2007 IEEE International Conference on
Robotics and Automation
Roma, Italy, 10-14 April 2007

# Real-time keypoints matching: application to visual servoing

Thi Thanh Hai Tran, Eric Marchand

*Abstract*— **Many computer vision problems such as recognition, image retrieval, and tracking require matching two images. Currently, ones try to find as reliable as possible matching techniques with a very little constraint of computational time. In this paper, we are interested in applying image matching technique into robotic problems such as visual servoing in which the computational time is a critical element. We propose in this paper a real time keypoint based matching method. The novelties of this method include a fast corner detector, a compact corner descriptor based on Principal Component Analysis (PCA) technique and an efficient matching with help of Approximate Nearest Neighbor (ANN) technique. We show that the method gives a very satisfying result on accuracy as well as the computational time. The matching algorithm is applied to control a robot in a visual servoing application. It works at 10-14Hz and is well robust to variations in 3D viewpoint and illumination.**

## I. Introduction

When dealing with vision-based robot control, real-time tracking is a fundamental issue. This problem received much interest in the literature and various approach can be considered: tracking based on point of interest [25], 2D features [11], [3], 2D templates [10], [2], 3D model [7], [4], [6], etc. A presentation of real-time tracking algorithms for visual servoing purposes is given in [19]. Although usually efficient, these methods failed to address two important issues: initialization and failure recovery (which in fact is a re-initialization problem). Addressing all these issues (initialization, tracking, failure recovery) within the same approach can be achieved considering tracking as a recognition or image matching issue.

Image matching consists to automatically establish the correspondence between primitives extracted from two images. First solutions for image matching have been suggested already in the late fifties [12]. Since then a steady increase in the interest for image matching has occurred. But matching still remains one of the most challenging tasks in the computer vision field. The reason comes not only from the high implicit information contained in the image to be discriminally represented, but also from noise during image acquisition, changes of camera viewpoints, illumination, occlusion, etc. Recently, some keypoints based matching methods obtained impressive results in object recognition/classification [21], [17], [22]. The high precision and the robustness of these methods to some transformations such as scale change, illumination change, rotation are due to a very careful design of keypoint detector as well as keypoint descriptor. Consequently, they are usually time-consuming.

Considering robot control, not only the accuracy is required but also the computation cost. In order to address these issues, a real-time, robust image matching method is proposed. The main idea is to explore advantages from existing matching techniques, adapt them so that a good trade-off between the computational time and the precision can be achieved. In addition, we shift some of computational burdens into offline training. This allows our method meets well requirements of visual servoing task.

The contributions of this paper are found in each step of the matching algorithm:

- **Keypoint detection:** We propose a criterion which eliminates quickly edge points or points in uniform regions from corner points in an image. This is done from the full-resolution image.
- **Keypoint description**: Each keypoint is described by a compact descriptor (e.g. 20-elements vector). This speeds up significantly the matching. The idea is to use PCA technique to reduce dimensionality of feature space. Eigenspace is pre-built in training phase, so does not take time in running phase.
- **Keypoint matching:** Using ANN technique for point matching is efficient in computational time as well as precision. For each reference image, keypoints, descriptors, and corresponding $kd$-tree are precomputed. At running time, we detect keypoints and describe them from only current image. This reduces a half of time against some state of the art algorithms.

Finally, to validate our approach, the matching/tracking algorithm is used to performed positioning task using visual servoing [8].

The organization of the paper is as follows: Section II explains the keypoint detection and their description. Section III describes the matching algorithm. In section IV, experimental results of matching on servoing task will be analyzed. We conclude and give some ideas to improve the actual method in order to obtain higher matching performance in section V.

## II. Keypoints detection and representation

### A. Keypoints detection

Keypoint detection is the first step in a process of points matching. By definition, keypoints are points which contain more information than other points in the image. They allow a more compact representation of the image and help to recognize better the scene than all rough pixels.

Fast keypoint detection algorithm have been recently proposed [24], [16]. Following this way, in our work, a point is

Authors are with INRIA, IRISA, Lagadic, F-35000 Rennes, France, e-mail htran@irisa.fr, marchand@irisa.fr. Authors wish to thank France Telecom R&D for its financial support.

1-4244-0602-1/07/$20.00 ©2007 IEEE.

