J Real-Time Image Proc manuscript No.
(will be inserted by the editor)

Iñigo Barandiaran · Céline Paloc · Manuel Graña

Real-time Optical Markerless Tracking for Augmented Reality Applications

Received: date / Revised: date

Abstract Augmented Reality (AR) technology consists in adding computer-generated information (2D/3D) to a real video sequence in such a manner that the real and virtual objects appear coexisting in the same world. In order to get a realistic illusion, the real and virtual objects must be properly aligned with respect to each other, which requires a robust real-time tracking strategy - one of the bottlenecks of AR applications. In this paper we describe the limitations and advantages of different optical tracking technologies, and we present our customized implementation of both recursive tracking and tracking by detection approaches. The second approach requires the implementation of a classifier and we propose the use of a Random Forest classifier.

We evaluated both approaches in the context of an AR application for design review. Some conclusions regarding the performance of each approach are given.

Keywords Augmented Reality, Optical Markerless tracking, Tracking by Detection

1 Introduction

The term Augmented Reality (AR) refers to a technology that allows to add virtual information to the scene seen by the user. In contrast to Virtual Reality where the entire environment is completely virtual, AR combines both virtual and real objects in the same scene. Therefore, while Virtual Reality substitutes the reality, AR enhances it. It is furthermore important to distinguish between AR and the special effects of the film industry or TV production, where some virtual characters or virtual objects appear perfectly integrated within real objects. The main difference is that AR is meant to be used in real-time, while special effects production can be processed off-line allowing the use of sophisticated techniques which can be computationally expensive.

Most of the computational costs are due to the tracking process, in order to align properly the real and virtual objects with respect to each other and to produce a realistic illusion of fusion between the two worlds. The eyesight is one of the most important senses for the perception of the human being. Hence, any discrepancy between real and virtual object would be automatically detected by the human's eye and the AR effect would be missed. Despite the rapid development of computational power and specialized hardware such as programmable graphics units (GPUs), tracking technology still suffers from a notorious lack of robustness and high computational costs. These drawbacks get drastically worse in an uncontrolled context such as outdoor, where it is difficult to calibrate the environment, add landmarks, control lighting and limit the operating range to facilitate tracking. In this paper we address the tracking problem for AR applications in uncontrolled environments.

While a large variety of tracking systems are commercially available (mechanical, acoustic, magnetic, inertial and optical sensors), most of those systems are meant to be used in perfectly known contexts, where the variables that affect the tracking can be controlled easily. In uncontrolled environment, the tracking process should work without adapting the object or the environment to be tracked, such as placing special landmarks or references. This issue is known as markerless tracking. Optical sensors have been recently widely explored to answer markerless tracking [11].

Optical markerless tracking uses natural features such as edges, corners or texture patches, extracted from the images acquired by a camera. By using natural features, the use of artefacts such as reflective markers is avoided, allowing the system to be more flexible and being able to work in non-well controlled conditions. In our approach we use the fact that natural plane surfaces are common structures either in an indoor or outdoor scenario. The ground, the building facades or walls can be seen as planes. Therefore we propose to focus our work on optical markerless tracking for planar structures in unprepared environments.

Iñigo Barandiaran
VICOMTech Paseo Mikeletegi, 57
20009, San Sebastian, Spain
Tel.: +34-943309230
E-mail: ibarandiaran@vicomtech.org

Céline Paloc E-mail: cpaloc@vicomtech.org · Manuel Graña E-mail: ccpgrrom@si.ehu.es

The paper is structured as follows: in Section 2 we review some requirements for an optical markerless tracking method to be reliable. Section 3 describes two approaches we evaluated to solve the camera pose estimation problem. In Section 4, implementation details and some results are given. Finally, Section 5 summarizes some conclusions and future work.

2 Requirements for Optical Markerless Tracking

Optical markerless tracking technology is basically based on image processing, using a digital camera as a main input data device.

Figure (2) shows a typical design of an AR application based on optical tracking. A camera is capturing images from the world (environment). These images are transferred to a computing workstation where they are processed to extract useful information, such as the camera pose transformation.

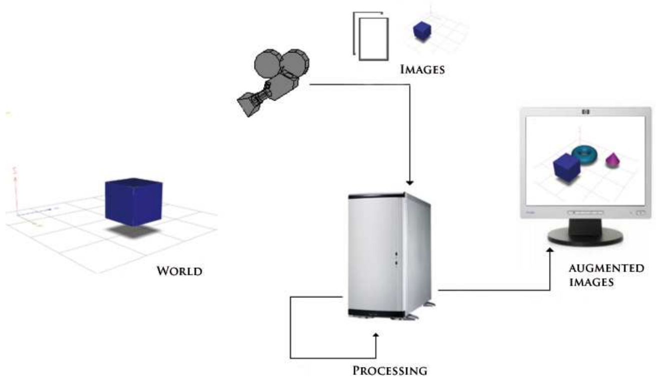

Fig. 1 Schema of an AR application based on optical markerless tracking.

