Fusing Online and Offline Information for Stable 3D Tracking in Real-Time*

Luca Vacchetti Vincent Lepetit Pascal Fua
Computer Vision Laboratory
Swiss Federal Institute of Technology (EPFL)
1015 Lausanne, Switzerland
Email: {Luca.Vacchetti, Vincent.Lepetit, Pascal.Fua}@epfl.ch

Abstract

We propose an efficient online real-time solution for single-camera 3–D tracking of rigid objects that can handle large camera displacements, drastic aspect changes, and partial occlusions. While the offline camera registration problem can be considered as essentially solved, robust online tracking remains an open issue because many real-time algorithms described in the literature still lack robustness and are prone to drift and jitter.

To solve these problems, we have developed a robust approach to 3–D feature matching that can handle wide-baseline matching: our method merges the information from preceding frames in traditional recursive tracking fashion with that provided by a very limited number of keyframes created during an offline stage. This combination results in a system that does not suffer from the above difficulties and can deal with drastic aspect changes. We use Augmented Reality applications to demonstrate its behavior because they are particularly demanding in terms of tracking performance.

1. Introduction

In this paper we propose an efficient online real-time solution for single-camera 3–D tracking that can handle large camera displacements, extreme aspect changes and partial occlusions. While the offline 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, tend to drift, can lose a partially occluded target object, and are prone to jitter that makes them unsuitable for applications such as Augmented Reality. To compute the motion in a given frame, we use a robust approach to 3–D feature matching that can handle wide-baseline matching. Our method merges the information from preceding frames in traditional recursive tracking fashion with that provided by a very limited number of keyframes. This combination results in a system that does not suffer from any the above difficulties and can deal with complex aspect changes such as those shown in Figure 1: We believe this result to be beyond the current state-of-the-art.

Traditional frame-to-frame recursive approaches to matching and those that rely on keyframes 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 and are subject to losing track altogether. To combine the strengths of these approaches, we have therefore developed a robust 3–D feature-matching technique that uses both preceding frames and keyframes that may have been seen from relatively different viewpoints.

Our tracker starts with a small user-supplied set of keyframes. The system then chooses the most appropriate one using an aspect-based method and, if necessary, can automatically introduce new keyframes as it runs. It relies on a 3–D model of the target object or objects, which, in practice, is not an issue since such models are also necessary for many of the actual applications that require 3–D tracking. Furthermore, they can be created using either automated techniques or commercially available products. 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's complexity.

We use Augmented Reality applications such as the one depicted by Figure 1 to highlight the quality of our results because they are particularly demanding in terms of tracking performance. In the remainder of the paper, we first discuss related 3–D tracking work. We then introduce our approach to 3–D feature tracking and to using keyframes. Finally, we present our detailed results.

2. Related Work

While the real-time tracking is not yet a solved problem, our understanding for offline camera registration from an image sequence [1, 2, 3] has progressed to the point where com-

