 2D image. They rely on finding the local minimum in an objective function such as edge- or segment- [8] or point- [9] based methods. Therefore, the optimization procedure may get trapped in a wrong local minimum, in particular in the presence of aspect changes or even when two edges of the same object become very close to each other. As a result, the tracker's behavior can become unpredictable.

Other approaches [23] derive the camera position by concatenating transformations between adjacent frames, by tracking natural features. The tracking is accurate and there is no jitter because feature matching is done with respect to very close frames. Unfortunately, for long sequences these methods suffer from the error accumulation problem.

A number of methods introduce reference frames [4] and some keep track of disappearing and appearing points [20]. However, they need to smooth the results, for example by means of Kalman filtering, to prevent jittering. As pointed by [20] such filtering is not really suitable for Augmented Reality applications: for example, in the case of a Head Mounted Display application, it is not realistic to have a simple model for the head motion.

Traditional frame-to-frame recursive approaches to matching and those that rely on reference frames both have their strengths and weaknesses. Keyframe-based techniques prevent drift, but cannot provide good precision for every frame without using a very large set of keyframes. Furthermore, they typically introduce jitter. Techniques based on chained transformations eliminate jitter but tend to drift on long sequences and are subject to losing track altogether. In Section 5 we propose a method that merges the information from preceding frames with that provided by key-frames, and that has the advantages of both approaches. To compare these different approaches, we conducted the following experiment. We used our feature matching approach to track the projector in the sequence of Figure 11 three different times:

1. using only off-line keyframes,
2. using only chained transformations,
3. combining both using our proposed method.

Figure 2 depicts the evolution of one of the camera center coordinates with respect to the frame index and we have verified that the behavior for all other camera parameters is similar. In all three graphs, we superpose the output of the tracker using one of the three methods mentioned above with "ground truth" obtained by manually calibrating the camera every 5 frames. The sequence made by using keyframes only exhibits jitter while the recursive one is quickly corrupted by error accumulation. The method presented in this paper keeps closer to the ground truth and avoids drift.

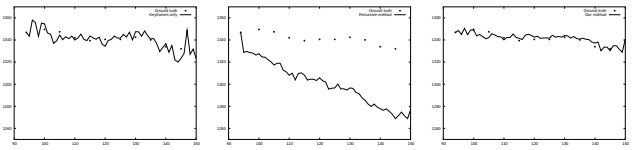

Figure 2. Plots showing one coordinate of the camera center tracked using three different methods. The dots represent the ground truth. First plot shows the jitter of keyframe method. Second plot highlights the error accumulation of an early version of our tracker, based on recursive estimation. Last plot shows the result of our method.

3. Keyframes

In this section we specify what exactly a keyframe is in our system. We also describe the re-rendering procedure that is used in our automatic initialization and tracking methods.

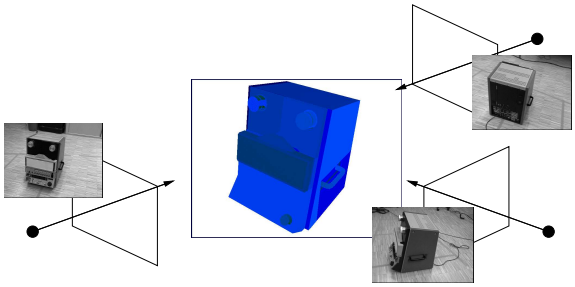

Figure 3. In an off-line procedure, keyframes of the object or the scene are shot.

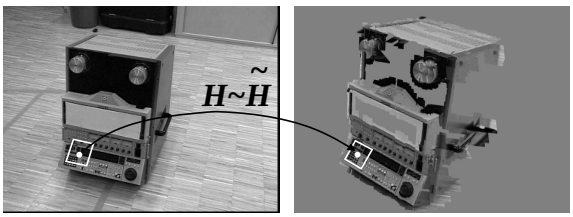

Figure 4. Pixels around interest points are transfered from the keyframe (left) to the re-rendered image (right) using a homography, which can be locally approximated by an affine transformation.

