Stable Real-Time 3D Tracking using Online and Offline Information*

L. Vacchetti, V. Lepetit and P. Fua
Computer Vision Lab
Swiss Federal Institute of Technology (EPFL)
1015 Lausanne, Switzerland
E-mail: luca.vacchetti, vincent.lepetit, pascal.fua@epfl.ch

Abstract

We propose an efficient real-time solution for tracking rigid objects in 3D using a single camera that can handle large camera displacements, drastic aspect changes, and partial occlusions. While commercial products are already available for offline camera registration, 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 address these problems, we have formulated the tracking problem in terms of local bundle adjustment and have developed a method for establishing image correspondences that can equally well handle short and wide-baseline matching. We then can merge the information from preceding frames with that provided by a very limited number of keyframes created during a training stage, which results in a real-time tracker that does not jitter or drift and can deal with significant aspect changes.

Index Terms

Computer vision, Real-time systems, Tracking.

I. Introduction

In this paper we propose an efficient real-time solution to single-camera 3D tracking that can handle large camera displacements, extreme aspect changes and partial occlusions. While commercial products are already available for offline camera registration, robust online tracking remains an open issue because it must be fast and reliable. 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 overcome these problems, we have developed an algorithm that merges the information from preceding frames in traditional recursive fashion with that provided by a very limited number of reference images, or *keyframes*. This combination results in a system that does not suffer from any of the above difficulties and can deal with aspect changes and occlusions. In essence, we combine the strengths of methods that rely on absolute information, such as keyframes, with those based on chained transformations. The former do not drift but cannot provide enough precision for every frame without a very large amount of absolute information, which results in jitter. The latter do not jitter but tend to drift or even to lose track altogether.

To this end, we have formulated the tracking problem as one of local bundle adjustment in such a way that it can be solved very quickly. For a particular frame, the input data are correspondences computed using a fast technique that can handle both short and wide-baseline matching and, thus, can deal equally well with preceding frames, seen from relatively similar viewpoints, and keyframes whose viewpoint may be quite different. 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

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

Fig. 1. First row: Video sequence with overlaid 3D model whose pose has been computed online using eight keyframes. Second row: We track a face using a generic model that has not been tailored for this specifi c man using one keyframe. Third row: The same method is used to track a 1000 frames video sequence of a corridor. We used four keyframes. The corresponding video sequences are submitted as supplementary material.

new ones as it runs. It requires a 3D 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 3D tracking. Furthermore, such models 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 consider any object that can be represented by a 3D mesh, such as those shown in Fig. 1.

II. Related Work

Offline camera registration from an image sequence [1], [2] has progressed to the point where commercial solutions have become available. By matching natural features such as interest points between images these algorithms achieve high accuracy even without *a priori* knowledge. Speed and causality not being critical issues, these algorithms take advantage of time-consuming but effective batch techniques such as global bundle adjustment.

By contrast, real-time registration methods tend to be less reliable since they can not rely on batch computations. Those that work without *a priori* knowledge impose constraints that are not always practical in complex environments: For example, [3] assumes that there are no correspondences errors, and [4] that the camera center is moving in such a way that one can check if the correspondences respect the epipolar constraint. The method presented in [5] uses robust pose detection to track features when there is a plane in the scene using chained homographic transformations, which may result in drift. Similarly, [6] tracks natural features and treats as outliers all the regions and points that do not have the same planar rigid motion. More recently, a method that relies on the fi ve-point algorithm [7] combined with bundle-adjustment over the last few frames has been shown to yeld good results if one is willing to tolerate a short delay in processing [8]. However, most of the methods of this nature derive the camera position by concatenating transformations between adjacent frames. Over short sequences, the tracking may be accurate and jitter-free but, over long ones, these methods often suffer from error accumulation, which produces drift, and cannot deal with severe aspect changes.

(a) (b) (c)

Fig. 2. Models and keyframes. (a) Reference views are acquired, registered and used to build the models. (b,c) Models used to track the projector and the corridor of Fig. 1.