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

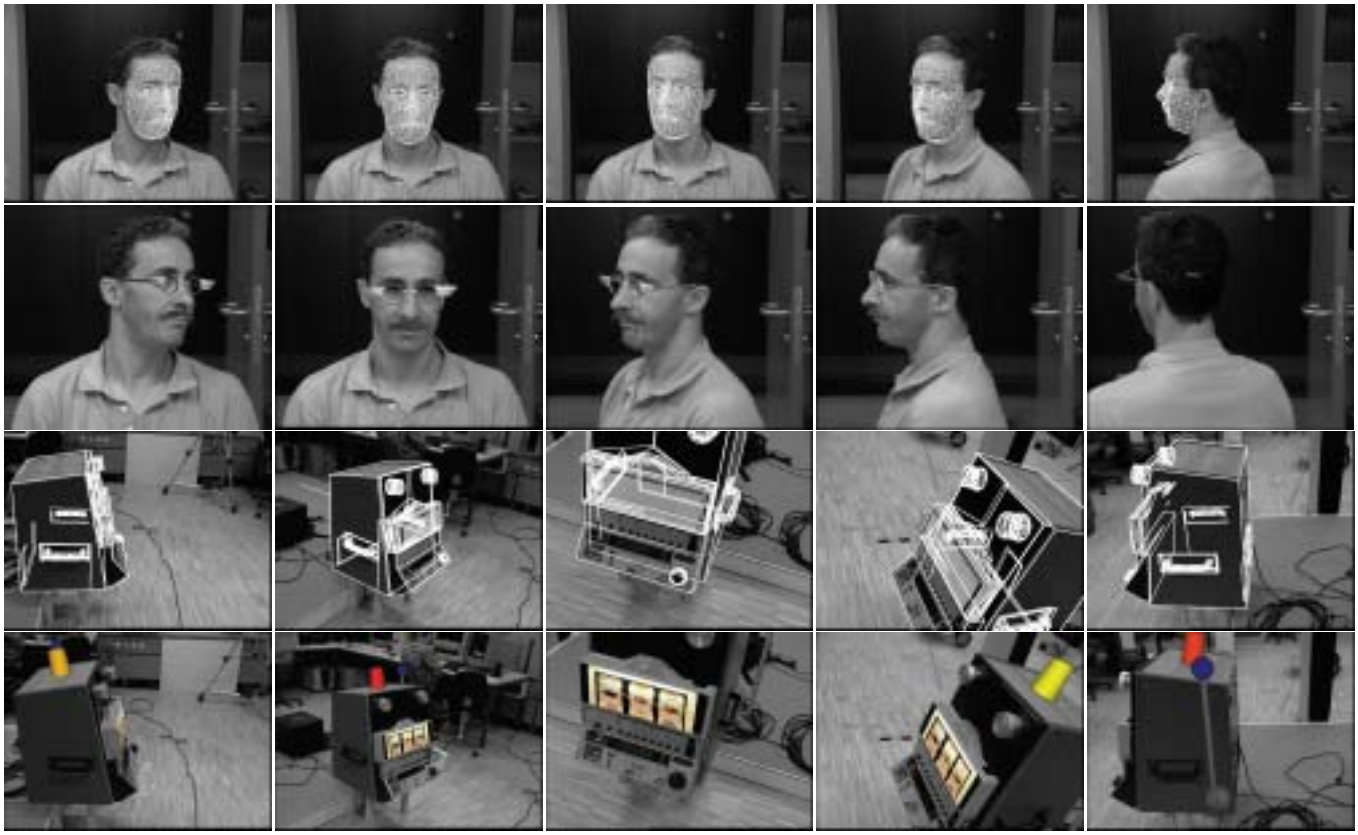

Figure 1: Tracking for augmented reality purposes. First and third rows: Video sequences with overlaid 3–D models whose pose has been computed online using our method. Second and fourth rows: The 3–D models have been used to augment the video sequences by adding glasses and a moustache to the subject and by adding a lever, slot-machine wheels and a jackpot light to the old projector, thus turning it into a slot-machine.

mercial solutions are now available. By matching natural features such as interest points between images these algorithms obtain high accuracy even without *a priori* knowledge. For example in [2] the authors consider the image sequence hierarchically to derive robust correspondences and to distribute error over the sequence. Speed not being a critical issue, these algorithms take advantage of time-consuming but effective techniques such as bundle adjustment.

Many other methods perform the same task for real-time applications but tend to be less reliable since they can not rely on batch computations. Those that work without *a priori* knowledge are not really practical: for example [4] assumes that no correspondences errors occur, and [5] assumes that the camera center is moving to check if the correspondences respect the epipolar constraint. Some popular methods[6] require fiducials for an accurate registration. Model-based approaches such as [7], [8] are reliable and try to compute a 3D pose that correctly re-projects the features of a given 3D model into the 2D image. These features can be edges, line segments, or points. To find the best fit, they use least-squares minimization to find a local minimum in an error function. Unfortunately the optimization procedure may fall into wrong local minima in some particular cases. This kind of approach can track an object with acceptable accuracy, but its behaviour is unpredictable, in particular in the presence of aspect changes or even when two edges of the same object become very close to each other. Various methods derive the camera position by concatenating transformations between adjacent frames: for example [9] tracks features in the case there is a plane in the scene, making use of robust pose detection. [10] tracks natural features with no model, considering as outliers all the regions and points that do not have the same planar rigid motion. These methods give good results over short sequences, the tracking is accurate and there is no jitter because the points and/or regions matching is done with respect to very close frames. Unfortunately, for long sequences these methods suffer from the error accumulation problem; they cannot deal with severe aspect changes. Other methods [11] and [12] take into account reference frames. The first one uses a very limited number of points for template matching and keeps track of

disappearing and appearing points. The second method uses two reference keyframes for tracking the whole sequence. In fact [11] states that the results need to be smoothed by means of Kalman filtering. [12] also uses a Kalman filter for jittering correction. They propose a solution for only two offline keyframes but they do not tell how to extend their method to many keyframes.

