Tu Bao Ho, Douglas Zuckerman, Pierre Kuonen
Akim Demaille, Ralf-Detlef Kutsche (Eds.)

2010 IEEE-RIVF International Conference on Computing and Communication Technologies

Research, Innovation and Vision for the Future

Hanoi, Vietnam

November 1-4, 2010

IEEE-RIVF 2010

Addendum Contributions

2010 IEEE-RIVF International Conference on Computing and Communication Technologies

Research, Innovation and Vision for the Future

Vietnam National University, Hanoi
Vietnam, November 1–4, 2010

Organized by
IEEE Vietnam Section
In cooperation with IEEE Region 10 Asia Pacific

Technically co-sponsored by
IEEE Communications Society
IEEE Computational Intelligence Society
Télécom ParisTech
EPITA

Supported by
RDECOM – International Technology Center – Pacific
Asian Office of Aerospace R&D
IFI Solution

Editors
Tu Bao Ho, Douglas N. Zuckerman, Pierre Kuonen
Akim Demaille, Ralf-Detlef Kutsche

2010 IEEE-RIVF
International Conference on Computing and Communication Technologies
Research, Innovation and Vision for the Future

IEEE Catalog Number: CFP1056A-PRT
ISBN: 978-1-4244-8072-2

Printed copies of this publication are available from:

Curran Associates, Inc
57 Morehouse Lane
Red Hook, NY 12571 USA
Phone: (845) 758-0400
Fax: (845) 758-2633
E-mail: curran@proceedings.com

Produced by IEEE eXpress Conference Publishing
For information on producing a conference proceedings and receiving an estimate, contact conferencepublishing@ieee.org
http://www.ieee.org/conferencepublishing

Organizing Committee

Honorary chairs

Hoang Van Phong – Minister of Science and Technology, Vietnam
John Vig – IEEE 2009 President, USA

General chairs

Patrick Bellot – Telecom ParisTech, Paris, France
Tru Cao – HCMUT, HCMC, Vietnam

International Advisory Committee

Nim Cheung – ASTRI, China
Janina Mazierska – Massey Univ., New Zealand
Jung-Uck Seo – Academy of Science, Korea
Jean-Marc Steyeart – Polytechnique, France
Anders Ynnerman – Linkoping Univ., Sweden

Program chairs

Tu Bao Ho – JAIST, Japan
Douglas N. Zuckerman – Telecordia & IEEE ComSoc, USA
Pierre Kuonen – HES-SO/EIA-FR, Switzerland

Organizing chairs

Luong Chi Mai – VAST, Hanoi, Vietnam
Nguyen Thi Minh Huyen – VNU-HN, Hanoi, Vietnam
Bich-Thuy Dong – VNU-HCMUS, HCMC, Vietnam

Tutorials chairs

Huynh Quyet Thang – HUT, Hanoi, Vietnam
Pham-Hi Duc – ECE, Paris, France

Workshops chairs

Viet-Ha Nguyen – VNU-HN, Hanoi, Vietnam
Richard Nguyen – SPAWAR Systems, San Diego, USA

Publication chairs

Akim Demaille – EPITA/LRDE, Le Kremlin-Bicêtre, France
Ralf D. Kutsche – TU Berlin, Berlin, Germany

Table of contents

A fully automated camera tracking system and its application to augmented reality

Thi Thanh Hai Tran
MICA International Research Center
Ha Noi University of Technology
Ha Noi, Viet Nam
Email: thanh-hai.tran@mica.edu.vn

Eric Marchand
Lagadic team, IRISA Inria,
Campus Beaulieu,
Rennes, France
Email: eric.marchand@irisa.fr

***Abstract*—One of the biggest problems in building an augmented reality system is registration in order to estimate camera pose. This is usually done by manually setting the camera near one of positions that the camera pose has been pre-computed. Then the pose of the camera taking next frames will be computed by using a tracking algorithm. The difficulty of this kind of approach is that the tracker can get lost easily. In that case it needs a human to interfere to get it on. This paper develops a fully automatic system for tracking based on matching keypoints at both steps: initialization and tracking. At initialization, the pose of the camera is reliably estimated by matching the first image against all reference images in database using SIFT-matching algorithm [1]. Then we apply a fast algorithm, proposed in our previous works [9] to match the current image with the previous one to speed up the tracking. In case where the matching is not satisfied, SIFT matching is applied to re-initialize the tracker. The main contribution of this paper is having developed one system of camera tracking, which is fully automatic and works in near real-time, so suitable to real-time applications such as augmented reality that we show the result in this paper.**