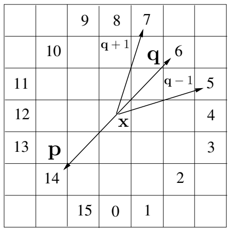

Fig. 1. Tests for a keypoint. Two opposite neighbors p and q of candidate point $\mathbf{x}$ are on the 16-circle.

identified as keypoint where the image signal is significantly different from those of two opposite neighbors. Formally, *Given an image* $\mathbf{I}$*. A point* $\mathbf{x}$ *is not considered as a keypoint if there exists two opposite points* $\mathbf{p}$ *and* $\mathbf{q}$ *such that:*

$$\begin{cases} |\mathbf{I}(\mathbf{x}) - \mathbf{I}(\mathbf{p})| \le \epsilon_d \\ |\mathbf{I}(\mathbf{x}) - \mathbf{I}(\mathbf{q})| \le \epsilon_d \end{cases} \qquad (1)$$

*where* $\epsilon_d$ *is a sufficient small threshold.* $\mathbf{p}$ and $\mathbf{q}$ are two points on a circle of 16 pixels around the candidate keypoint $\mathbf{x}$, as illustrated in Figure 1.

The criterion (1) will eliminate quickly edge and region responses. To avoid detecting keypoints on skewed edges, we do the test also on two skewed opposite points $\mathbf{q}+1$, $\mathbf{q}-1$. The test is started from one point on the circle and stopped when it returns true. In this case $\mathbf{x}$ is not a keypoint.

Once edge and region responses are eliminated, we reject remaining multiple adjacent responses by keeping only points which have extremal value of Laplacian. The Laplacian is approximated in a very simple manner:

$$\mathbf{L}(\mathbf{x}) = \sum_{\forall(\mathbf{p},\mathbf{q})} (\mathbf{I}(\mathbf{p}) + \mathbf{I}(\mathbf{q}) - \mathbf{I}(\mathbf{x})) \qquad (2)$$

where $\mathbf{p}, \mathbf{q}$ are two *right* opposite points on the 16-circle associated to the considered point $\mathbf{x}$.

Obviously, our detection will be realized more faster than a multi-scales or scale-space approach because only the original image is considered. This makes the method not invariant to scale. However, as we can see in the following, in the context of visual servoing where scale does not change strongly, detected keypoints from image still remain quite repeatable.

## B. Invariance to image orientation

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.

We propose to use a principle similar to the one presented in [17] for orientation assignment. Nevertheless, since keypoints are not detected in scale-space, only a histogram of gradient orientation is computed for all points within a region of size 7x7 centered at the keypoint. The orientation histogram has 36 bins covering the 360 degree range of orientation. Each sample added to the histogram is weighted by its gradient magnitude and by a Gaussian-weighted circular window with $\sigma = 3$ (radius of 16-circle). The most significant peak in the histogram corresponds to the canonical orientation of local gradient.

The assignment of orientation in this way costs lightly more expensive than the one proposed in [16] where an orientation which maximizes gradient magnitude is computed. However, the obtained orientations are more stable to noise.

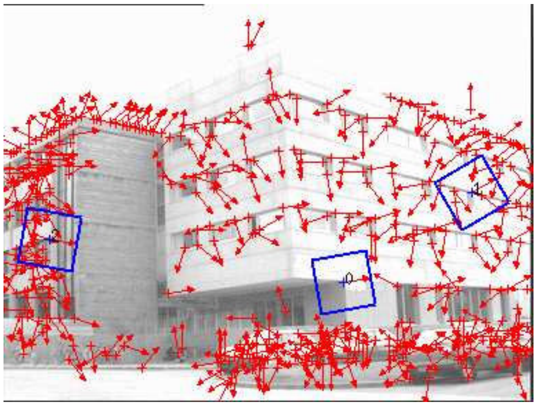

Fig. 2. Keypoints points detected from a building image. Each keypoint is assigned one canonical orientation. The descriptor is built using the local patch (blue squares) around the keypoint, in the canonical orientation.

Figure 2 shows keypoints detected from an image of buildings. Each keypoint is assigned an orientation, depicted in the Figure by a red arrow. We can see that almost keypoints represent corners of buildings in the scene. They are quite similar but the descriptors built in respective canonical orientations are discriminant.