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 1 three different times:

1. using only offline keyframe,
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 of Fig. 2(a) exhibits jitter while the recursive one of Fig. 2(b) is quickly corrupted by error accumulation. The method presented in this paper Fig. 2(c) keeps closer to the ground truth and avoid drift.

3. Simple Recursive Tracking

In this section, we outline our approach to tracking the camera-object displacement frame by frame. We assume the algorithm is applied to a pair of arbitrary frames but we do not yet make any assumption on how these frames are chosen. This is the general form of 3D object tracking by means of natural features and is summarized below.

3.1. Initialization

We use a calibration grid to compute intrinsic parameters offline. The algorithm starts when the user moves the camera or the object close to a known position that may be shown on the screen. This does not need to be done precisely, an approximate position is sufficient. The matching algorithm receives as input the incoming image and a "bootstrap" reference frame; if the frames are close enough, the point matching number increases above a given threshold and the tracking starts.

3.2. Robust Pose Estimation Through Point Matching

First, we detect the strongest interest points in the current source image using the Harris corner detector [13]. Let the interest points detected at the time t be:

$$m_t = \{m_t^0 ... m_t^n\}.$$

Given a previous frame, let m_{t-1} be the set of 2D points that we detected in it and M_{t-1} be the 3D position. Assuming that $[R|T]$ is known in the previous frame, but new parts of the object may have appeared, we want to take into account the new 2D interest points. So we back-project them in order to find their 3D coordinates M_{t-1}, keeping only the interest points that are on the object surface and discarding all the others. To do so, we first use a "Facet-ID" image to detect on which face of the 3D model each 2D point lies. That image is generated by encoding the index i of each facet f_i as a unique color, and projecting the whole model into the image plane, using a standard OpenGL rendering. Once the facet-ID is known we can use the efficient algorithm presented in [14] to find the intersection with the found facet and the line passing through the camera centre of projection and the 2D point in the image plane. Being the 3D position M_{t-1} in that frame known:

$$\begin{aligned} m_{t-1} &= \{m_{t-1}^0 ... m_{t-1}^n\} \\ M_{t-1} &= \{M_{t-1}^0 ... M_{t-1}^n\}, \end{aligned}$$

such that:

$$m_{t-1}^i = A[R_{t-1}|T_{t-1}]M_{t-1}^i;$$

where M_{t-1} and R_{t-1} and T_{t-1}, the camera rotation and translation estimated for the previous frame, are expressed in the object coordinate system. A is the internal parameters matrix. We are looking for the R_t and T_t matrices for the current frame. We match the 2D points between m_{t-1} and m_t, choosing for each point in the set m_{t-1} the one in the set m_t that maximizes a correlation measure that is insensitive to illumination changes [15]. As a result, some of the current image points m_t are matched to the previous image points m_{t-1}:

$$m_t^j \leftrightarrows m_{t-1}^i.$$

Since M_{t-1}^i must re-project on m_t^j we should have:

$$A[R_t|T_t]M_{t-1}^i = m_t^j.$$

Therefore also the 3D points belonging to M_t^j can be associated to the 3D points of M_{t-1}^i, giving in this way the 3D coordinates of the unknown points:

$$M_t^j = M_{t-1}^i.$$

The 3D points are the same for both the images if the 2D points have been correctly matched. Once all the 2D-3D correspondences are done, we have enough information to compute the camera position in the object reference system. This is done using the algorithm proposed in [16] and the robust estimator RANSAC to discard outlier matches [15].