I. Introduction

Augmented Reality (AR) is currently a very attractive research and development topic because of its wide application in advertising, navigation devices, military and emergency services, prospecting, art, sightseeing, architecture, entertainment and education, etc. AR applications require the alignment of the real camera and the virtual camera used for virtual scene rendering. To achieve this it is necessary to compute the camera pose with respect to the scene for each image (**camera tracking problem**). The problem of pose estimation is defined as the problem of determining the relative position between the camera and the object (the scene). The relative position includes the position and orientation of the camera w.r.t. the object, assuming that the intrinsic parameters of the camera are a prior known.

Automatic, **fast** and **reliable** pose estimate is a difficult problem. To estimate pose, many methods based on markers have been proposed [6], [5], [4]. The use of markers increases the robustness and reduces computational requirements. However, their use can be very complicated, as they require certain maintenances.

Recently, some authors suggest to substitute markers by images [8], [3], [7]. This takes advantage of the fact that, for pose estimation purpose, having markers is not always possible but several training images may be available. Therefore, estimating the camera pose of a given image can be done by matching this image with each among training images and determining the displacement between two corresponding cameras.

Although image matching is much studied in computer vision field, it is not widely applied in Augmented Reality (AR). The main reason is recognition algorithms in the literature are generally carefully designed to be invariant to viewpoint, illumination changes, occlusion, scale changes, etc. But more they are robust to such changes, more they take times for computing and then less they are suitable for real-time applications.

The work described in this paper is within the context of AR in which the computational time is a critical element. Therefore, our objective is to find a method that obtains a good trade-off between the *computational time* and the *precision* of the camera pose. In addition, we desire that the system can work automatically, without human intervention.

Based on our previous work presented in [9] concerning a fast method for matching keypoints, in this paper, we propose to develop a fully automated camera tracking system, which consists of two phases: initialization and tracking. At initialization, the pose of the camera is reliably estimated by matching the first image against all reference images in database using SIFT-matching algorithm [1]. To speed up the tracking, we are based on the fast keypoint matching algorithm proposed in [9]. In case where the fast matching is not satisfied, SIFT matching is applied to re-initialize the tracker.

The main contribution of this paper is to propose a framework for fully automated tracking in which we use a simple but very efficient keypoint detector and a strategy to shift some of computational burdens into training phase, which provide a very satisfying results. In comparison with the previous works, our (re)-initialisation is much more reliable thanks to SIFT based keypoints matching. In addition, to compute camera pose, we use 3D model information that allows our method works with all types of scenes (2D/3D), not limited to planar scene as in our previous work [9]. This framework is validated in an augmented reality application.

This paper is organized as follows. We first present the architecture of our tracking system (section II). Then, we

explain each component of the system in the sections III, IV, V. Section VI describes how the pose of the camera will be determined and tracked. Section VII describes how the tracking is applied into augmented reality and show experimental results.

II. Framework of camera tracking system

We propose a method for camera tracking using a set of *reference images* as model. Our tracking system consists of 2 main flows that correspond to two phases of tracking system: (re)-initialization and tracking (see figure 1).

At initialization, the first image will be matched with all reference images (of the same scene) in the database and the most similar one will be chosen. Note that the pose of the camera capturing each reference image has been pre-computed. As the first image may be very different from reference images, we use SIFT based matching technique, described in [1].

Once the matching ends, result (set of corresponding points) obtained from matching the candidate image with the most similar one will be used to determine the transformation between two corresponding cameras. The initial pose of camera will be therefore simply inferred.

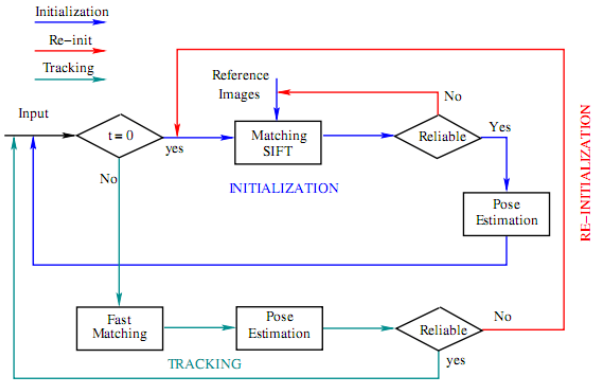

Fig. 1. General tracking system consist of 2 steps: Initialization, tracking and re-initialization if necessary.