Model-based approaches, such as those proposed in [9], [10], [11], avoid drift by finding camera poses that correctly re-projects some fixed features of a given 3D model into the 2D image. These features can be edges, line segments, or points. The best fit is found through least-squares minimization of an error function, which may lead to spurious results when the procedure becomes trapped in erroneous local minima. As a result, such approaches will track objects with acceptable accuracy for a while, but their behavior can become erratic when aspect changes occur, or simply when target object features become very close to each other in projection. Using optical flow in addition to edge-information has been shown to help [12] but is sensitive to illumination effects. Similarly, texture-based approaches [13], [14], [15], which rely on global features such as the whole pattern of a face, do not drift but are typically less reliable than feature-based methods. They can easily fail when even a small part of the object moves out of the scene or when there a strong illumination change occurs.

Using some form of absolute information is one way to eliminate the failure modes described above. For example, in [16], a very limited number of points are used for template matching against keyframes and the system keeps track of disappearing and appearing points. Similarly, in [17], two reference frames are used for tracking the whole sequence. Both approaches require smoothing by means of Kalman filtering for jitter correction and exploit only two offline keyframes, without providing obvious ways to use more if they are available.

The approach presented in [18] solves some of the problems by combining absolute information from 3D edges with feature point tracking by enforcing the epipolar constraint over pairs of successive frames. Drawbacks of this method are that the camera center must translate and that the fundamental matrix computation for pairs of close frames can be unstable. This paper describes a method that has no such limitations and extends an earlier publication [19].

III. Tracking as Model Based Bundle Adjustment

As discussed above, both the recursive and absolute information based-approaches to real-time tracking have advantages and drawbacks. Our contribution is to combine their strengths to eliminate both drift and jitter. To this end, we formulate tracking in terms of a bundle-adjustment problem and show how it can be made to run in real-time without giving up accuracy.

In this section, we first show how we exploit 2D correspondences between incoming images and keyframes to compute a rough registration. We then refine our approach by also taking into account the information provided by neighboring frames. The strategy we use to establish the correspondences will be discussed in the following section. In the remainder of this section, we will assume that the internal camera parameters are known and fixed. In Section V, we will show that, in practice, rough estimates suffice because the algorithm is relatively insensitive to changes in focal length.

A. Matching Against a Key Frame

Let us assume that during a training phase, some images of the target object have been captured and registered using convential techniques based on 2D-3D correspondences, as will be discussed in more

detail in Section III-D. We use a standard corner detector [20] to extract 2D feature points and compute their 3D positions by back-projecting them onto the object model. We will refer to these images and associated pose parameters as *k*eyframes.

These keyframes could serve to register an incoming image at time t using a simple approach: Let the m_t^i be the 2D feature points the corner detector extracts from it. By matching them against those found in keyframe r and whose 3D position is known, we establish 2D-3D correspondences, that is we associate to m_t^i the 3D point $M_r^{v(i)}$, where $v(i)$ is an index in the set of keyframe points. The matching is performed using the correlation-based technique described in Section IV. The simplest way to estimate the camera position would then be to minimize the reprojection error

$$r_t = \sum_{i=1}^{k} \rho_{TUK}\left(\left\|m_t^i - \phi\left(P_t, M_r^{v(i)}\right)\right\|^2\right) \quad , \tag{1}$$

with respect to the orientation and translation parameters that defi ne P_t, the pose at time t, where

- $\phi\left(P, M\right)$ denotes the projection of 3D point M given the pose P;
- ρ_{TUK} is the Tukey M-estimator used for reducing the influence of wrong matches [21].

However, in practice, this would result in jitter because the successive camera positions would be recovered independently for each frame. As discussed below, we avoid this by also considering correspondences with neighboring frames.

B. Taking Neighboring Frames Into Account

We can also establish correspondences between feature points found in the current frame and in the ones preceding it. We use the technique of Section IV to extract and match these points at each iteration. Let the n_t^i be 2D coordinates of point i in frame t and let N^i be the corresponding, unknown, 3D coordinates. We could incorporate this additional information by minimizing

$$\sum_{j=1}^{t} r_j + \sum_{i=1}^{k} \sum_{j \in \Theta^i} \rho_{TUK}\left(\left\|n_j^i - \phi\left(P_j, N^i\right)\right\|^2\right) \quad , \tag{2}$$

where r_j is defi ned in Eq. 1, with respect to $P_1 \ldots P_t$ the camera poses up to time t, and to the 3D points N^i, where Θ^i is the set of frames where the i-th feature appears. In practice, to limit the required amount of computation, we restrict ourselves to the current and previous frames. The problem then becomes minimizing

$$\min_{P_t, P_{t-1}, N^i} \left(r_t + r_{t-1} + \sum_i s_t^i\right) \quad , \tag{3}$$

with

$$s_t^i = \rho_{TUK}\left(\left\|n_t^i - \phi\left(P_t, N^i\right)\right\|^2 + \left\|n_{t-1}^{w(i)} - \phi\left(P_{t-1}, N^i\right)\right\|^2\right) \quad , \tag{4}$$

where the interest point n_t^i detected in the current frame is matched with the point $n_{t-1}^{w(i)}$ detected in the previous frame.