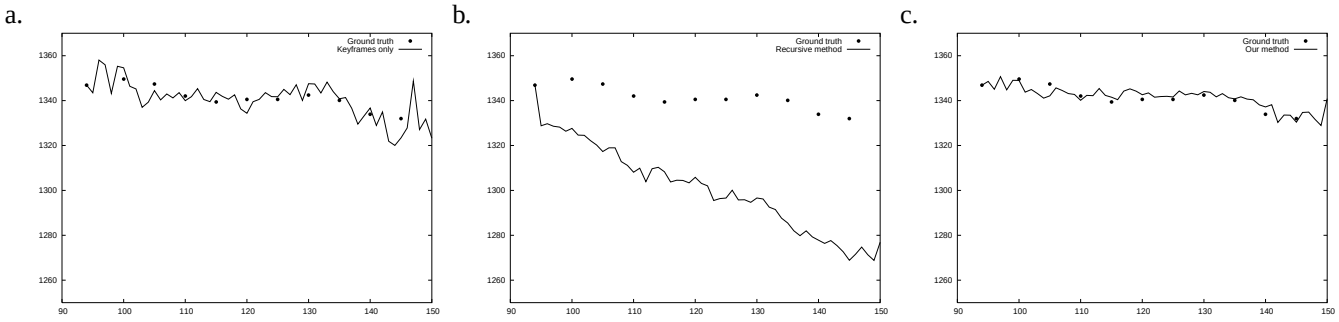

Figure 2: Plots showing a sequence tracked using three different methods. The dots represent the ground truth. The first plot shows the low precision and jittering resulting from using only offline keyframes, the second one highlights the error accumulation of the recursive method. The third plot corresponds to our method.

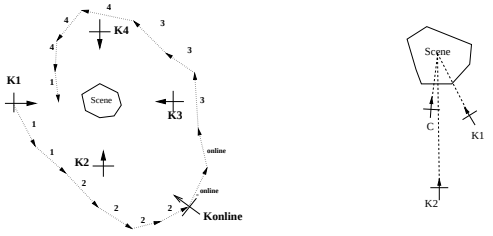

Figure 3: Online and Offline Keyframes. a) Tracked camera displacement with four offline keyframes and one online keyframe. The dotted arrows represent the camera displacement from one frame to the next, and the number shows which keyframe is being used. K1 to K4 are the camera positions of the offline keyframes. When the current camera position gets too far from any known offline keyframe, a new online keyframe denoted Konline is generated. b) Choosing the best keyframe between $K1$ and $K2$. C is the previous camera position.

4. Keyframe Based Tracking

In short, the simple method presented in the previous part works with very good precision without jittering. However, the simple recursive approach is too weak from the point of view of error accumulation, and it is not suitable for a real-time environment. Thus, one must improve the method with some additional information. This can be done using some *a priori* knowledge, supplied by the keyframes. The following section explains how to use the keyframes in order to track any sequence with no drift and no limits on the camera position.

In this section, we explain how to build the keyframe offline set, how to track by matching the reference keyframes, and which criterion we use to choose the best keyframe for the match. During the training stage the user creates offline keyframes, and in the tracking stage the previous information is used to track. During the tracking the camera may move too far away from any known keyframe: In that case a new "online" keyframe is added to the others and it will be re-used when the camera position again passes close to it.

4.1 Creating Keyframes

During the offline stage, the user is asked to choose a set of images representing the scene from many different points of view or, at least, the positions that the camera will probably reach. Usually it is enough to take few pictures all around the object. While tracking the sequences presented in this paper we only used 14 keyframes because our object is rather big and we can only walk around it; for the head we used only one keyframe. Afterwards the user is asked to accurately calculate the $[R|T]$ for every key frame he choosed. There are many methods to calculate the $[R|T]$. In our early test stage we were using a simple Posit implementation: it is enough to get the 2D position of 3 known points in every image to calculate the object pose. The user can even make use of commercial post-production tools, such as the ones of RealViz$^{(tm)}$ or 2D3$^{(tm)}$. The commercial products can retrieve the object position over the whole sequence with good accuracy, since they work offline. When $[R|T]$ is known for every keyframe, the user has completed the offline stage. Then the system performs interest point detection and back-projects the points that lie on the object surface. In short, building a keyframe means collecting the following data for each frame: The deinterlaced bitmap image of the scene, the two sets m and M of 2D and 3D points, the corresponding surface normals $\vec{n}$, which will be used for wide baseline matching, R and T, and some additional information for the visibility criterion.