After the initialization, between two instants (2 frames) as the camera doesn't change so quick, the current image seems to be quite similar to the previous one. We apply a fast method to detect and match points [9]. If the matching result is not reliable, a SIFT-matching algorithm will be applied to the current image to re-initialize the tracker.

Both matching algorithms consists of 3 steps: keypoint detection, keypoint description, and keypoint matching. They are different at two first steps. In the following, we describe these two first steps of each algorithm.

III. Review of SIFT detector

SIFT consists of 4 steps: (i) scale-space extrema of Laplacian of Gaussian (LoG) extraction; (ii) keypoint localization; (iii) canonical orientation assignment; (iv) keypoint description. First, local extrema of Laplacian in scale-space are extracted. This is efficiently done by constructing a Gaussian pyramid and detecting local extrema of difference of Gaussians (DoG). By this way, keypoints are invariant to scale change. These detected points will be next re-localized to improve precision in localization. Each point is then assigned a canonical orientation (see the original paper of Lowe [1] and section IV of this paper) such that following which the description of the keypoint is invariant to rotation. The description of the keypoints is finally designed by building an array of histograms of gradient orientations. This description is more compact and significantly discriminant than the signal image itself.

We use SIFT to match images (the first image against each reference image) at initial step, which allows us to be able to determine initial pose with a quite different viewpoint at the first time with respect to times taking reference images. We implemented the SIFT detection and descriptor and obtain result of matching which is quite similar to the binary provided by Lowe[1]. The number of matches provided by our algorithm is less than his but it allows to flexibly turn the algorithm on (see figure 2 for example).

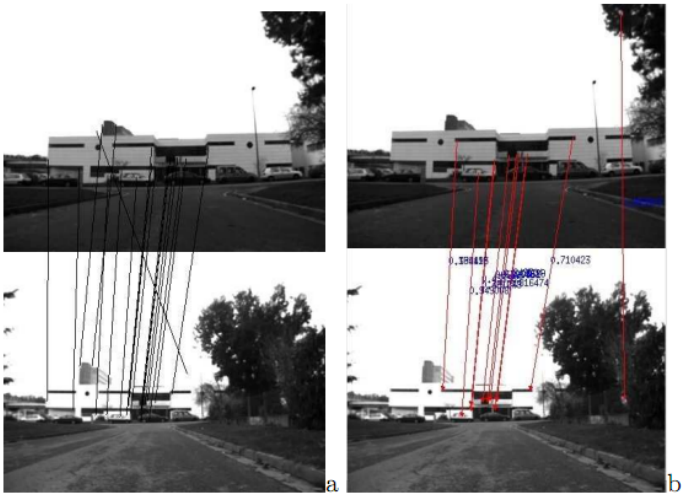

Fig. 2. Obtained matching result (a) by Lowe's binary. (b) by our implementation.

IV. Fast keypoints detection and description

In our previous paper [9], we presented a simple but very fast method for matching keypoints. During tracking, as the camera does not change so quick, consecutive images seem to be not too different each other. We propose to use this method to obtain a good trade-off between the computational time and the precision of image matching. The method consists of 4 steps: keypoints detection, canonical orientation assignment, computation of eigenspace and local region description.

A. Keypoint detection

A variation of the Rosten's method [11] has been proposed in [9]. We detect keypoints at which the signal image changes significantly. A point is not considered as a keypoint if its

energy is smaller than those of two opposite points in the 16-circle. The criterion is much more simpler than the one proposed in [11] but gives the similar result of detection. Once edge and region responses are eliminated, we reject remaining multiple adjacent responses by keeping only points which have extremal value of Laplacian (figure 3).

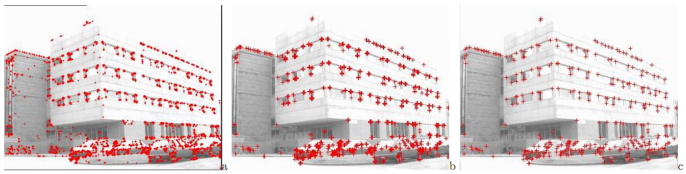

Fig. 3. Keypoint detection: (a) Points detected by Rosten detector; (b) Points provided by our detector. We can see that our criterion is more simpler but gives similar result of detection; (c) Remaining points after verifying Laplacian criterion. We notice that lots of multiple responses have been rejected

B. Canonical Orientation Assignment

By assigning a consistent orientation to each keypoint based on local image properties, the keypoint descriptor can be represented relative to this orientation and therefore achieve invariance to image orientation.