C. Transfer Function

The formulation outlined above would still result in a computationally intensive algorithm if we treated the 3D coordinates of the N^i as optimization variable. However, as shown in [22], we can exploit the fact that the N^i are on the surface of the 3–D model to eliminate those coordinates from the s_t^i terms of Eq. 4 and, thereby, drastically reduce the number of unknowns by possibly several hundreds, three per point. Only 12 unknowns, six for the pose at time $t-1$, six for the pose at time t, need to be optimized.

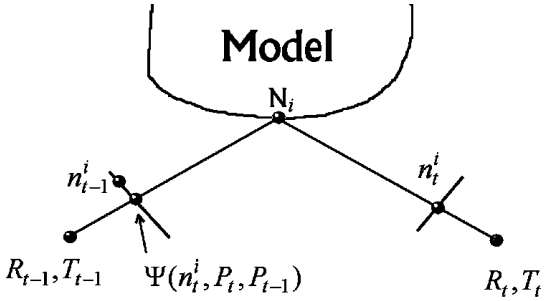

Fig. 3. The transfer function Ψ backprojects n_t^i in frame t to the model's surface at N_i and reprojects N_i into frame $t-1$.

First, note that the optimization of Eq. 3 can be rewritten as

$$\min_{P_t,\, P_{t-1}} \left(r_t + r_{t-1} + \min_{N^i} \sum_i s_t^i \right) \tag{5}$$

since r_t and r_{t-1} are independent of the tracked points N^i. Instead of estimating the N^i, the s_t^i terms can be approximated using a transfer function that involves only the point projections. As shown in Fig. 3, given a point n in the first frame and the poses P and P of the two frames, such a transfer function $\Psi(n, P, P')$ returns the point n' such that there is a 3D point N belonging to the model surface that satisfies $n = \phi\,(P, N)$ and $n' = \phi\,(P', N)$. s_t^i can then be approximated as

$$s_t^i = \rho_{TUK}\left(\left\| \Psi\left(n_{t-1}^{w(i)}, P_{t-1}, P_t \right) - n_t^i \right\|^2 \right) \quad . \tag{6}$$

This formulation, however, is not symmetric because the current and previous frames play different roles. To make it symmetric, we take s_t^i to be

$$s_t^i = \rho_{TUK}\left(\left\| \Psi\left(n_{t-1}^{w(i)}, P_{t-1}, P_t \right) - n_t^i \right\|^2 + \left\| \Psi\left(n_t^i, P_t, P_{t-1} \right) - n_{t-1}^{w(i)} \right\|^2 \right) \quad . \tag{7}$$

This is a simplified version of the formulation presented in [22], which results in a straightforward implementation that is well adapted to our real-time constraints.

In theory, computing the transfer function Ψ can be expensive. In practice, at the start of each optimization, we use the pose estimated for the previous frame to quickly estimate to which facet a 2D point backprojects and take Ψ to be the homography induced by that facet. To further speed up this process, we use a "Facet-ID" image created by encoding the index *i* of each facet of the target object's model as a unique color and using OpenGL to render the model into the image plane. Given a 2D point which is the projection of a 3D point of the model, the mentioned method allows us to efficiently find the facet this point belongs to.

D. Keyframes

At each time step t, the system selects a keyframe to evaluate the r_{t-1} and r_t terms of Eq. 5. Keyframes are therefore central to our approach since they provide the absolute information that makes our system robust. Here, we first briefly explain how they are built during training. We then describe how we switch from one to another at runtime and create new ones as needed.