4.2. Visibility Criterion for Keyframe Choice

The first step is to choose the best keyframe. This choice is a critical task on which the quality of the matching depends. The keyframe's aspect must be as close as possible to the

current frame. As shown in Figure 3b, simply evaluating the camera position is not enough. The point C represents the current camera position, and K1 and K2 are two keyframes. Just taking the keyframe that minimizes the euclidian distance means that the closest keyframe is K1. However its aspect is not as close as K2, which is further away but has a closer line of sight. To correct this problem, we should evaluate the angle between the two lines of sight. However, this is still not a complete method, because it does not take into account object non convexities and self occlusions. Instead, we use an appearance-based method. We use the following criteria:

$$\sum_{\forall f \in Model} \left(\text{Area}(f, A_P[R_P|T_P]) - \text{Area}(f, A_K[R_K|T_K])\right)^2,$$

where $\text{Area}(f, P)$ is the 2D area of the facet f after projection by P. We reuse the method we introduced in Subsection 3.2 for an accelerated OpenGL rendering of the object model. Every facet is rendered in a different color, representing the facet index, using the camera R and T estimated for the previous frame. We histogram this image and compare the result to the keyframe histograms, which have been created offline during the learning stage. We get the contribution of the area of every single facet in the model as it is reprojected in the 2D image. Every histogram bar represents the number of occurrences of every facet's pixels. This method has constant complexity, and requires only a single read of the image. In Figure 5 we show the rendered images and the correspondent histograms respectively for C, K2 and K1, the camera positions given in Figure 3.b.

4.3 Wide Baseline Matching

This section presents our method to handle the perspective distortion on the correlation window. Conventional methods make use of a square bi-dimensional correlation window. This technique gives good points matching under the assumption of very small perspective distortion between two frames. However, to effectively use keyframes, the ability to match distant frames becomes essential. Consequently we specify a point matching algorithm between a square 2D window in the current frame and a perspective distorted window in the keyframe image, that we call the "re-rendered" image. We skew the 30×30 pixel patches around each interest point from the keyframe image in order to bring them to a position close to the current one. Each patch in the keyframe is related to the corresponding image points in the "re-rendered image" by a planar homography. Given the patch corresponding plane π having coordinates $\pi = (\vec{n}^T, d)^T$ so that for points on the plane $n^T X + d = 0$, the general expression for the homography induced by the plane is (according to [15]):

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

Figure 4: From the left: The keyframe, the current frame, the re-rendered key frame with respect to the previous camera position estimate.

for two views defined by their projection matrices $P = A[I|0]$ and $P' = A'[R|t]$. The homography equation for the general case can easily be obtained by changing the reference system. We get:

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

with

$$\delta R = R_P R_K^T;\ \delta t = -R_P R_K^T t_K + t_P;$$
$$\vec{n'} = R_K \vec{n};\ d' = d - t_k^T(R_K \vec{n}).$$

where $A_K[R_K|t_K]$ and $A_P[R_P|t_P]$ are the projection matrices of the key frame and the previous frame.

The resulting image is a re-rendering of the interest points' neighbourhood in a more convenient position as shown in Figure 4. This method allows us to effectively match views even where there is as much as 60 degrees of rotation. An alternative solution to the homography would have been to re-render a 3D representation of the object using an OpenGL textured 3D object, but we choose the other way to have a more precise result around the points.

4.4. Combining Online and Offline Information for Jittering Correction

There is a trade-off between accurate tracking with no jittering and a robust tracking with no drift. Tracking with respect to previous frames offers better precision than keyframes but involves error accumulation. Tracking with the keyframes is less precise because usually fewer points are matched, giving a poor precision as we have seen. This is due to the distance between the two frames we are trying to match.

In our approach we therefore attempt to combine the strengths of both the online and offline information as follows: first, we match the current frame with the chosen keyframe and apply RANSAC to the set of points we found, discarding the outliers and retaining a set of points M_K free from error accumulation. Then, we perform a modified RANSAC estimation over the matches between the current frame with the previous one: if an $[R, T]$ sample tested by the RANSAC estimator rejects some points in M_K, this sample is not considered by this second stage. This way,

this stage estimates the values $[R, T]$ using all the points in M_K, which provide reliable but partial information, and the matches between the previous and the current frames that provide additional information.

As we will show in the results section, this technique eliminates jitter without requiring predictive techniques such as Kalman filtering that are not particularly suitable for Augmented Reality.