## 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 propose to use 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.

PCA is a standard technique which enables to linearly-project high-dimensional samples onto a low-dimensional feature space, that is called eigenspace. Such method has been shown to be very well-suited to representing keypoint patches [15].

The building of the eigenspace consists in following steps:

- Extract patches $P_1$, $P_2$, ..., $P_M$ (training patches) in the canonical orientation at each keypoint detected from training images. Each patch is centered and of the same size $N \times N$ (with $N = 17$). If the number of patches is not large enough, we create more patches by using synthesizing technique [16].

- Represent each patch $P_i$ by a vector that is the gradient vector $\mathbf{\Gamma}_i$ of $(N-2)^2$ elements (points at boundary are not taken into account) (see Figure 3). Gradient magnitude is determined by:

$$\mathbf{G}(\mathbf{x}) = \nabla \mathbf{I}_x^2(\mathbf{x}) + \nabla \mathbf{I}_y^2(\mathbf{x}) \tag{3}$$

where $\nabla \mathbf{I}_x(\mathbf{x})$ (resp. $\nabla \mathbf{I}_y(\mathbf{x})$) is the image gradient along the $x$ axe (resp. $y$ axe).

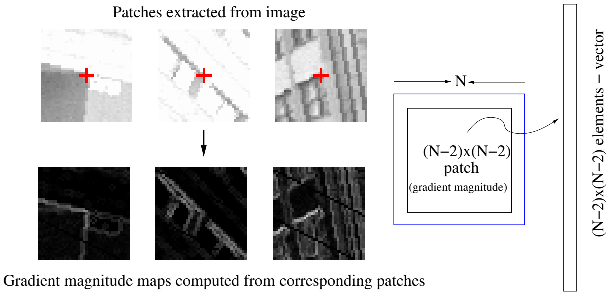
Patches extracted from image
N
(N−2)x(N−2)
patch
(gradient magnitude)
(N−2)x(N−2) elements − vector
Gradient magnitude maps computed from corresponding patches

Fig. 3. Patches are extracted and each is described by a vector of gradient magnitudes.

- Normalize the gradient vector $\forall i = 1 \ldots M$:

$$\mathbf{\Omega}_i = \mathbf{\Gamma}_i - \mathbf{\Psi} \text{ with } \mathbf{\Psi} = \frac{1}{M}\sum_{i=j}^{M} \mathbf{\Gamma}_j$$

- Compute the covariance matrix $\mathbf{C}$:

$$\mathbf{C} = \frac{1}{M}\sum_{n=1}^{M} \mathbf{\Omega}_n \mathbf{\Omega}_n^\top = \mathbf{A}\mathbf{A}^\top \tag{4}$$

$\mathbf{C}$ is a $(N-2)^2 \times (N-2)^2$ matrix. $\mathbf{A} = [\mathbf{\Omega_1} \ldots \mathbf{\Omega}_M]$ is a $(N-2)^2 \times M$ matrix.
- Compute the eigenvalues $e_i$ of $\mathbf{C}$ and the corresponding eigenvectors $\mathbf{v}_i$ by applying Singular Value Decomposition technique (SVD) on the covariance matrix $\mathbf{C}$.
- Finally, keep only $K$ eigenvectors corresponding to $K$ largest eigenvalues. These vectors create a new basis of eigenspace of $K$ dimensions. $K = 20$ is chosen experimentally which is small enough to allow a good discriminant descriptor of keypoints.

### 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. Computing keypoint descriptor in the eigenspace follows these steps:

- *Step 1:* subtract gradient magnitude vector $\mathbf{\Gamma}$ by the average vector $\mathbf{\Psi}$ : $\mathbf{\Omega} = \mathbf{\Gamma} - \mathbf{\Psi}$
- *Step 2:* project $\mathbf{\Omega}$ onto eigenspace:

$$\widehat{\mathbf{\Omega}} = \sum_{i=1}^{K} w_i \mathbf{v}_i \quad \text{with } w_i = \mathbf{v}_i^\top \mathbf{\Omega}$$

- *Step 3:* represent $\mathbf{\Omega}$ as a $K$-elements vector: $\widehat{\mathbf{\Omega}} = (w_1 \ldots w_K)^\top$

Each patch is represented as 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 neighbors algorithm. In addition, it tolerates intra-class variations and recognizes better the extra-class variation.