During training, the user is asked to choose a set of images representing the target object from one viewpoint, as in the case of the face of Fig. 1, or more, as in the case of the projector or the corridor of Fig. 1. Given this set, we use software packages such as those sold by RealViz^{tm} or 2D3^{tm} to compute the P pose parameters of Section III-A for each keyframe. Because they can rely on batch computation, their results can be expected to be reliable. For example, ImageModeler from Realviz can be used to estimate the pose parameters while interactively build the 3D model. In practice, this task takes few hours, depending of the object complexity. If the 3D model is known, calibrating few keyframes is a

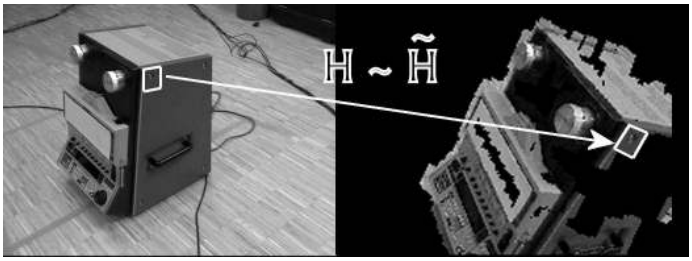

Fig. 4. Pixels around interest points are transferred from the keyframe (left) to the re-rendered image (right) using a homography, which can be locally approximated by an affine transformation.

matter of minutes. As discussed above, our system then detects interest points, back-projects those that lie on the object surface, and stores the results in a database, thus completing its training.

Choosing the right keyframe to use when minimizing the criterion of Eq. 5 is an important task on which the quality of the matching depends. The keyframe's aspect must be as close as possible to that of the current frame. Since the camera position for the current frame is not yet known, we choose the keyframe closest to the camera position and orientation estimated for the previous frame, according to a Mahalanobis distance that takes into account both the position and orientation of the camera.

When the current camera position gets too far from any known keyframe, the number of matches may get too low to provide a satisfactory registration. To avoid restricting the range of possible camera motions, new *online* keyframes are automatically created at runtime when the pose becomes too different from those stored in the database. When the number of inlier matches drops below a threshold, an online keyframe is generated from a previous frame for which the estimated viewpoint is more reliable. For such online keyframes, the viewpoint accuracy can not be expected to be as accurate as the ones for offline keyframes, and therefore can result in a slight loss of tracking accuracy. Nevertheless, this problem will disappear when the camera comes close again to an offline keyframe.

IV. Short and Wide Baseline Real-Time Matching

Our algorithm relies on matching 2D feature points across pairs of images that may or may not be consecutive. When matching points between consecutive frames, the two viewpoints are fairly similar and we use a simple matching strategy: Each point in the first image is matched to the point in the second image that maximizes a normalized cross-correlation score among those in its image neighborhood. This process is repeated by reversing the roles of the two images and we keep the point pairs which are matched to each other. Normalized cross-correlation is only invariant to illumination changes when the material is Lambertian. We can nevertheless handle objects with specularities as long as they are local and we can correctly match enough feature points elsewhere. This is for example the case for the human face of Fig. 1.

When matching an input image against the keyframe one, the viewpoints the two pictures have been taken may be considerably different. In order to perform wide baseline matching in real time we must extend our matching strategy. Therefore, we synthesize an *intermediate image* as depicted by Fig. 4 by skewing the pixel patches around each interest point from the keyframe viewpoint to the previous frame viewpoint, having an aspect that is always close to the current image. By locally approximating the object surface around the interest points by a plane, each patch in the keyframe can be related to the corresponding image points in the re-rendered image by a homography. Given the plane π associated to a patch around interest point m_0 and represented by its normal $\overrightarrow{n}$ and its distance to the origin d, the homography H it induces can be expressed simply [23].

Let $P = A[R \mid T]$ be the projection matrix of the input viewpoint, where A represents the intrinsic parameters, R the rotation and T the translation. Similarly, let $P_K = A_K[R_K \mid T_K]$ the projection matrix

Interest point extraction	8 ms
Matching (KF+previous Frame)	15 ms
Robust viewpoint estimation	7 ms
Local Adjustment	10 ms

Fig. 5. Computation times on a Pentium IV, 2.6GHz for our tracker, for 500 extracted interest points, on 320×240 images.

for the keyframe. H can then be written as

$$H = A_K \left(\delta R - \delta T n'^T / d' \right) A^{-1} \ , \tag{8}$$

with

$$\delta R = R R_K^T \ ; \ \delta T = -R R_K^T T_K + T \ ; \ \overrightarrow{n}' = R_K \overrightarrow{n} \ ; \ d' = d - T_K^T (R_K \overrightarrow{n}). \tag{9}$$

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

$$m' = Hm_0 + J_H(m_0) \cdot (m - m_0) \tag{10}$$

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

V. Results

In this section, we present several examples that highlight our tracker's ability to handle a wide range of situations and objects, such as the ones shown in Fig. 1.