4.5 Offline and Online Keyframes

Assuming we already have a consistent set of keyframes, we show in this subsection how to employ them to track a sequence. As shown in Figure 3a, while the camera moves around the scene, the system switches from one keyframe to the other, always choosing the one that is is closest to the images currently being seen. When the current camera position gets too far from any known offline keyframe, a new online keyframe denoted Konline is generated. It will be added to the keyframe set and treated like the other ones. The criterion we use for deciding to generate an online keyframe is a test on the matched point number and the robust pose discarded points number. As it is always based on previous frames, this method might potentially suffer from the same drift problem as the recursive method. However, the drift is not a problem in this case. We accumulate error only when we create an online keyframe, since we calculate a new 3D position based on the $[R|T]$ that we computed and not from the real one. However, this error accumulation does not occur at every frame because the newly generated keyframe can be reused for tracking many frames. For example, in our sequences an online keyframe can be used for tracking 40 or 50 consecutive frames. Moreover, after some time the camera will again pass close to a known position, re-using the keyframes that have been generated online. An interesting characteristic of this method is that when some error has been accumulated over a part of the sequence, it will be reset to zero when an offline frame is used. The online frames can be considered as a kind of "second chance" method used to recover when there are no offline keyframes, and it has only to guarantee no complete divergence before the camera gets close to an offline frame.

5. Experiments and Results

The non optimized version of the tracker runs at near real-time, at about 4 frames per second using a conventional machine for 720×568 images and about 15 frames per second for 320×200 images. Since, for many critical geometric computations, we used general methods based on openGL rendering, our method can work with difficult objects at the same speed as simple ones. The speed depends only on the number of interest points lying on the object, which can be

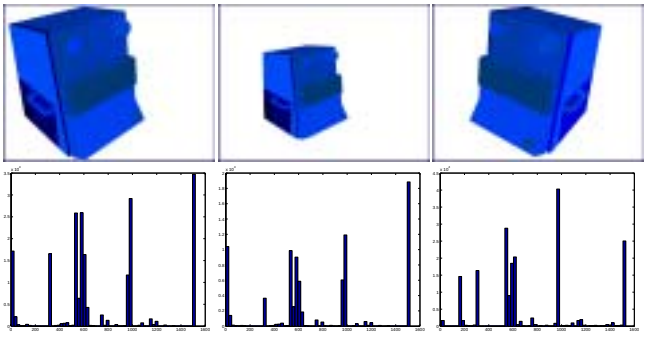

Figure 5: Different aspects for the camera positions C, K2 and K1 in Figure 3.c, and their respective histograms.

Figure 6: Some keyframes for the projector sequences.

decreased or increased simply by changing the corner detection threshold. The tracked object can assume any position with respect to the camera. For instance, there is no problem if the camera is inside the object to be tracked, and we can handle compound objects, or self-occluding parts of the scene, as long as they do not move with respect to each other. We set up a variety of demonstrations, to show that this method can be used for many different categories of objects. In the example depicted by Figure 1d, the camera is moving around an old video projector doing a complex movement with 180 degree rotations. In a second video sequence the scene is partially occluded by a human operator, in order to roughly simulate the behaviour of a camera on a Head Mounted Display (see figure 8). Not all the videos are attached to the paper, for example a 360 degree camera displacement around the projector is omitted (see Figure 9). We use the same 14 offline keyframes for all the sequences. Some of them are shown in Figure 6. Compared to some of the online sequences, they have different light conditions and the camera was farther from the projector. In the MPEG video corresponding to the Figure 1d for every frame are shown the current keyframe used for tracking and its re-rendering (in the top right corner). Some keyframes are online and some are offline. The model was created by a designer using Maya, and it took 4 hours of work. There is a small mistake in the model: the position of one of the two cylinders on the front face is not very accurate, however it does not corrupt the result, though some points are refused by the robust pose estimation.

In the second example depicted by Figure 1, we track a

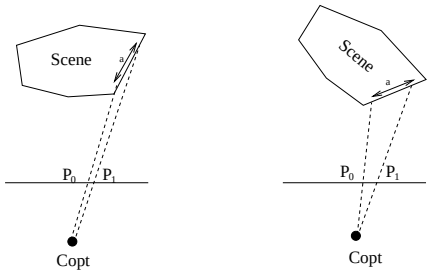

Figure 7: The reprojection error when a face of the model is almost parallel to the line of sight (left) and in the opposite case.

head that is rotating completely. The model has been reconstructed offline from another short video sequence. Even though we only have the face model and not the whole head, have been able to track 180 degree rotations . We ran our tracker on this sequence giving only one offline key frame. Figure 1 shows some frames with the face model superposed, and the last ones do not show the model but some virtual objects have been added. Since the occlusions are evaluated by means of the same face model (missing ears and the back of the head), not all the occlusions are perfect. We believe that the head can be tracked by means of much less accurate models, as in [17], however our intention is to demonstrate that we can deal with complex objects. All the video sequences are available at the address: http://cvlab.epfl.ch/˜vacchetti/research.html