3.1 Creating Keyframes

During the off-line stage, a small set of images, that we call keyframes, representing the scene or the object from different viewpoints, has to be chosen and calibrated. For the results presented in this paper, about ten snapshots taken from all around the objects are well enough (see Figure 3). The calibration can be automatic using commercial post-production tools, such as the ones by RealViz or 2D3. For the sequences presented in this paper, we developed a simple application in which the camera is calibrated by choosing very few points on the 3D model and matching them with the corresponding 2D points in the image. In this way creating a keyframe set is just a matter of some minutes.

When the projection matrix is known for every keyframe, the system performs interest point detection using the Harris corner detector [**?**] and back-projects the points that lie on the object surface. Finally, a keyframe collects the following data: The two sets of corresponding 2D and 3D points, the bitmap image of the 2D point neighborhood and the corresponding surface normals $\vec{n}$, used for keyframes rerendering (see below), and the camera projection matrix P_K, decomposed into the internal parameters matrix A_K, the rotation matrix R_K and the translation vector T_K, so that $P_K = A_K[R_K|T_K]$.

3.2 Rerendering a Keyframe

Both our initialization and tracking methods use synthesized images of the modeled part of the scene, as seen from a new viewpoint. They are used during the learning stage by our initialization method and by the tracker to match distant frames in a fast way to effectively use the keyframes.

We synthesize the new image, that we call the "re-rendered" image, by skewing pixel patches around each interest point from the keyframe image to the given viewpoint. An alternative solution would have been to re-render an image of the object using an OpenGL textured 3D object, but we chose the first way to have a more precise result around the points and to speed up the system.

Locally approximating the object surface around the interest points by a plane, each patch in the keyframe is related to the corresponding image points in the re-rendered image by a homography (Figure 4). Given the corresponding 3D plane π of a patch around interest point m_0 having coordinates $\pi = (\vec{n}, d)$ so that for points on the plane $\vec{n}^\top X + d = 0$, the expression for the homography H induced by the plane can be easily deduced from [11]. If the new viewpoint is defined by the projection matrix $P = A[R|T]$, the expression of H is:

$$H = A_K(\delta R - \delta T.\vec{n'}^\top/d')A^{-1} \quad (1)$$

with

$$\delta R = RR_K^\top;\ \delta T = -RR_K^\top T_K + T;$$
$$\vec{n'} = R_K\vec{n};\ d' = d - T_K^\top(R_K\vec{n}).$$

The matrix H transfers the pixels m around m_0 to the pixels m' in the rerendered image so that: $m' = H.m$. To save computation time, this transformation can be approximated around m_0 by an affine transformation, obtained by the first order approximation

$$m' \simeq H.m_0 + J_H(m_0).(m - m_0)$$

where J_H is the Jacobian of the function induced by H.

In our system, the 2D mapping is done using optimized functions provided by the IPP library of Intel.

4. Automatic Initialization Method

Following recent approaches in object recognition, our automatic initialization method relies on a learning stage, where a database of key feature points is constructed. Each key point consists of a 3D point corresponding to an interest point that lies on the object model in the keyframes, and a descriptor based on its local appearance in the images.

The recognition phase then consists in robustly matching feature points in the image with the points present in the database. The matching process is performed according to a similarity measure between their descriptors. To make this process efficient, the point local appearance must be characterized in a way that is invariant to viewpoint changes, and that also allows fast comparisons between points. To this end, we propose to use an eigen image approach. In the remainder of this Section, we first give a brief presentation of eigen images, and show how they can be used to define a viewpoint invariant description of points. Finally, we discuss the matching strategy and give some results.

4.1. Viewpoint Invariant Local Description

As we discussed in the Related Work Section, different descriptors have been defined in the literature, but they are too time consuming. We use an idea similar to the one developed in [15] to make the point local description robust to viewpoint changes, where the local descriptor consists of a set of image patches of the point seen from different viewpoints. Here this idea is used in conjonction with eigen images to achieve low computation time. We also make use of the object model, which is not available in [15].