Fig. 5 shows the computation times required to perform key operations on a 2.6 GHz PIV. They are small enough for our tracker to run at 25 frames/second on 320 x 240 images. The frame-rate falls to 15 frames/second on 640 x 480 images because the interest points extraction is heavier. The tracker is initialized by manually putting the target object in a position that is close to the one of the keyframes. It does not need to be very accurate, because the tracker starts as soon as there are enough matched points between keyframe and incoming image.

The tracker is robust to:

Aspect changes: In the example of Fig. 1, eight keyframes allow our tracker to handle the drastic aspect changes of the projector. Similarly, only four keyframes were required to track the camera trajectory in the corridor.

Model inaccuracies: The face model used to track the face of Fig. 1 was a generic face model that was not tailored for a particular person, which does not affect the tracking quality. Similarly, there is a small error in the 3D projector model of Fig. 2(b): The position of one of the two cylinders on the front face is inaccurate, which does not result in any obvious ill effects.

Occlusions: In the video of Fig. 6(a) the object goes out of the scene until only a small part of it remains visible. Then it comes back but a hand hides a large portion of it. Because the algorithm considers local features, the tracking is not disrupted by partial occlusions as long as enough feature points remain visible.

Focal length changes: Fig. 6(b) shows a sequence in which the focal length increases from 12 to 36 mm. The tracker uses a fixed value for the focal length and compensates by adjusting its estimate of the object's distance.

Scale changes: Fig. 6(c) shows some frames from a sequence where the tracked object is undergoing large scale changes with no tracking failure. As discussed in Section III-D, the tracker automatically creates new online keyframes when the scale changes too much.

Illumination changes: Fig. 6(d) shows a sequence in which the lighting decreases. The tracking remains stable even though the object ends up being barely visible.