6. Discussion

In this section we discuss the problems arising when the tracked objects go through aspect changes and we illustrate how we correct the drift. If there are only partial aspect changes, and the points are regularly distributed over the tracked object, the errors may not be accumulated quickly, and may even cancel each other when moving in opposite directions. In this way long sequences of over one thousand frames may be successfully tracked without encountering much drift. In Figure 7 we analyze a camera rotating around the scene. We evaluate a small error in the pixel space, e.g. say that the point P_0 is assumed to be in the position P_1 in the left part of the image. If these pixels represent a face that is almost parallel to the line of sight, the error a in the 3D position of the point is very large. If the object does not change its aspect, we are still in a safe state, but if the camera turns and the side is facing the camera (Figure 7b), the re-projection of 3D position error will be much bigger than its previous reprojection. If at this point we do ray casting — for adding new incoming points to our set — many background points are considered as lying on the object and the tracking will be corrupted, since the robust pose is fooled by this wrong information. Our experiments show that, without drift correction, the tracking may fail after less than 180 degree rotation, that may roughly correspond to 100 frames in our sequences, which is much faster than when there is only simple camera displacement.

7. Conclusion

In this paper we presented a robust and jitter-free tracker that combines natural feature matching and the use of keyframes to handle any kind of camera displacement using real-time techniques. We use the model information to track every aspect of the target object, and to keep following it even when it is occluded or only partially visible, or when the camera turns around it. A set of keyframes is created off-line and, if there are too few of them, new keyframes can be automatically added online. We exploit offline and online information to prevent the typical jittering and drift problems. The matching algorithm is designed to match frames having very different aspects and in the presence of rotations of up to 60 degrees. We choose the most appropriate keyframe using aspect-based techniques and we exploit hardware accelerated functions to implement many critical parts. We can use our tracker for a large set of objects, with no constraints on the kind of camera motion.

Our plans for future work include offline bundle adjustment after the end of tracking in order to achieve perfect registration of the online keyframes, automatically creating new offline keyframes. Further development will be done to incrementally extend our scene model during tracking, exploiting the camera displacement information to retrieve additional points following the same rigid motion of the model. In this way every time the program runs it improves its performance.

References

[1] C. Tomasi and T. Kanade, "Shape and Motion from Image Streams under Orthography: A Factorization Method," *International Journal of Computer Vision*, vol. 9, no. 2, pp. 137–154, 1992.

[2] A.W. Fitzgibbon and A. Zisserman, "Automatic Camera Recovery for Closed or Open Image Sequences," in *European Conference on Computer Vision*, Freiburg, Germany, June 1998, pp. 311–326.

[3] M. Pollefeys, R. Koch, and L. Van Gool, "Self Calibration and Metric Reconstruction in Spite of Varying and Unknown Camera Parameters," in *ICCV*, 1998, pp. 90–96.

[4] A. Azarbayejani and A. P. Pentland, "Recursive Estimation of Motion, Structure and Focal Length," *IEEE Transactions*