The term camera pose refers to the transformation, translation and orientation, between the objects or environment coordinate system and the camera coordinate system, as depicted in Figure 2. This transformation should be estimated dynamically, as fast as possible, in order to realistically integrate virtual objects between real ones, during the tracking sequence. This estimation must be very accurate so that virtual objects appear rigidly fixed to the real world. If this transformation is inaccurate, the objets will not appear correctly aligned, as shown in Figure 3. Depending on the quality of the images, or the user motion, the pose estimation might be difficult to solve and the tracking process might fail. In these cases, some user interaction, such as the manual input of some specific points, or initial camera pose estimation could be required to help the system to compute the next camera values.

The requirements stated in this section pay special attention to the software features as well as its integration with the hardware. Key features of software are usability of the user

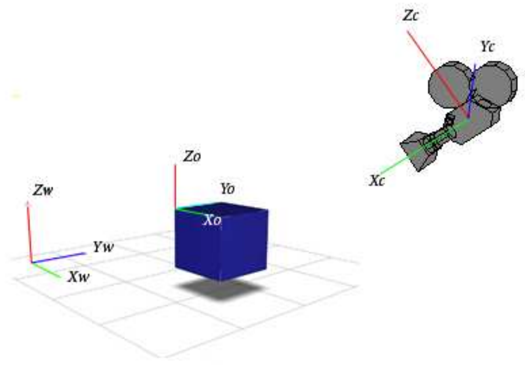

Fig. 2 World, object and camera coordinate systems.

interface, efficiency, real-time performance and robustness; the most important aspects related to hardware are portability, reliability, and costs.

2.1 Robustness

The tracking must be achieved with a minimum level of robustness, without failing, or continuously re-initializing the tracking process. Besides tracking loss, some other problems may appear such as drift or jitter. The drift problem refers to the displacement of the origin of the world coordinate system. When recursive techniques are applied to estimate the camera pose, some error accumulation over time may occur. This error accumulation causes the impression that the virtual objects are floating between real ones. The jitter problem is caused by small variation in the camera pose transformation between frames, even when there is no variation in neither the objects nor the camera. This difference causes the effect that virtual objects appear flickering in the images, not being rigidly fixed in the real world. Such effects should be avoided as much as possible.

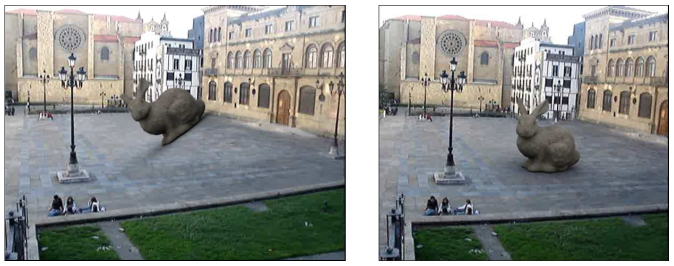

Fig. 3 (a)Wrong camera pose estimation, (b) Correct camera pose estimation, the virtual object appears correctly aligned with real world.

2.2 Performance

The tracking process must be fast, minimizing the time needed for the pose computation, so that achieve near real time system performance is achieved. Any delay between image acquisition and final image generation will deteriorate the effect of integration of virtual and real objects. In some cases, the rendering of the final image can be synchronized with the virtual object generation, for example when using a video see through head mounted display. However, with this approach the frame rate could be lower than real-time. If the generated images are highly delayed, the user will perceive the difference between the physical stimulus of its movements and the visual stimulus, making the AR application uncomfortable. Such delay should be reduced as much as possible.

2.3 Set-up Time

The initial setup on-site includes the following tasks:

1. Camera calibration: The calibration step is mainly related to the estimation of the internal camera parameters, i.e. focal length, principal point, and possible radial or tangential distortion. The calibration should be performed only once.
2. Model acquisition: Some a priori knowledge of the environment or object to be tracked must be obtained. This task consists in acquiring and processing some information of the environment that the user wants to track in order to construct a model used as a reference.

The initial on site setup must be as short as possible for the AR application to be practical.

3 Methods

Our approach to obtain the camera pose is based on the tracking of plane surfaces. The 3D world planes (ground, building facades, walls) and their projection in the image are related by a plane to plane projective transformation,also known as homography or collineation. It can be modelled as a 3x3 matrix H with eight degrees of freedom. The camera pose can be recovered by estimating this homography between a world plane and its image. This estimation can be carried out by tracking points lying on the world plane, and matching them frame by frame.

For optical markerless tracking two main groups can be distinguished: recursive and tracking by detection techniques. Recursive techniques start the tracking process using an initial guess or a rough estimation, and then refine or update it over time. They are called recursive because they use the previous estimation for calculate the next one.