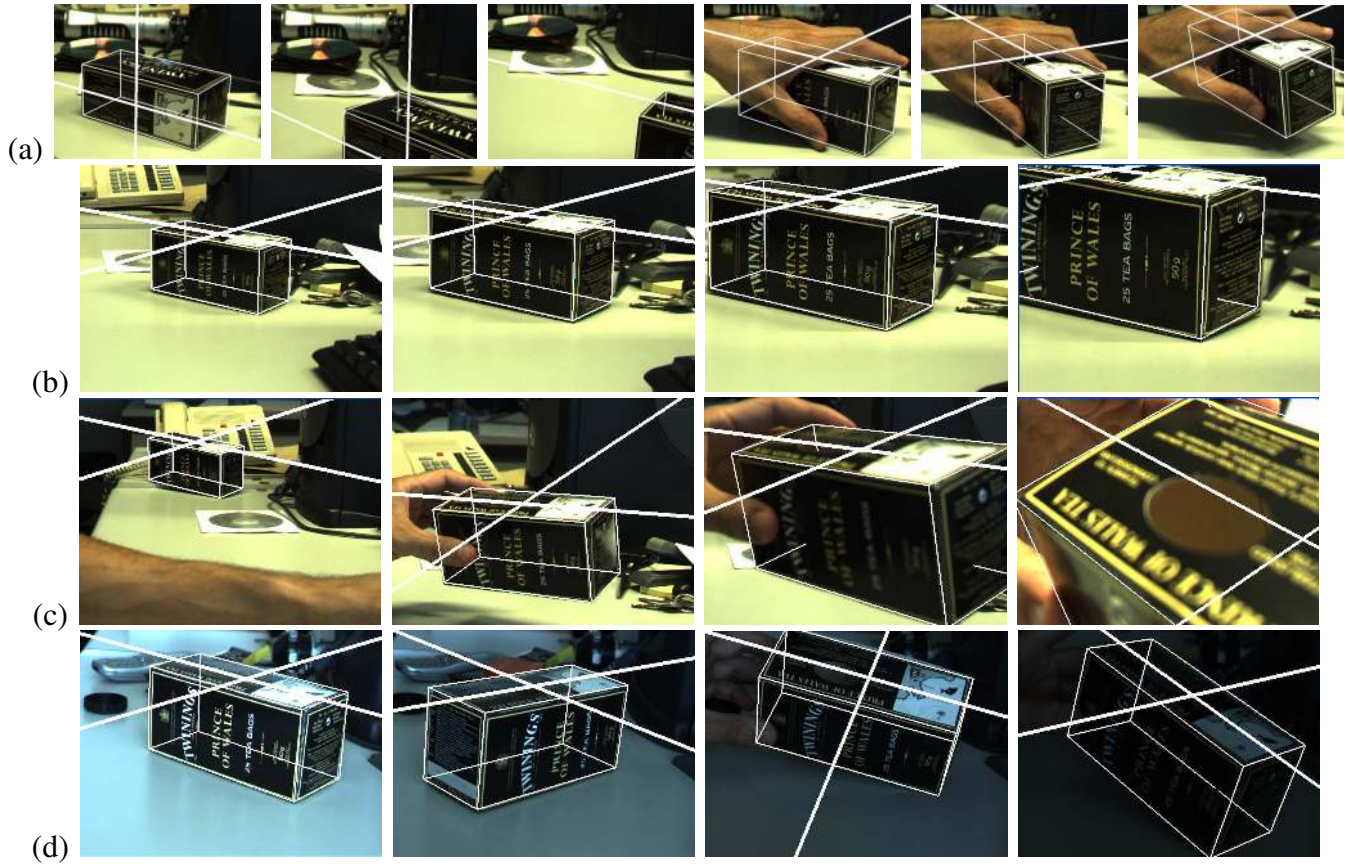

Fig. 6. Handling difficult situations. (a) Occlusions and object moving out of the picture. (b) Changes in focal length. (c) Large changes in scale. (d) Changes in illumination.

The algorithm has been successfully demonstrated at the CVPR 2003 and ISMAR 2003 conferences in Madison, WI and Tokyo where it was used to track the faces of many people and add virtual glasses and mustaches. The corresponding videos are available at

http://cvlab.epfl.ch/research/augm/augmented.html.

As many tracking algorithms, our tracker can break down in some conditions, such as when the image gets blurred or the quality is not sufficient for the feature detection. A more specific problem appears when online keyframes provide erroneous information, which can happen when the target object is mostly occluded. In such cases, the system may create a keyframe that also captures the occluding object and corrupts the result.

VI. Quantitative Evaluation

To evaluate the accuracy of our tracker, we performed the tests depicted by Fig. 7: The top and bottom rows depict the recovered evolution of the three coordinates of the camera center in the first 400 frames of the projector sequence of Fig. 1 and of the tea box sequence of Fig. 8. For comparison purposes, we also tracked the objects in those sequences using the RealViz^{tm} MatchMover package. It uses batch techniques that yield very accurate results and allows manual intervention, which we have used to ensure correctness. We can treat its output as the ground truth. Furthermore, the commercial package was run on full size images, whereas our tracker used reduced-size images to satisfy the real-time requirements. As shown in Fig. 7, the trajectory estimated by the batch techniques and the one estimated by our method remain close to each other. The median distance difference is about 1.5 cm for the tea box sequence while the camera distance to the object is about 50 cm. For the projector sequence, the median difference is approximately 4 cm.