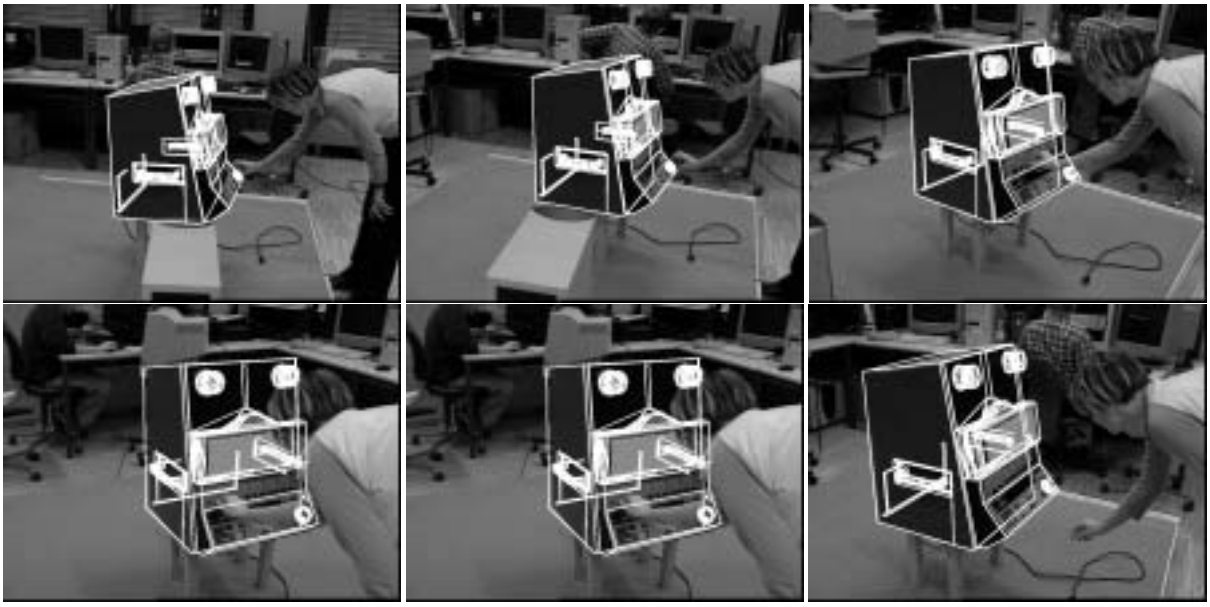

Figure 8: Video sequence with occlusions.

on Pattern Analysis and Machine Intelligence, vol. 17, no. 6, pp. 562–575, 1995.

[5] P. A. Beardsley, A. Zisserman, and D. W. Murray, "Sequential update of projective and affine structure from motion," *International Journal of Computer Vision*, vol. 23, no. 3, pp. 235–259, 1997.

[6] K. N. Kutulakos and J. R. Vallino, "Calibration-free augmented reality," *IEEE Transactions on Visualization and Computer Graphics*, vol. 4, no. 1, pp. 1–20, /1998.

[7] T. Drummond and R. Cipolla, "Real-time tracking of multiple articulated structures in multiple views," in *ECCV (2)*, 2000, pp. 20–36.

[8] E. Marchand, P. Bouthemy, F. Chaumette, and V. Moreau, "Robust real-time Visual Tracking Using a 2D-3D Model-Based Approach," in *International Conference on Computer Vision*, Corfu, Greece, September 1999, pp. 262–268.

[9] G. Simon, A. Fitzgibbon, and A. Zisserman, "Markerless tracking using planar structures in the scene," in *Proc. International Symposium on Augmented Reality*, October 2000, pp. 120–128.

[10] U. Neumann and S. You, "Natural feature tracking for augmented reality," *IEEE Transactions on Multimedia*, vol. 1, no. 1, pp. 53–64, 1999.

[11] S. Ravela, B. Draper, J. Lim, and R. Weiss, "Adaptive tracking and model registration across distinct aspects," in *IEEE International Conference on Intelligent Robots and Systems (IROS)*, 1995, pp. 174–180.

[12] K.W. Chia, A.D. Cheok, and S.J.D. Prince, "Online 6 dof augmented reality registration from natural features," in *Proc. International Symposium on Mixed and Augmented Reality*, 2002.

[13] C.G. Harris and M.J. Stephens, "A combined corner and edge detector," in *Fourth Alvey Vision Conference, Manchester*, 1988.

[14] T. Moeller and B. Trumbore, "Fast, minimum storage ray-triangle intersection," in *Journal of graphics tools, 2(1):21-28*, 1997.

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

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

[17] M. Cascia, S. Sclaroff, and V. Athitsos, "Fast, reliable head tracking under varying illumination: An approach based on registration of texture-mapped 3d models," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 4, April 2000.

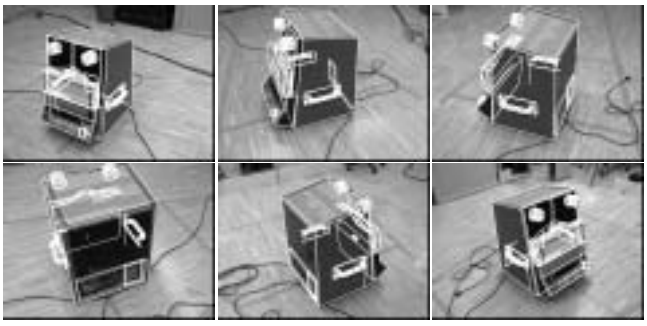

Figure 9: Video sequence in which the camera is rotating around the object doing a 360 degree loop.