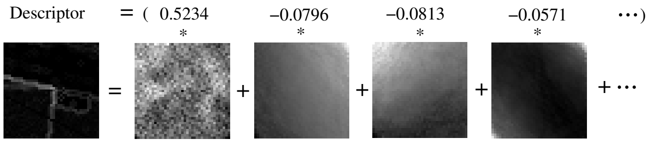

Fig. 4. Vector of gradient magnitudes is a linear combination of eigenvectors.

Figure 4 illustrates how a patch is described in eigenspace. At left, we have an input is a gradient magnitude map. At right, we show 4 eigenpatches corresponding to 4 first eigenvectors in the basis. The input vector is a linear combination of these eigenvectors. The multiplicative coefficients $w_i$ form a vector descriptor for the patch in eigenspace.

## III. Keypoints Matching

To match points in two images, keypoints are detected (section II-A) and projected (section II-D) onto pre-built eigenspace (section II-C). Basically, matching keypoints now consists in searching for the nearest neighbor. Nevertheless, for efficiency issue specific algorithms have to be considered.

### A. Approximate nearest neighbor based point matching

In the nearest neighbor problem a set of data points coded as by descriptor in $K$-dimensional space are given. These points are preprocessed into an appropriate structure, so that given any query point $\widehat{\mathbf{\Omega}}$, the nearest points to $\widehat{\mathbf{\Omega}}$ can be reported as quickly as possible. Although nearest neighbor searching can be performed efficiently in low-dimension spaces, search time grows exponentially as a function of dimension [14].

To efficiently match two sets of points, we use approximate nearest neighbor technique proposed by Mount [1], [23]. The idea is to organize feature points into a $kd$-tree structure and compute the nearest neighbors approximately. The similarity between two feature points in eigenspace is measured by:

$$||\widehat{\mathbf{\Omega}} - \widehat{\mathbf{\Omega}}^k|| = \sum_{i=1}^{K} \frac{1}{e_i}(w_i - w_i^k)^2 \tag{5}$$

Computing the approximate nearest neighbors allows to achieve significantly faster running times although it can undergo some matching errors. We overcome this error by using second-closest neighbor criterion, as proposed in [17]. Concretely, a match is considered as a correct match when it has the closest neighbor significantly closer than the closest incorrect match. All matches in which the second-closest ratio is greater than some threshold $\epsilon_r$ will be rejected.

### B. Outliers rejection using RANSAC

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 [9].

More precisely, an homography ${}^a\mathbf{H}_b$ links the projection of matched point ${}^a\mathbf{x}_i$ and ${}^b\mathbf{x}_i$: $\forall i, {}^a\mathbf{x}_i = {}^a\mathbf{H}_b\, {}^b\mathbf{x}_i$ (for planar scenes) where ${}^a\mathbf{x}_i$ and ${}^b\mathbf{x}_i$ are points 2D homogeneous coordinates and ${}^a\mathbf{H}_b$ is a $3 \times 3$ matrix. At each iteration of RANSAC, the homography ${}^a\mathbf{H}_b$ is estimated using the method presented in [18] which requires at least 4 couples of points for planar scene or 8 for non-planar scene. Although the computed homography is not used in the current version of our system, this method allows to reject efficiently the remaining outliers.

## IV. Experimental results

### A. Context of experiments

The aim of the experiments is to validate if the proposed matching algorithm is sufficiently fast and reliable for vision-based control applications. The considered task is to control the end-effector of a robot to achieve a positioning task. This is a classical problem in robotic (eg. grasping task). It consists in 2 steps. In the first learning step, the camera is moved to its desired position. The desired image $\mathbf{I_d}$ of the target corresponding to this position is acquired. We then detect keypoints from this image, project them onto the pre-built eigenspace and organize them into $kd$-tree structure. As the eigenspace is pre-built at learning phase, this step takes only around tens millisecond, depending on if the object is complex or not. Note that we do not need to build eigenspace in this phase because the nature eigenspace does not influence strongly the matching result (as well indicated in [15]).