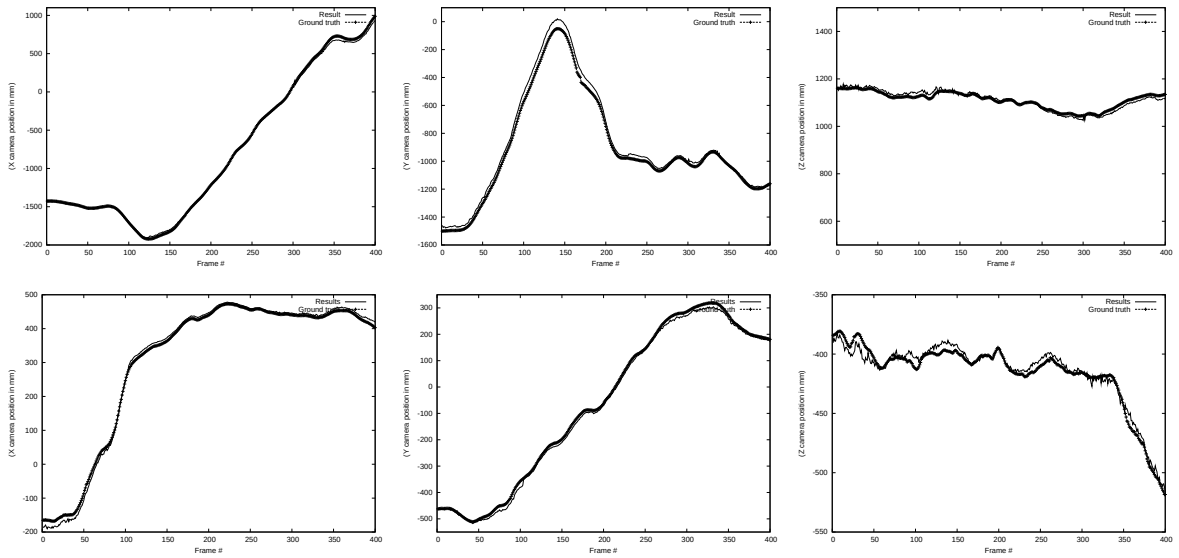

Fig. 7. Comparing against ground truth. In each plot, the thick line represents the true values of one of the three coordinates of the camera center while the dotted line depicts those recovered by our algorithm for the first 400 frames of the sequences. Top row: Projector sequence of Fig. 1. Bottom row: Tea box sequence of Fig. 8.

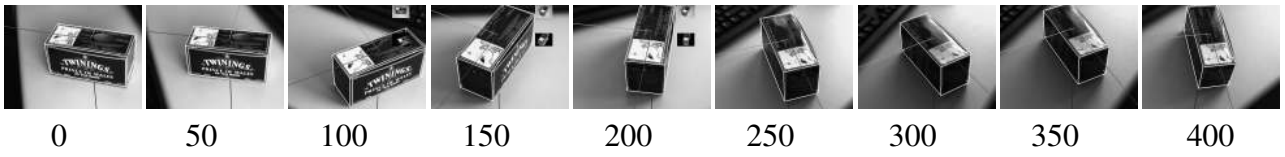

Fig. 8. A 400-frame sequence obtained by tracking a teabox and used for the quantitative evaluation of Section VI.

To demonstrate that this good performance is indeed the result of combining offline and online information, and not of either approach alone, we used our generic feature matching approach to track the projector in the sequence of Fig. 1 in three different ways:

- Matching only against the previous frame.
- Matching only against the keyframe.
- Matching against both.

As before, the plots of Fig. 9 show the evolution of one of the camera center coordinates with respect to the frame index. The first plot demonstrates that the "offline keyframes only" method suffers from jitter. The second plot depicts the recursive method, where error accumulation corrupts the tracking. The third plot represents the result of merging the two approaches, as proposed in this paper.

VII. Conclusion and Future Work

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 reference keyframes is created off-line and, if there are too few of them, new frames 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 select, at each time step, the most appropriate keyframe and we exploit hardware accelerated functions to implement many critical parts. We can use our tracker for textured objects for which a model exists. This could be further improved by integrating the contour information in order to increase the precision and to handle a larger class of objects.