Canonical orientation of a keypoint is defined as an orientation of gradient that have majority of points surrounding this keypoint. To determine canonical orientation, a histogram of gradient orientation is computed for all points within a region of size 7×7 centered at the keypoint. The most significant peak in the histogram corresponds to the canonical orientation of local gradient.

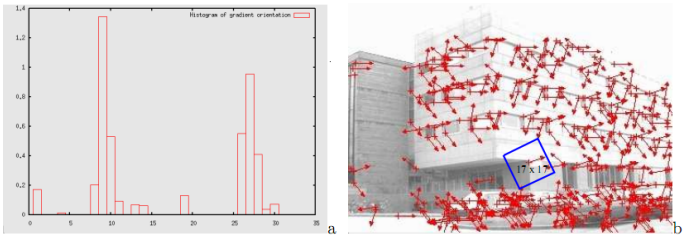

Fig. 4. (a) Example of a histogram of gradient orientation computed at the local region centered at the keypoint. The most significant peak in the histogram corresponds to the canonical orientation of the keypoint. (b) Keypoints detected from a building image. Each keypoint is assigned at least one canonical orientation. The descriptor is built using the local patch (blue square) around the keypoint, in the canonical orientation.

The assignment of orientation in this way costs lightly more expensive than the one proposed in [8] where an orientation which maximizes gradient magnitude is computed. However, the obtained orientations are more stable to noise.

C. Computation of eigenspace

Considering a set of oriented keypoints, the next step is to compute a descriptor for the local region around a keypoint that is highly distinctive yet is as invariant as possible to variations, such as change in illumination or 3D viewpoint. Obviously, we can extract an intensity region around each keypoint and match these using a correlation measure. However the intensity correlation is too sensitive to noise and the search in such high dimensional space is very time consuming. We use then gradient magnitude computed from normalized image which allows an invariance to illumination changes. Furthermore, to reduce high dimensions, Principal Component Analysis (PCA) technique is considered.

D. Local region description

Once an eigenspace is built, we have a new basis $(\mathbf{v_1}, \mathbf{v_2}, ..., \mathbf{v_K})$ to describe patches. Each patch now is represented by a K-elements vector $\widehat{\mathbf{\Omega}}$ which is considerably smaller than the original vector $\mathbf{\Gamma}$ (eg. 20 against 39×39=1521 with patch size N = 41). Obviously, this representation is more compact than the original one and thus allows a faster search using nearest neighbours algorithm. In addition, it tolerates intra-class variations and recognizes better the extra-class variation.

V. Matching keypoints

Matching keypoints detected from 2 images is an essential step before pose estimation. The keypoints matching algorithm presented in this section will be applied to both phases: initialization and tracking. To match points in two images, keypoints are detected (section III, section IV-A) and described (section III) or projected (section IV-D) onto pre-built eigenspace (section IV-C). Basically, matching keypoints now consists in searching for the nearest neighbor.

To efficiently match two sets of points in high dimension, we use the technique for searching approximate nearest neighbour proposed by Mount [16]. The idea is to organize feature point set into a kd-tree or BBD[1]-tree structure such that an approximate nearest neighbour of a query can be computed in $O(c_{d,\epsilon}log(n))$ time, where d is the number of dimensions, n is the number of points in the set.

Computing the approximate nearest neighbours allows to achieve significantly faster running times although it can undergo some matching errors. We overcome this error by using second-closest neighbour criterion, as proposed in [1].

Apart from using of second-closest criterion, we add a more robust criterion to reject outliers matching. Specifically, once keypoints from two images have been matched, a robust estimation of the multi-view geometry that links the two images is computed using RANSAC [13].

VI. Robust estimation of camera pose

Considering our augmented reality issue the goal is to compute the pose of the camera w.r.t the world frame in order to insert correctly virtual objects in the real scene observed by the camera.

We assume that the pose ${}^{r_k}M_W$, $k = 1, M$ for each of the M reference image and a 3D model of the scene are known. Our goal is to compute ${}^{t}M_W$ for the camera of each frame.

[1]Balanced Box Decomposition

We propose one solution based on a statistically robust non-linear minimization algorithm [18]. Pose estimation can be decomposed in an off-line and runtime processes.