Contrary, tracking by detection techniques can do a frame by frame computation independently from previous estimations. In this case, some a priori information about the environment or the objects to be tracked is needed.

We have worked on the camera pose estimation problem using two different approaches. The first one is based on recursive tracking and the second one based on tracking by detection method. The latter requires the implementation of a keypoint classifier, which directly impacts on the tracking performance. In the following, we present these methods in detail.

3.1 Camera Pose Estimation

As described in section 2 this problems tries to find the geometric transformation between two coordinate systems, more precisely, between the world or object coordinate system and the camera coordinate system. When this transformation is obtained, a virtual camera can be transformed accordingly, and so the virtual objects can be accurately aligned in the images.

The camera pose represents a transformation compound of a rotation matrix R and a translation vector t between two coordinate systems.

$$Rt = \begin{pmatrix} R_{11}\ R_{12}\ R_{13} \mid t_x \\ R_{21}\ R_{22}\ R_{23} \mid t_y \\ R_{31}\ R_{32}\ R_{33} \mid t_z \end{pmatrix} \qquad (1)$$

Besides the external camera parameters of Equation 1, the internal or intrinsic camera parameters define how the camera projects the points on to the image plane, excluding geometrical distortions:

$$K = \begin{pmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix} \qquad (2)$$

where c_x and c_y represents the coordinates of the principal point, f_x and f_y the focal length and s the skew, all in pixel units. This internal camera parameters of Equation 2 are independent of the camera pose.

The whole world to image projection mechanism can be described as:

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = KRt \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \qquad (3)$$

Equation (3) represents the projection of a point defined in world coordinate system, to a point in image coordinate system, both in homogeneous coordinates, given a pinhole calibrated camera model. This equation is usually represented more briefly as $m = PM$, where $P = KRt$.

If we choose that the Z coordinate equals zero for all points M of the world plane π, we obtain:

$$m = PM = [p_1 p_2 p_3 p_4] \begin{bmatrix} X \\ Y \\ 0 \\ 1 \end{bmatrix} = [p_1 p_2 p_4] \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} \qquad (4)$$

where each p_i represents a column vector of the matrix P. Then the mapping between points on the world plane $M_\pi = (X,Y,1)^t$ and their image m is a planar homography $m =$

HM_π, where $H = [p_1 p_2 p_4]$. This expression can be written as:

$$H = K[r_1, r_2, t] \tag{5}$$

where r_i are the columns of the rotation Matrix R and t the translation vector.

For the estimation of the homography H, it is needed to find some point correspondences in both planes $M_{\pi i} \Leftrightarrow m_i$, where $M_{\pi i}$ is the point in the world plane π and m_i is its corresponding point in the image. In the context of markerless tracking, these points are known as natural features.

The expression $m = HM_\pi$ can be rewritten as:

$$mxHM_\pi = 0 \tag{6}$$

Each correspondence $M_{\pi i} \Leftrightarrow m_i$ gives rise to two linearly independent equations in the entries of H. Therefore, as a homography has eight degrees of freedom, only four coplanar non-collinear points are needed. Given four or more correspondences, we obtain a set of equations $Ah = 0$ where A is a matrix of equation coefficients contributed from each correspondence and h is the vector of unknown entries of H.

If more than four correspondences are given, then the linear system $Ah = 0$ is over-determined and usually noisy. So, only an approximated solution can be obtained, for example by using singular value decomposition (*SVD*). This method is known as Direct Linear Transformation (*DLT*)[8].

As mentioned above, a calibrated camera is represented as $P = KRt$. Once H and the internal camera parameters K are known, the camera pose Rt can be recovered from equation 5,as:

$$HK^{-1} = (R^1 R^2 t) \tag{7}$$

where K^{-1} is the inverse of the internal camera parameters matrix, t is the translation vector, $r_1 r_2$ are the two columns of the camera rotation matrix. The third column of the rotation matrix R^3 can be obtained by the cross-product of r_1 and r_2.

Since we are using the internal camera parameters for the camera pose estimation, the camera must be calibrated beforehand. This calibration task is carried out by taking several images of a calibration pattern, for example a picture with white and black squares from different distances and points of view as shown in Figure 4 [20].

3.2 Recursive Tracking

In the first steps of the development of the markerless tracking module, we implemented a recursive tracker to solve the homography estimation. Recursive tracking techniques start the tracking process from an initial guess or a rough estimation, and then refine or update it over time. They are called recursive because they use the previous estimation to propagate or calculate the next estimation.

In this recursive approach, the initialization of the tracking process consist in selecting manually in the images the projection of four points lying on the same plane in the 3D world. Once these four points are available, the first homography estimation takes place starting the tracking process. During the tracking process, this homography is continuously updated by extracting points from images, using the Harris operator [9], matching them between previous and current images and calculating a new homography.