During the off-line learning stage, one database of key points per keyframe is build. We will discuss how to use these several databases below. Each keyframe is re-rendered (see Section 3.2) from different viewpoints around the camera position computed for the keyframe, slightly changing the camera orientation and translation. We associate to the 3D points M_i present in the keyframe the image patches (denoted $V_{i,j}$ onwards) around their projections in the re-rendered keyframes as shown in Figure 5.

To perform fast template matching, we make use of the eigen image methodology. This approach has been shown to be remarkably efficient for robot positioning [18], face [17], hand [3] or navigation landmark recognition [6]. Given a set S of p images, eigen images approaches construct a family of basis images that characterize the majority of the variation in S by a principal component analysis (PCA). Then only few multiplications are required to compare two patches instead of the large number involved by the classical evaluation of the correlation. By combining this comparison with a hash table, this allows us to perform a large number of point comparisons in a short time. We normalize the brightness of the images to make the correlation measure insensitive to illumination changes [6].

The eigen space of all the patches $V_{i,j}$ is computed, and finally, the key point is defined as a 3D point and the set of its patches expressed in the eigen space:

$$K_i = (M_i, V'_{i,1}, \ldots, V'_{i,l}),$$

where l is the number of re-rendered keyframes, and the

Figure 5. Some patches associated to a key point.

$V'_{i,j}$ the expression of the patches $V_{i,j}$ in the eigen space. The set of patches is a way to represent the possible local appearances of the 3D point seen from viewpoints around the keyframe viewpoint. In practice we use patches of size 16×16 pixels, and 8 re-rendered keyframes.

4.2. Matching Strategy

When our initialization system is given an image, it detects the interest points m_k present in this image, and computes the expressions W'_k in the eigen space of the patches centered in these points. Then each point m_k is matched with the key points K_i that verify

$$\exists\, j \text{ so that } \|W'_k - V'_{i,j}\|^2 < T^2,$$

The search for correspondents is performed efficiently since we use eigen images to represent the patches. Each point m_k may be matched with 0, 1 or several key points. Then we can register the camera position from the correspondences between the 2D points m_k and the 3D object points M_i, using the POSIT algorithm [7] and the robust estimator RANSAC [11]. POSIT computes the camera position and orientation from 2D/3D correspondences without needing any initialization, and RANSAC handles the spurious correspondences.

4.3. Results

Figure 6 shows some examples of our automatic initialization method. The recovered viewpoints are accurate enough to initialize our tracker even when the camera position is relatively far to the one corresponding to the tracker: either the object orientation and distance can be different. This procedure takes about 150 ms, detailed computation times are given Figure 7.

Two ways to use this procedure are possible: the image from the camera is matched against one keyframe only, and the user has to move the camera to the neighbourhood of the camera position related to the chosen keyframe, in a fairly large interval as shown in Figures 1 and 6. The second possibility consists in matching the camera image against all the keyframes and choosing the one that gives the more matches. This solution allows the user to start from any position, but it takes more time.