At off-line process, from each reference $I_{r_k}, k = 1, M$ a set of N keypoints have been extracted $(x_i, y_i), i = 1, ...N$. Since pose and scene model are known, it is possible to compute for keypoint its depth expressed in the reference camera frame. This can be done by computing the intersection of a line that pass through the camera optical axes and point (x_i, y_i) and the model of the scene leading to a measure of ${}^{r_k}Z$. Perspective equations allow then to compute ${}^{r_k}X_i = x_i^{r_k} Z_i$ and ${}^{r_k}Y_i = y_i^{r_k} Z_i$. Let us note that only keypoints for which the ray intersect the model are maintained in the database. Knowing, the 3D point coordinates ${}^{r_k}\mathbf{X} = ({}^{r_k}X_i, {}^{r_k}Y_i, {}^{r_k}Z_i)$ in the reference camera frame, it is the possible to compute their coordinate in the world frame ${}^{W}\mathbf{X} = {}^{r_k}M_W{}^{-1}{}^{r_k}\mathbf{X}$.

At run-time process, in initialization phase, let us consider that current image I_t has been matched with one of the M reference images (named I_r). Keypoint $\mathbf{x}_i = (x_i, y_i)$ has been extracted from the current image and matched with a keypoint of reference image I_r leading to a 2D / 3D matching between $\mathbf{x}_i$ and ${}^{W}\mathbf{X_i}$ since the world coordinates of each keypoint in the reference image have been computed during the off-line process.

In the current version of our system we choose a non-linear minimization method extended to consider a robust estimation process based on M-estimators. The goal of this process is then to minimize, for the parameters ${}^{t}M_W$ (camera pose parameters), the error Δ between the set of observed data s^* and the data s of the same features computed by forward-projection according to the current pose:

$$\Delta = \sum_{i=1}^{N} (\mathbf{s}_i - \mathbf{s}_i^*)))^2 = \sum_{i=1}^{N} (pr_\xi({}^{t}M_W), {}^{W}\mathbf{X}_i) - \mathbf{s}_i^*)^2 \quad (1)$$

where N is the number of matching keypoints between two images. $pr_\xi({}^{t}M_W)$ is the projection model according to the intrinsic parameters ξ and camera pose ${}^{t}M_W$. It is supposed that ξ is available, to minimize this error, we proposed to use control law presented in [18]. At the convergence of the algorithm, we obtain the current pose of camera. The figure 5 gives an example of current pose computed from this method.

During tracking, as the camera does not change so quick, we can match keypoints detected from two consecutive images and the displacement ${}^{t}M_{t-1}$ between the camera from the time $t-1$ and t will be determined following the same principle above. Once the camera displacement is determined, the computing of current pose of camera ${}^{t}M_W$ is straightforward:

$${}^{t}M_W = {}^{t}M_{t-1}{}^{t-1}M_W \quad (2)$$

VII. Experimental results

In this section we first presents results directly related to the matching issue (section 4 and 5). We then consider the full system with two experiments: one in outdoor urban environment and one for a "Magicbook" [12] like application.

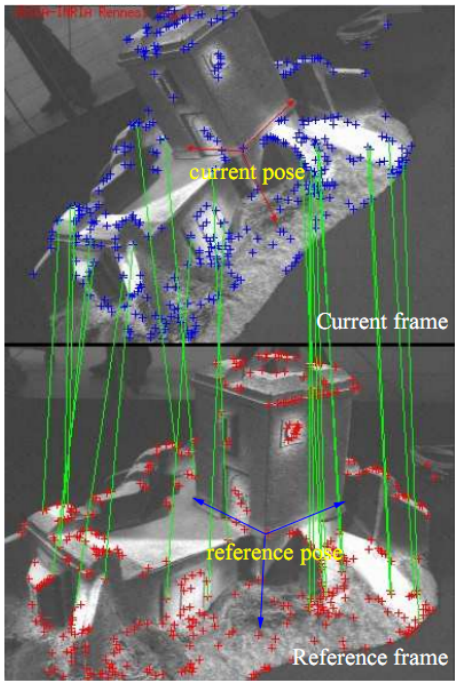

Fig. 5. Pose of camera corresponding to the current frame is computed based on the control law

Let us note that for all the sequences we used the same eigenspace learnt using a large database.