After some unknown displacements of the camera or the object, the robot is controlled so that that the current image features reach their desired position in the image. This is done by detecting and describing keypoints from the current image then applying matching algorithm to search correspondences. The error of position of matched points are used to command 6-d.o.f of the robot. The positioning task ends when the error is smaller than a given threshold. At convergence, the camera is located at the same position wrt to the object in learning phase.

These processes have been tested at IRISA-INRIA Rennes on a gantry robot and have been implemented using the ViSP package [20].

### B. Visual servoing

We consider the generic positioning task. The goal of visual servoing is essentially to minimize the error $\Delta = \mathbf{s} - \mathbf{s}^*$ between a set of visual features $\mathbf{s}$, that depends of the actual camera location, and a set of desired visual features $\mathbf{s}^*$. The control law that performs $\Delta$ minimization is usually handled using a least square approach [13].

In our case, $\mathbf{s}^*$ describes the set of points extracted from the desired image $\mathbf{I_d}$ using the method presented in Section II-A. Assuming that $n$ points have been detected in $\mathbf{I_d}$, we then have $\mathbf{s}^* = (x_1^*, y_1^*, \ldots, x_n^*, y_n^*)^\top$. $\mathbf{s}$ contains information about the matched points in the current image $\mathbf{I}$. Obviously the number $m$ of matched point is such that $m \leq n$. $\mathbf{s}$ is then defined as $\mathbf{s}^* = (x_1, y_1, \ldots, x_n, y_n)^\top$ with $(x_i, y_i) = (0, 0)$ if point $(x_i^*, y_i^*)$ has not been matched. Since non matched points must not be considered in the control law we also defined a diagonal $n \times n$ matrix $\mathbf{D}$. $\mathbf{D} = diag(\ldots, \omega_i, \ldots)$ with $\omega_i = 0$ if point $(x_i^*, y_i^*)$ has not been matched, and 1 otherwise. The control law is given by [5]:

$$\mathbf{v} = -\lambda(\mathbf{DL})^{+}\mathbf{D}(\mathbf{s} - \mathbf{s}^*). \qquad (6)$$

where $\mathbf{v}$ is the computed camera velocity and $\mathbf{L_s}$ is the interaction matrix related to the point [8].

### C. Results

Two experiments are reported, the former (named "marvels experiment") consider a positioning task wrt to a planar scene with textured posters, while the latter (named "castle experiment") consider a positioning task wrt a complex 3D object. Figures 5 and 7 shows the reference and initial image of the positioning tasks along with match points. As mentioned, one of the interest of such tracking by matching approach is that initialization (ie, matching between $\mathbf{s}$ and $\mathbf{s}^*$) which is usually a tough problem in visual servoing is, here, a trivial issue. Partial or total occlusion is also easily handled (only the number of matched points decreases).

Figure 6 (resp. Figure 8) show the camera velocity (Figure 6ab and 8ab) and the norm of the error $\|\mathbf{s} - \mathbf{s}^*\|$ which decreases as expected. Let us note that this is rough results. The extracted position of the matched points are not filtered which may introduce noise in vector $\mathbf{s}$ and then in the computed camera velocity. Kalman filter may be easily considered to cope with this issue.

Table I gives some informations about the computational time at each operation in the matching algorithm. In general the matching works at 10Hz on a Pentium IV, 2.6GHz. When these images are quite similar (robot near to desired position), the speed increases to 14Hz.

| Operation | Times (ms) |
|---|---|
| Keypoints Extraction | 10ms |
| Keypoints Characterization | 30ms |
| ANN Matching | 20ms |
| RANSAC based outliers rejection | 30ms |

TABLE I

Computational time for image matching.

The last experiment (Figure 9) demonstrates the good behavior of our system when partial or complete occlusions occur (see also the video). When the occlusion is complete, no match are found and the robot stops. When occlusion ends, new matches are found and visual servoing continue.