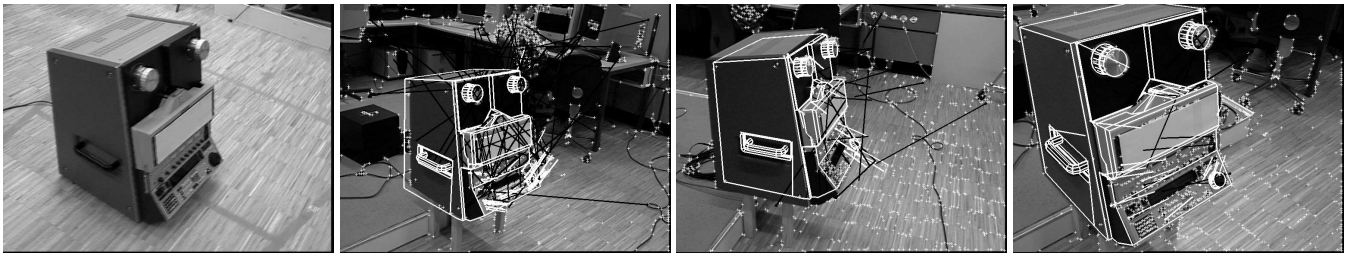

Figure 6. Three examples of automatic initialization. The used keyframe is the first image on the left. Note that the camera positions are quite different. The white lines represent the correct matching, the black ones the spurious matches removed by the robust viewpoint estimation.

Interest point extraction	20 ms
Interest point characterization	10 ms
Matching	90 ms
Robust viewpoint estimation	10 ms

Figure 7. Computation times (on a Pentium IV, 2.3GHz) of the initialization procedure, for 500 interest points detected in the image, matched against a database of about 1000 key points with 8 patches each. The eigen space computation is performed off-line and takes a few seconds per keyframe.

5. Tracking Method

Our tracking method is suitable for any kind of 3D textured object that can be described as a polygonal mesh. It starts with the 2D matching of interest points, and then it exploits them to infer the 3D position of the points on the object surface. Once the 3D points on the object are tracked, it is possible to retrieve the camera displacement in the object coordinate system using robust estimation. In the remainder of this section we will describe in detail our tracking algorithm. First we present a simpler version that relies only on the keyframes. This method works well even on long sequences but suffers from jittering. We show how to prevent this problem by adding information from previous frames in the sequence. This method is more complex but allows a remarkably stable insertion of virtual objects.

5.1. Keyframe Based Tracking

At every time step t, we detect Harris interest points (denoted m_t^i onwards) in the current source image. One of the keyframes is chosen in order to maximize the number of common interest points, and the points m_t^i are matched with the points of this keyframe, considering an appearance based measure to establish correspondence. The keyframe choice and our algorithm to perform the wide baseline matching between the current frame and the keyframe are described below. Note that we consider only one keyframe at a time: using more keyframes would certainly yield a more accurate viewpoint estimation, but also more computation time.

The 3D position of each point in the keyframe have been precomputed, and expressed in the world coordinate system. They are propagated to the matched points in the frame at time t, and from these 2D-3D correspondences, the camera rotation R_t and translation T_t for this frame can then be estimated using a numerical minimization with a M-estimator [12] initialized with the position recovered in the previous frame as initial guess. More formally, we minimize the residual sum:

$$r_t = \sum_i \rho_{\text{TUK}} \left(\phi_t(M_K^{\mu(i)}, m_t^i) \right)$$

over R_t and T_t, where

- $\phi_t(M, m)$ is the reprojection error in the frame acquired at time t (with respect to R_t and T_t) of 3D point M matched with 2D point m;
- the interest point m_t^i is matched with the 3D point $M_K^{\mu(i)}$ of the keyframe chosen at time t;
- ρ_{TUK} is the Tukey M-estimator used for reducing the influence of wrong matches.

As we previously discussed, a pose estimation based exclusively on keyframes would have poor accuracy when few points are matched or when they are not regularly spread out in the scene and in the image. In Section 5.2, we show how to merge the two sources of information.

Keyframe choice At every time step, the keyframe to be matched with the current frame must be chosen in order to

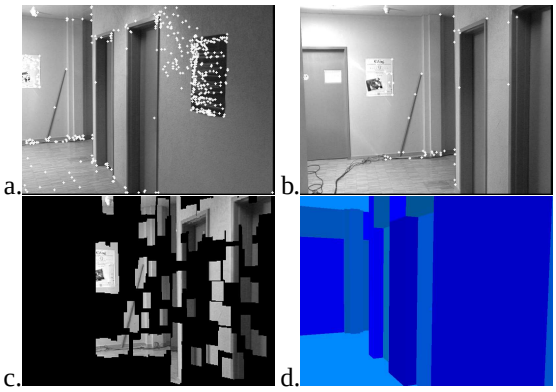

Figure 8. a. A keyframe; b. the current frame, c. the re-rendered keyframe with respect to the previous camera position estimate; d. an example of facets index rendering used for keyframe choice.