A. Matching results

In this section we will focus our evaluation on computational time as our objective indicated at the beginning of this paper aims at real-time application. To build eigenspace, we use 10 images of different nature. Experiments show that the set of reference images do not influence much the matching result. Some parameters are experimentally chosen to obtain a good trade-off between the precision of matching and the computational time: patch size $N = 17$, number of the largest eigenvalues used to build eigenspace $K = 20$, threshold for second-closest ratio $\epsilon_r = 0.6$, pixel precision for RANSAC $\epsilon_p = 0.3$.

In our previous works, we noticed that the algorithm is very robust to occlusion and illumination change. In both cases, it provides reliable and enough matches for pose estimation. Table I gives information about computational time at each step of matching algorithm. When two images are not too different, the pose estimate works at 18Hz (384x288 image). When these images are quite similar (consecutive frames in a sequence), the speed can be improved up to 25Hz.

B. Application to outdoor augmented reality

To demonstrate the performance of our tracking system, we apply it to an AR application in real 3D urban environment. University Campus in Rennes is used as a test environment. For each building in the campus, 3D model is available. Coordinates of each corner in the model are expressed in a world coordinate frame. Videos are taken by a camera with a

Operation	Times (ms)
Interest Point Extraction	8ms
Interest Point Characterization	5ms
ANN Matching	11ms
RANSAC based outliers rejection	1ms

TABLE I
COMPUTATION TIME ON A CORE 2 DUO, 3GHZ, FOR 400 KEYPOINTS DETECTED FROM CURRENT IMAGE, MATCHED AGAINST 350 POINTS OF KEY-FRAME.

384x288 resolution. The lighting condition at the test moment is very different from at the training one. The number of keypoints in almost frames varies from 200 to 400. With these parameters, the system performs at about 14Hz on a 2.6GHz P4.

Unlike to AR system recently presented in [17] in which the tracking system requires the user to start manually from well-known position in the environment, our system initializes automatically by matching the first images against all reference images.

During tracking, when the number of matches is not enough (at least 4 in theory, 8 in practice) to compute the deplacement of the camera or when the matching is not reliable (eg. the residual is bigger than a threshold), we consider the tracker failed and restart it (meaning that the current image is matched against all the reference image of the database).

Reference images (see Figure 6) are taken by a handheld camera prior to the test sequence acquisition. Those camera poses are pre-computed. For each reference image, we pre-computed also keypoints and descriptors. So at run-time, keypoint detection and description will be done only for the candidate image ; this allows to save half time of computation. All parameters used in matching algorithm remain the same as in section VII-A.

Figures 6 show some results of matching and their use to determine pose of the camera. Current images are on the first row, reference ones are on the second row. Each match is presented by a line connecting them from the current image to the reference one. Each line color represents a plan index on that lie couple of keypoints. On the figures, we overlaid 3D models of buildings present in the scene.

With the use of information about 3D models, keypoints are correctly matched. A keypoint on one plane matches only with a keypoint of the same physical point on this plane. The matching algorithm starts to find corresponding keypoints in current image for all keypoints in reference image lying on the biggest plane. Then it finds corresponding keypoints among remaining ones in current image for all keypoints in reference image lying on the second biggest plane and so on.

Matching accuracy is evaluated regarding how correctly are 3D models of buildings overlaid on theirs images. We note that in this experiment, we rotate the camera a lot. Many times, the first or the second building is completly occluded but the tracker still works thanks to the possibility of system to flexibility and automatically switch to a suitable one between two matching algorithms.

C. "Magicbook" like experiment

The goal of this experiment is to match each new page of the book with the correct reference page in the database (all the pages have been stored) and then to compute the pose of the book. As can be seen on the images of the match results (see Figure 7) the current is always correctly matched even if the image is partly occluded when the page are turned. In that case the displacement between the reference and current image is computed as reported in section V.

VIII. CONCLUSION

The works presented in this paper follows our previous works concerning only keypoints matching algorithm. This paper presented a fully automated camera tracking system, which uses like SIFT matching algorithm to initialize or re-initialize the tracker and a fast matching algorithm to match consecutive images [9]. This system overcomes a common drawback of almost state of the art tracking techniques: automatic initialize and re-initialize whenever the tracker get lost. Without human intervention, this system works in near realtime and provides satisfying result for pose estimation.

The implementation of the tracking system is plugable. We can plug any other keypoints detector or descriptor in both initialization and tracking steps. By this way, it provides possibility to improve each component of the system. In addition, this system is not designed for a specialized application. It can be applied into different realtime applications as surveillance or visual servoing.