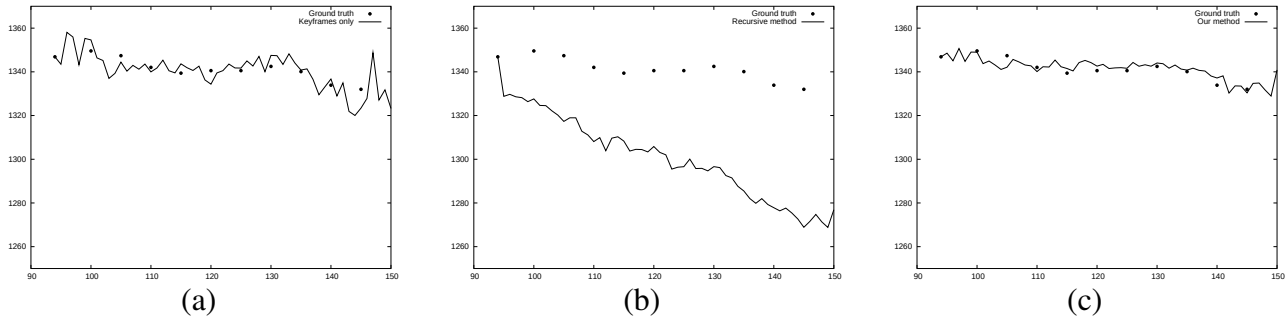

Fig. 9. Comparing three different approaches. In each graph, we plot one of the camera center's coordinates recovered using the different methods against the ground truth depicted by the individual dots. (a) Keyframe-based tracking. (b) Recursive tracking. (c) Our method that eliminates both the jitter visible in (a) and the drift visible in (b).

Future work will focus on estimating the camera position from a single image in order to initialize the tracker and to recover when it fails. Recent advances in textured 3D object recognition [24] should make this possible with good accuracy and reasonable speed. In [25], we investigated a preliminary, but promising, approach to automated initialization. During a training phase, we construct for each keyframe a database of appearances of the keypoints as they would be seen under many different views. At runtime, given a new image of the target object, the camera can then be registered online by matching the feature points present in the image against the database. This initialization method, as the tracking approach presented here, relies on natural feature points matching and has the same desirable properties such as accuracy and robustness to partial occlusions, lighting changes and cluttered background. Nevertheless it is still slow for interactive applications. Our goal will therefore be to shift some of the computational burden from runtime matching to precomputed training.

REFERENCES

[1] 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.

[2] M. Pollefeys, R. Koch, and L. VanGool, "Self-Calibration and Metric Reconstruction In Spite of Varying and Unknown Internal Camera Parameters," in *International Conference on Computer Vision*, 1998.

[3] A. Azarbayejani and A. P. Pentland, "Recursive Estimation of Motion, Structure and Focal Length," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 17, no. 6, pp. 562–575, 1995.

[4] 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.

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

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

[7] O.D. Faugeras, *Three-Dimensional Computer Vision: a Geometric Viewpoint*, MIT Press, 1993.

[8] D. Nister, "An efficient solution to the five-point relative pose problem," in *Conference on Computer Vision and Pattern Recognition*, Madison, WI, June 2003.

[9] T. Drummond and R. Cipolla, "Real-time visual tracking of complex structures," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 7, pp. 932–946, july 2002.

[10] 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.

[11] D. G. Lowe, "Robust model-based motion tracking through the integration of search and estimation," *International Journal of Computer Vision*, vol. 8(2), no. 113–122, 1992.

[12] D. DeCarlo and D. Metaxas, "The Integration of Optical Flow and Deformable Models with Applications to Human Face Shape and Motion Estimation," in *Conference on Computer Vision and Pattern Recognition*, 1996, pp. 231–238.

[13] 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.

[14] F. Jurie and M. Dhome, "Hyperplane approximation for template matching," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 7, pp. 996–100, july 2002.

[15] G.D. Hager and P.N. Belhumeur, "Efficient region tracking with parametric models of geometry and illumination," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 10, pp. 1025–1039, 1998.

[16] 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*, 1995, pp. 174–180.
[17] K.W. Chia, A.D. Cheok, and S.J.D. Prince, "Online 6 DOF Augmented Reality Registration from Natural Features," in *International Symposium on Mixed and Augmented Reality*, 2002.
[18] G. Simon, V. Lepetit, and M.-O. Berger, "Computer vision methods for registration : Mixing 3d knowledge and 2d correspondences for accurate image composition," in *International Workshop on Augmented Reality, San francisco, USA*, 1998.
[19] 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.
[20] C.G. Harris and M.J. Stephens, "A combined corner and edge detector," in *Fourth Alvey Vision Conference, Manchester*, 1988.
[21] P.J. Huber, *Robust Statistics*, Wiley, New York, 1981.
[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] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, Cambridge University Press, 2000.
[24] David G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 20, no. 2, pp. 91–110, 2004.
[25] V. Lepetit, L. Vacchetti, D. Thalmann, and P. Fua, "Fully Automated and Stable Registration for Augmented Reality Applications," in *International Symposium on Mixed and Augmented Reality*, Tokyo, Japan, September 2003.