maximize the number of common interest points. We use the following criterion:

$$\sum_{f \in Model} \left(\text{Area}\left(A[R_{t-1}|T_{t-1}], f\right) - \text{Area}\left(A_K[R_K|T_K]\right)\right)^2,$$

where $\text{Area}(P, f)$ is the 2D area of the facet f after projection by P, and $A_K[R_K|T_K]$ and $A[R_{t-1}|T_{t-1}]$ the projection matrices of the keyframe and the previous frame. The facets area in the keyframes (the terms $\text{Area}\left(A_K[R_K|T_K]\right)$) can obviously be pre-computed. Areas in the previous frame are efficiently computed using an OpenGL rendering of the object model where every facet is rendered in a different color, representing the facet index. The area of every single facet is estimated by summing the number of occurrences of the facet's pixels. This method has constant complexity, and requires only a single reading of the image.

Wide baseline matching while tracking Conventional methods to match points between images make use of a square bi-dimensional correlation window. This technique gives results under the assumption of very small perspective distortion between two frames. However, to effectively use keyframes, the ability to match distant frames in a fast way becomes essential. Consequently we rerendered the chosen keyframe from the viewpoint estimated at time $t-1$. The re-rendered image is in a more convenient position as shown in Figure 8 and can be matched with the current frame using a simple, conventional method. This method allows us to effectively match views even where there are as much as 30 degrees of rotation.

5.2. Merging Information from Keyframes and Previous Frames Using Local Adjustment

One of the key issues of our method is to effectively combine keyframes and previous frame information. In [25], we gave a simple method that gives good results, but that tends to accumulate error when too few points from the keyframes are available. In this paper, we give a solution based on a local adjustment, that makes the inserted virtual objects more stable, and compensates relatively bad initialization, such as the ones provided by our automatic initialization method.

2D points lying on the projected 3D model in the previous frame are matched with points in the current frame. These points are the projection of 3D points lying on the 3D model. To coherently merge the information from the keyframe and the previous frame, we simultaneously optimize the reprojection errors in these frames over the 3D position of these points, and over the viewpoints related to the previous and the current frames. The problem becomes:

$$\min_{\substack{R_t, T_t \\ R_{t-1}, T_{t-1} \\ M_i}} \quad r_t + r_{t-1} + \sum_i s_t^i \tag{2}$$

with

$$s_t^i = \rho_{\text{TUK}}\left(\phi_t(M_i, m_t^i) + \phi_{t-1}(M_i, m_{t-1}^{\nu(i)})\right),$$

where the interest point m_t^i detected in the current frame is matched with the point $m_{t-1}^{\nu(i)}$ detected in the previous frame.

The important point here is that the 3D position M_i of the tracked points are also optimized, but constrained to stay on the 3D model surface. The formulation of this objective function allows us to satisfy both the constraints from the matches with keyframes and between the successive frames, without assumption of the accuracy of viewpoints previously estimated. Equation (2) can be rewritten as:

$$\min_{R_t, T_t, R_{t-1}, T_{t-1}} \left(r_t + r_{t-1} + \min_{M_i} \sum_i s_t^i\right) \tag{3}$$

since r_t and r_{t-1} are independent of the tracked points M_i.