However, there exists some problems to be improved:

- First, we used SIFT technique to match images at initial phase. This technique is shown to be robust to illumination or scale changes. However, in our experiment, when illumination changes alot, descriptors of a same physical point do not remain the same. There are also some confusing matches which need to be rejected by adjusting geometrical verification. We are thinking about using not strictly optical information but also others sources (eg. inertial sensor) to improve pose estimatation accuracy.
- Second, total residual computed after matching could be a measure to detect matching failure. However, it is not always a reliable measure. Actually, the tracker is re-initilized whenever the number of matches is not enough to compute camera pose or it is re-initialized after certain number of frames to avoid drifting error. To optimize the tracking time, the problem of detecting matching failure needs to be considered.

REFERENCES

[1] David G. Lowe, *Distinctive Image Features from Scale-Invariant Keypoints*, International Journal of Computer Vision, Vol. 60, pp. 91–110, 2004.

[2] A. I. Comport, E. Marchand, M. Pressigout, and F. Chaumette, *Real-time markerless tracking for augmented reality: The virtual visual servoing framework*, IEEE Transactions on Visualization and Computer Graphics, 12(4), 2006.

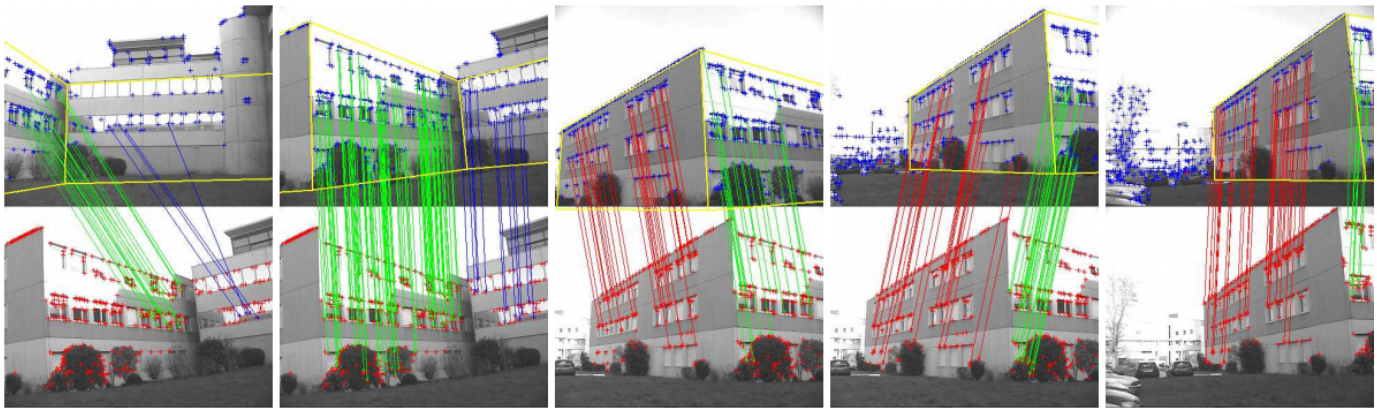

Fig. 6. Results of the matching using the approximate nearest neighbor approach. One the first row are current images. One the second row are reference images. Lines connecting keypoints between two images represent matches. Color of line represents which plan the keypoints belong to. Once the pose of the current image is computed, the 3D model of the building (yellow lines) are projected correctly on the image.