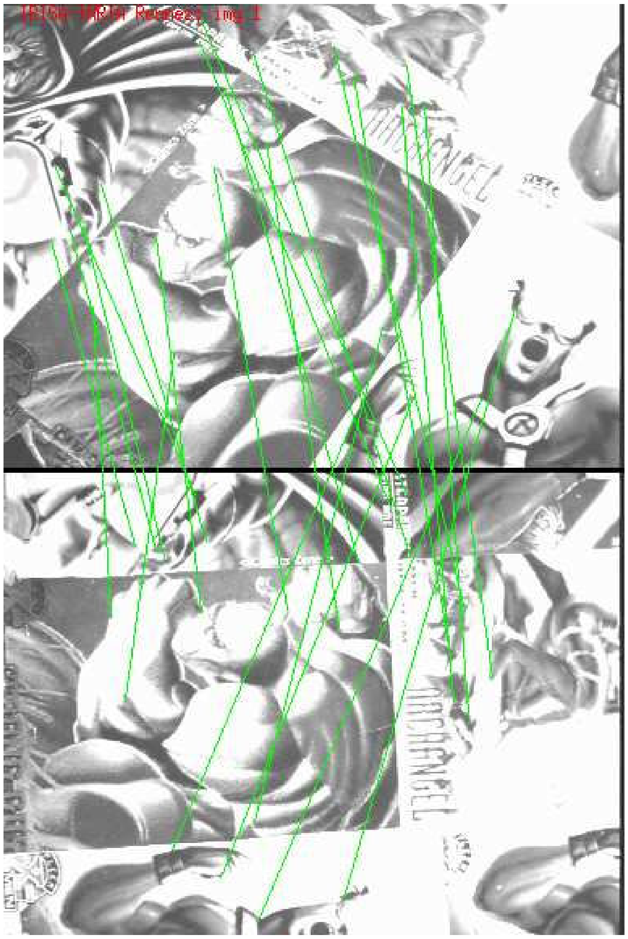

Fig. 5. Initial (top) and reference (bottom) image for the marvel experiment. Green lines link two matched point.

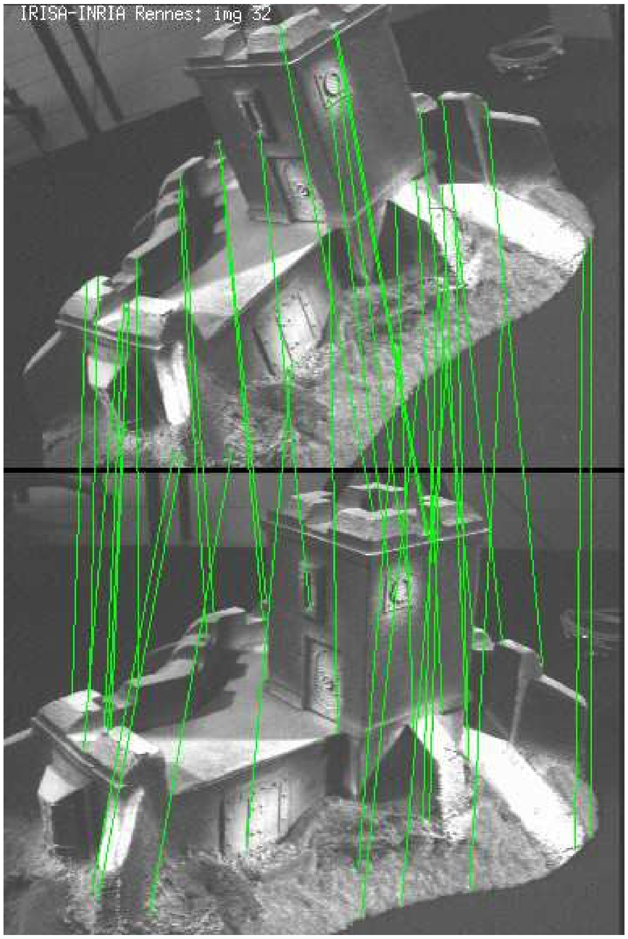

Fig. 7. Initial (top) and reference (bottom) image for the castle experiment. Green lines link two matched point.