We will follow the reasoning of [22], who have a similar term in off-line Model Based Bundle Adjustment, to eliminate the M_i in order to simplify the minimization problem: instead of estimating the M_i, it is equivalent to estimating its projection in the two images. Then, according to [22], the terms s_t^i can be approximated using a transfer function that involves only the point reprojection. Such a transfer function $\Psi(m_1, P_1, P_2)$ returns the point m_2 so that it exists a 3D point M belonging to the model surface $m_1 = P_1 M$

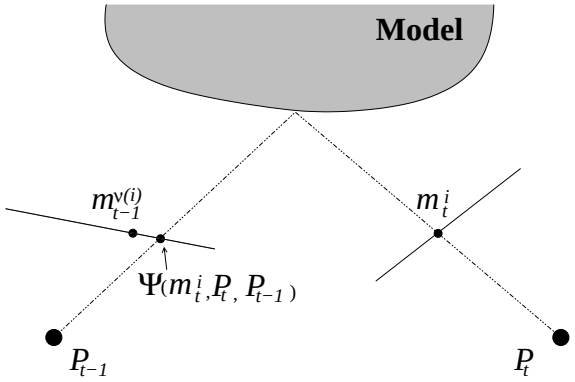

Figure 9. Computing s_t^i: the camera positions P_{t-1} and P_t are simultaneously optimized on-line.

and $m_2 = P_2 M$. Finally s_t^i is approximated by (see Figure 9):

$$s_t^i = \rho_{\text{TUK}} \left(\left\| \Psi(m_{t-1}^{\nu(i)}, P_{t-1}, P_t) - m_t^i \right\|^2 + \left\| \Psi(m_t^i, P_t, P_{t-1}) - m_{t-1}^{\nu(i)} \right\|^2 \right)$$

where $P_{t-1} = A[R_{t-1}|T_{t-1}]$ and $P_t = A[R_t|T_t]$.

Efficient computation In practice we use the viewpoint recovered using the keyframe only as an initial guess. The computation of the transfer function Ψ can be theoretically prohibitive, but since estimated 3D points are then close to their actual position, we reasonably know the facet on which the point actually lies, and Ψ can be approximated by the homography induced by the facet. The robust estimator handles errors on facet assignments and false matches. Since we start from a good initial estimate, optimization is very quick and converges in a couple of iterations.

5.3. Failure Detection

When the number of points matched with the keyframe is below than a threshold (lets say 10 points), the tracker is stopped and the initialization procedure is reinvoked, trying to match the new frames with the keyframe the tracker was using. The tracking is restarted when enough points are matched by the initialization procedure.

5.4. Results

After an important work of optimization our tracker nearly runs at real-time (about 15 frames/sec) for images of size 768 $\times$ 576 pixels (computation time details are given Figure 10), and we have a working prototype system running at full real-time (25 frames/sec) on 320 $\times$ 200 images from a Firewire camera. We set up a variety of demonstrations, to show that this method can be used for many different categories of objects.

Interest point extraction	20 ms
Keyframe re-rendering	15 ms
Point matching	20 ms
Viewpoint estimation	10 ms

Figure 10. Computation times (on a Pentium IV, 2.3GHz) of our tracker, for 500 extracted interest points, using images of 768 $\times$ 576 pixels.

Aspect changes In the example depicted by Figure 11, the camera is moving around an old video projector doing a complex movement with 180 degree rotations. The model was created by a designer using Maya, and it took 4 hours of work. The use of keyframes allows our tracker to deal with aspect changes during the sequence.

Occlusions Figure 12 shows snapshots of a second video sequence, where the same object is partially occluded by a human operator, in order to roughly simulate the behaviour of an Head Mounted Display. Since our algorithm considers local features, the tracking is not corrupted by partial occlusions as far as enough feature points are visible.

Generality Our algorithm is able to track objects of very different types. In the third example depicted by Figure 13, we track a human head of which the model has been reconstructed off-line from another short video sequence. Even though we only have the face model and not the whole head, we have been able to track 180 degree rotations. We ran our tracker on this sequence giving only one keyframe.

Scene tracking Figure 14 depicts an example where the camera is moved at the intersection of two corridors. Despite the huge aspect changes throughout the sequence, the camera trajectory is correctly recovered using the 3D model of the corridors and only four keyframes.