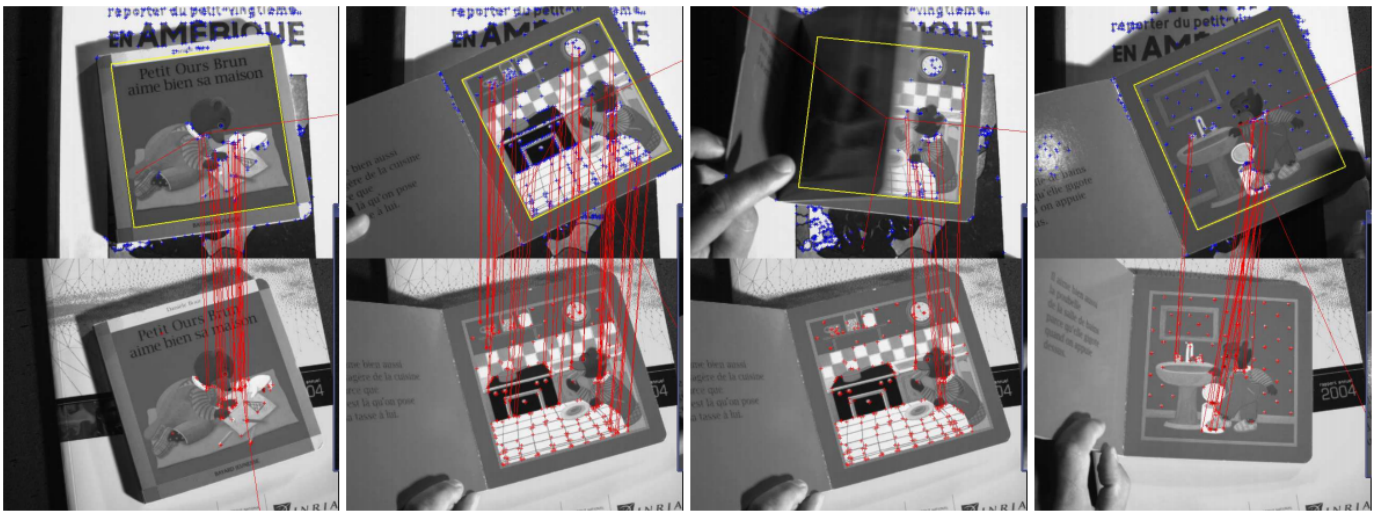

Fig. 7. A "Magicbook" like experiment. A database with all the page of this kid book have been build, and each new image is match against these reference image. When a closest image is found (see the evolution of the reference image on the last row) the pose of the book is computed and its model added to the image

[3] L. Vacchetti, V. Lepetit, and P. Fua, *Stable real-time 3d tracking using online and offline information*. IEEE Transactions on Pattern Analysis and Machine Intelligence, 26(10):1391–, 2004.

[4] R. Quiros M. Rubio, E. Pulido, J. Heurta, and E. Camahort, *Wide area marker based tracking*. 2005.

[5] H. Park and J. II. Park, *Invisible marker tracking for ar*. In IEEE/ACM Int. Sym. on Mixed and Augmented Reality (ISMAR), 2004.

[6] L. Davis, F. G. Hamza-Lup, and J. P. Rolland, *A method for designing marker-based tracking probes*. In ISMAR, pages 120–129, 2004.

[7] V. Lepetit and P. Fua, *Keypoint recognition using randomized trees*. PAMI, Sept 2006.

[8] V. Lepetit, J. Pilet, and P. Fua, *Point matching as a classification problem for fast and robust object pose estimation*. In CVPR, volume 2, pages 244–250, 2004.

[9] H. Tran and E. Marchand, *A realtime keypoints matching: application for visual servoing* In IEEE. Int. Conf. Automation and Robotic, ICRA, 2007

[10] P. Indyk, *Nearest Neighbors in high dimensional spaces*. Handbook of Discrete and Computational Geometry, CRC Press, 2004.

[11] E. Rosten and T. Drummond, *Machine learning for high-speed corner detection*. In ECCV, 2006.

[12] M. Billinghurst, H. Kato and I. Poupyrev, *The MagicBook — Moving Seamlessly between Reality and Virtuality*, IEEE Computer Graphics and Applications, Vol. 21, pp. 6–8, 2001

[13] N. Fischler and R.C. Bolles, *Random sample consensus: A paradigm for model fitting with application to image analysis and automated cartography*, Communication of the ACM, 24(6):381–395, June 1981.

[14] E. Malis and F. Chaumette, *2 1/2 d visual servoing with respect to unknown objects through a new estimation scheme of camera displacement*, Int. Journal of Computer Vision, 37(1):79–97, June 2000.

[15] Y. Ke and R. Sukthankar, *Pca-sift: A more distinctive representation for local image descriptor*, In CVPR, 2004.

[16] D.M. Mount, *ANN programming Manual*, Dpt. of Computer Science and Inst. of Advanced Computer Studies, Univ. of Maryland, College Park, Maryland, 2005.

[17] G. Reitmayr and Tom W. Drummond, *Going out: Robust Tracking for Outdoor Augmented Reality*, Proc. ISMAR, pp.109–118, 2006

[18] A.I. Comport, E. Marchand, M. Pressigout, F. Chaumette, *Real-time markerless tracking for augmented reality: the virtual visual servoing framework*, IEEE Trans. on Visualization and Computer Graphics, 12(4):615-628, Juillet 2006.