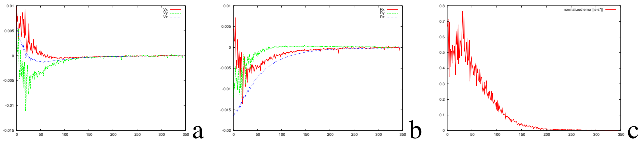

Fig. 6. Marvel experiment: Camera velocity (a) translation (b) rotation ; (c) norm of the error

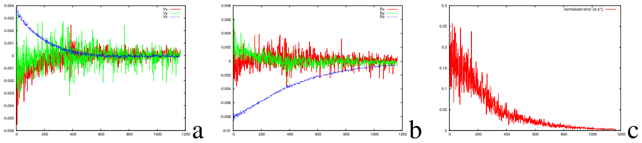

Fig. 8. Castle experiment: Camera velocity (a) translation (b) rotation ; (c) norm of the error

## V. Conclusions

In this paper, a method for tracking by matching has been proposed. Thanks to the definition of a very simple but efficient keypoint detector, efficient keypoint description and matching, this method is showed to be very efficient for real-time application like visual servoing. The matching algorithm works at 10-14Hz and is well robust to 3D viewpoint as well as illumination changes. Efficient has a price, the number of points detected and matched is smaller than in some state of the art literature methods, but it is enough for applications such as visual servoing or pose estimation.

In comparison with some existing matching methods such as SIFT[17], PCA-SIFT[15], [16], in term of computational time, our method is significantly faster than SIFT or PCA-SIFT thanks to the very fast keypoint detector. Compared to method proposed by Lepetit *et al.* [16], our method are lightly more time consumming at step of computing canonical orientations. A quantitative comparision of recognition rate betweeen methods should be performed.

The performance of the tracking by matching algorithm can be improved at some following directions. Keypoints should be detected and matched in scale space in order to give reliable result when scale are different. Specifically, training images (desired image) can be processed at several scales. At runtime, the current image will be matched with all smoothed image and the best match will be taken into account. Multi-scales approach saves times better than scale-space approach because all computations for desired image are done offline.