Robustness to 3D model errors We experimentally noticed that the 3D model accuracy is not an important issue. For example, there is a small mistake in the 3D model of the object of Figures 11 and 12: The position of one of the two cylinders on the front face is not very accurate, however it does not corrupt the result. Likewise, the same face model of Figure 13 has been used to track other people faces (of various shapes) with success.

These videos and others can be seen at http://cvlab.epfl.ch/research/augm/

6. Conclusion

In this paper we presented an Augmented Reality system that consists of a robust, drift- and jitter-free tracker, which is initialized by an automatic procedure. Our system relies on purely passive vision techniques, and can be used for a large class of objects, with no constraints on the kind of camera motion. We use model information and off-line defined keyframes to keep on following the target object even when it is occluded or only partially visible, or when the camera turns around it. We combine off-line and on-line information to prevent the typical jitter and drift problems.

We believe our system to go well beyond the current AR state-of-the-art: its level of automation, its robustness and its visual comfort due to the stability of the inserted visual objects make it suitable for practical Augmented Reality applications.

References

[1] N. Allezard, M. Dhome, and F. Jurie. Recognition of 3d textured objects by mixing view-based and model-based representations. In *International Conference on Pattern Recognition*, pages 960–963, Barcelona, Spain, Sep 2000.

[2] A. Baumberg. Reliable feature matching across widely separated views. In *Conference on Computer Vision and Pattern Recognition*, pages 774–781, 2000.

[3] M. J. Black and A. D. Jepson. Eigentracking: Robust matching and tracking of articulated objects using a view-based representation. In *European Conference on Computer Vision*, pages 329–342, 1996.

[4] K. Chia, A. Cheok, and S. Prince. Online 6 DOF Augmented Reality Registration from Natural Features. In *International Symposium on Mixed and Augmented Reality*, 2002.

[5] P. David, D. DeMenthon, R. Duraiswami, and H. Samet. SoftPOSIT: Simultaneous Pose and Correspondence Determination", ECCV 02, Copenhagen, May 2002. In *European Conference on Computer Vision*, Copenhagen, Denmark, May 2002.

[6] V. C. de Verdiere and J. Crowley. Local appearance space for recognition of navigation landmarks. *Journal of Robotics and Autonomous Systems*, 31:61–70, 2000.

[7] D. DeMenthon and L. S. Davis. Model-based object pose in 25 lines of code. In *European Conference on Computer Vision*, pages 335–343, 1992.

[8] T. Drummond and R. Cipolla. Real-Time Tracking of Complex Structures with On-Line Camera Calibration. In *British Machine Vision Conference*, September 1999.

[9] Y. Genc, S. Riedel, F. Souvannavong, and N. Navab. Markerless tracking for augmented reality: A learning-based approach. In *International Symposium on Mixed and Augmented Reality*, 2002.

[10] C. Harris and M. Stephens. A combined corner and edge detector. In *Fourth Alvey Vision Conference, Manchester*, 1988.

[11] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2000.

[12] P. Huber. *Robust Statistics*. Wiley, New York, 1981.

[13] F. Jurie. Solution of the Simultaneous Pose and Correspondence Problem Using Gaussian Error Model. *Computer Vision and Image Understanding*, 73(3):357–373, 1999.

[14] H. Kato and M. Billinghurst. Marker Tracking and HMD Calibration for a Video-Based Augmented Reality Conferencing System. In *IEEE and ACM International Workshop on Augmented Reality*, October 1999.

[15] D. G. Lowe. Object recognition from local scale-invariant features. In *International Conference on Computer Vision*, pages 1150–1157, 1999.

[16] K. Mikolajczyk and C. Schmid. An affi ne invariant interest point detector. In *European Conference on Computer Vision*, pages 128–142. Springer, 2002. Copenhagen.

[17] M.Turk and A.Pentland. Eigenfaces for recognition. *Journal of Cognitive Neuroscience*, 3(1):71–86, 1991.

[18] S. Nayar, H. Murase, and S. Nene. Learning, positioning, and tracking visual appearance. In *IEEE International Conference on Robotics and Automation*, pages 3237–3244, San Diego, CA, May 1994.

[19] S. K. Nayar, S. A. Nene, and H. Murase. Real-Time 100 Object Recognition System. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(12):1186–1198, 1996.

[20] S. Ravela, B. Draper, J. Lim, and R. Weiss. Adaptive tracking and model registration across distinct aspects. In *International Conference on Intelligent Robots and Systems*, pages 174–180, 1995.

[21] C. Schmid and R. Mohr. Local grayvalue invariants for image retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(5):530–534, May 1997.

[22] Y. Shan, Z. Liu, and Z. Zhang. Model-Based Bundle Adjustment with Application to Face Modeling. In *International Conference on Computer Vision*, Vancouver, Canada, July 2001.

[23] G. Simon and M.-O. Berger. Real time registration of known or recovered multi-planar structures: application to ar. In *British Machine Vision Conference*, Cardiff, UK, 2002.

[24] M. Swain and D. Ballard. Color indexing. *International Journal of Computer Vision*, 7(1):11–32, 1991.

[25] L. Vacchetti, V. Lepetit, and P. Fua. Fusing Online and Offline Information for Stable 3–D Tracking in Real-Time. In *Conference on Computer Vision and Pattern Recognition*, Madison, WI, June 2003.

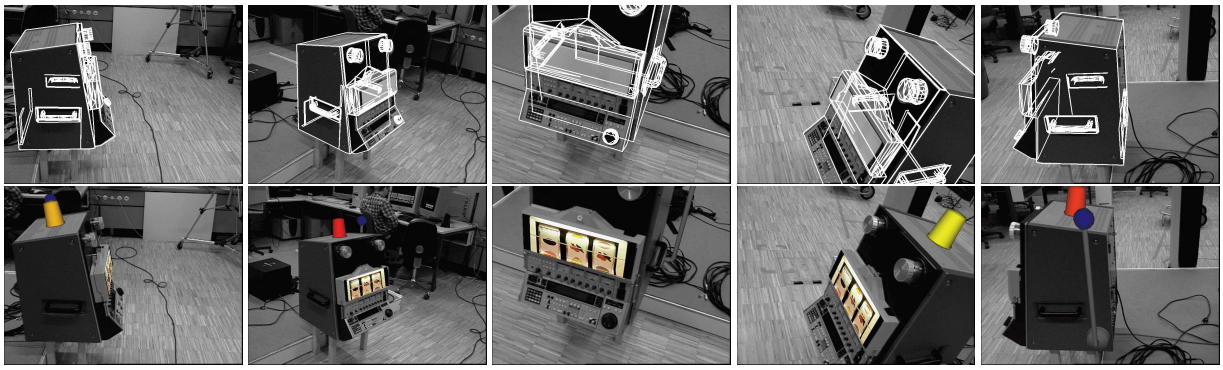

Figure 11. A lever, slot-machine wheels and a jackpot light to the old projector, thus turning it into a slot-machine.

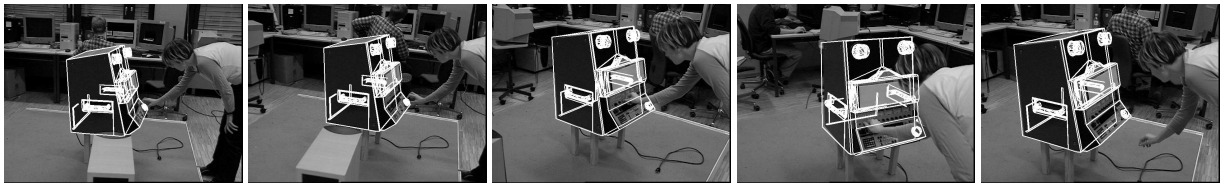

Figure 12. Video sequence with occlusions.

Figure 13. A 3D model of the face is tracked to augment the video sequence by adding glasses and a mustache to the subject.

Figure 14. Corridor tracking results. Only four keyframes have been used.