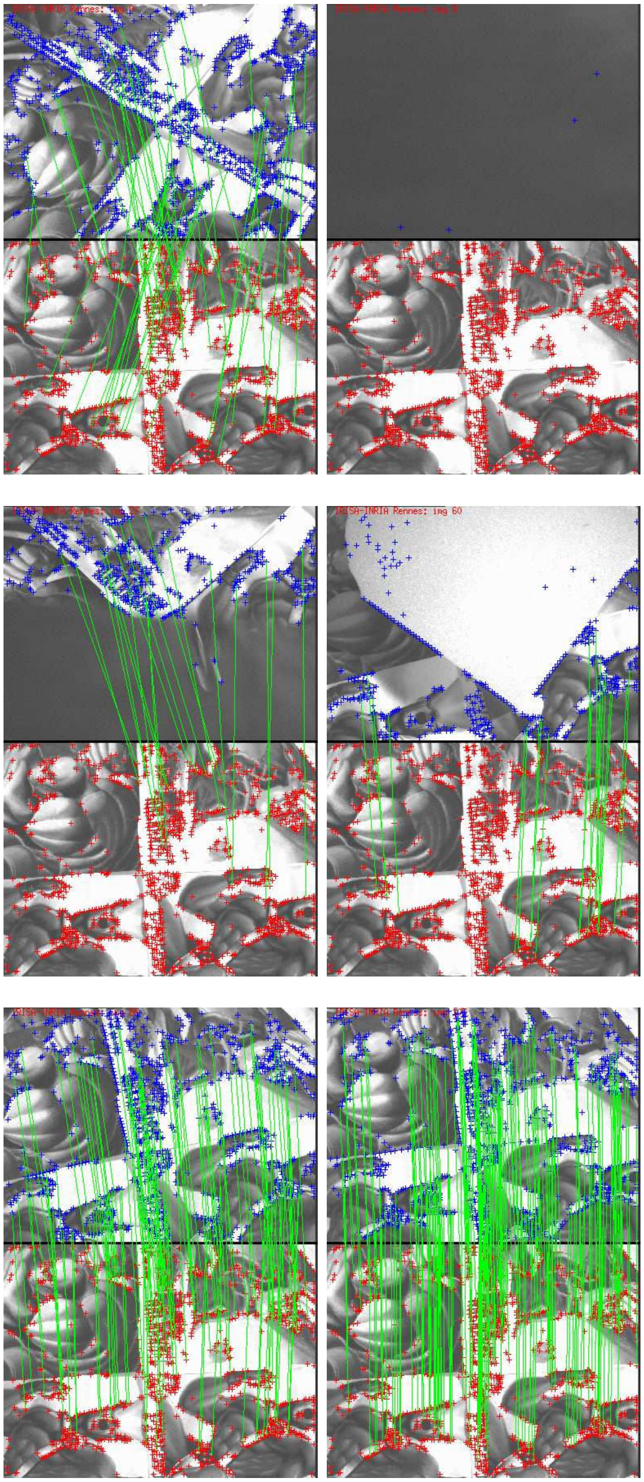

Fig. 9. Six images of the marvel sequence. Note that multiple occlusions are done (partial or complete). When the occlusion is complete, no match are found and the robot stops. When occlusion ends, new matches are found and visual servoing continue.

## References

[1] S. Arya, D.M. Mount, N.S. Netanyahu, R. Silverman, A.Y. Wu. An optimal algorithm for approximate nearest neighbor searching. *Journal of ACM*, 45(6):891–923, November 1998.

[2] S. Benhimane, E. Malis. Homography-based 2d visual tracking and servoing. *Int. Journal of Computer Vision*, 2007.

[3] S. Boukir, P. Bouthemy, F. Chaumette, D. Juvin. A local method for contour matching and its parallel implementation. *Machine Vision and Application*, 10(5/6):321–330, April 1998.

[4] A.I. Comport, E. Marchand, F. Chaumette. Robust model-based tracking for robot vision. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, IROS'04*, volume 1, pages 692–697, Sendai, Japan, September 2004.

[5] A.I. Comport, E. Marchand, F. Chaumette. Statistically robust 2d visual servoing. *IEEE Trans. on Robotics*, 22(2):415–421, apr 2006.

[6] 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, July 2006.

[7] T. Drummond, R. Cipolla. Real-time visual tracking of complex structures. *IEEE Trans. on PAMI*, 24(7):932–946, July 2002.

[8] B. Espiau, F. Chaumette, P. Rives. A new approach to visual servoing in robotics. *IEEE Trans. on Robotics and Automation*, 8(3):313–326, June 1992.

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

[10] G. Hager, P. Belhumeur. Efficient region tracking with parametric models of geometry and illumination. *IEEE Trans. on PAMI*, 20(10):1025–1039, October 1998.

[11] G. Hager and K. Toyama. The XVision system: A general-purpose substrate for portable real-time vision applications. *CVIU*, 69(1):23–37, January 1998.

[12] G. L. Hobrough. Automatic stereo plotting. *Photogramm. Eng. Remote Sens.*, 25(5):763–769, 1959.

[13] S. Hutchinson, G. Hager, and P. Corke. A tutorial on visual servo control. *IEEE Trans. on Robotics and Automation*, 12(5):651–670, October 1996.

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

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

[16] V. Lepetit and P. Fua. Keypoint recognition using randomized trees. *IEEE Trans. on PAMI*, 28(9):1465–1479, September 2006.

[17] D. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 60(2):91–110, 2004.

[18] E. Malis, F. Chaumette, and S. Boudet. 2 1/2 D visual servoing with respect to unknown objects through a new estimation scheme of camera displacement. *IJCV*, 37(1):79–97, June 2000.

[19] E. Marchand and F. Chaumette. Feature tracking for visual servoing purposes. *Robotics and Autonomous Systems*, 52(1):53–70, June 2005. special issue on "Advances in Robot Vision", D. Kragic, H. Christensen (Eds.).

[20] E. Marchand, F. Spindler, and F. Chaumette. ViSP for visual servoing: a generic software platform with a wide class of robot control skills. *IEEE Robotics and Automation Magazine*, 12(4):40–52, December 2005.

[21] K. Mikolajczyck, C. Schmid. Indexing based on scale invariant interest point. In *IEEE ICCV'01*, pages 525–531, Seattle, 2001.

[22] K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, J. Matas, F. Schaffalitzky, T. Kadir, L. Van Gool. A comparison of affine region detectors. *IJCV*, 65:43–72, 2005.

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

[24] E. Rosten, T. Drummond. Fusing points and lines for high performance tracking. In *IEEE ICCV*, volume 2, pages 1508–1515, Beijing, China, 2005.

[25] J. Shi, C. Tomasi. Good features to track. In *IEEE CVPR'94*, pages 593–600, Seattle, Washington, June 1994.