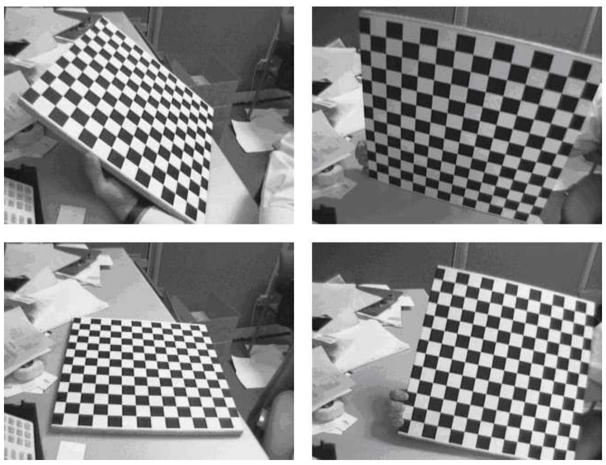
Fig. 4 Images of a calibration pattern taken with different orientations and scales.

When only four points are used to estimate the homography, it is said that a minimal solution is obtained. In this context, minimal means that any error generated in the location of any of the four points will degenerate the estimation. Depending on the generated error, the estimated homography can be completely distorted.

For this reason the homography estimation is typically performed using more than four points correspondences. A point correspondence is considered an outlier if it is generated from another plane or if it is a wrong correspondence (mismatch) between points of the same plane. Algorithms using random sampling are well-known methods for robust matching even in the presence of outliers. Those algorithms, applied to homography estimation search randomly a combination of four points from the available candidates and estimate a transformation. The estimated transformation is then tested with the rest of points. The transformation that obtains more support (number of inlier points) after some fixed number of iterations is selected as the best one. In our prototype, we integrated the RANSAC method.

During the estimation process several errors may occur, such as point miss-matching due to severe changes of the illumination conditions or fast movements (motion blur). Due to the recursive nature of this kind of tracking, this approach is highly prone to error accumulation. The error accumulation over time may induce a tracking failure, requiring a re-initialization of the tracking process, which can be cumbersome and not feasible in practical applications.

3.3 Tracking by Detection

Other approaches are known as tracking by detection. In this kind of techniques some information of the environment or the object to be tracked is known a priori. They are also known as model-based tracking because the identification of some features in the images (texture patches or corners) corresponding to a known model are used to recognize such objects.

This kind of tracking does not suffer from error accumulation because, generally, does not rely on the past. Furthermore, these methods are able to recover from a tracking failure since they are based on a frame by frame estimation. They can handle problems such as matching errors or partial occlusion, being able to recover from tracking failure without intervention [19].

Tracking by detection needs data about the objects to be tracked prior to the tracking process itself. This data can be in form of a list of 3D edges (CAD model) [18], color features, texture patches or point descriptors [13, 14]. Then the tracker is trained with this a priori data, to be able to recognize the object from different points of view. A good survey about different model-based tracking approaches can be found in [11, 21].

Some authors propose the use of machine learning techniques to solve the problem of wide baseline keypoint matching [3, 15]. Supervised classification systems requires a preprocessing, where a system is trained with a determined set of known examples (training set) that represents variations in all their independent variables. Once the system is trained, it is ready to classify new examples. Some of the most widely used supervised classifiers are for example, k-Nearest Neighbors, Support Vector Machine or decision trees. While k-Nearest Neighbors or Support Vector Machine can achieve good classification results, they are still too slow and therefore not suitable for real-time operation [10].

Recently the approach based on decision trees has been successfully applied to tracking by detection during feature point matching task, by training the classifier to establish correspondences between detected features in a training image and those in input frames [15].

In previous work [2], and based on the work of [12], we showed that Random Forest is a suitable classifier that can be applied in markerless tracking. This classifier is computationally fast and able to support a large number of different classes in high dimensional spaces (the number of features in each class).

In the following section the approach based on Random Forest is described in more detail.

3.3.1 Random Forest

We propose a supervised classification method based on Random Forest for interest point matching. This classifier is a multi-classifier based on Random Trees. This classifiers are a specific variation of a decision tree [4]. When the tree is constructed and trained, each node contains a discriminant criteria which allows to decide how to go down or traverse the tree. The classifier is able to detect key-point occurrences even in the presence of image noise, variations in scale, orientation and illumination changes.

A Random Tree is called random because instead of performing exhaustive search in order to find the best combination of features to define a discriminant criteria in each node, just some random combinations of them are evaluated. When the number of different classes to be recognized and the size of the descriptor of such classes is high, an exhaustive analysis is not feasible. Additionally, the examples to be used for the training process are selected at random from the available ones. The combination of several random trees forms a multi-classifier known as Random Forest. One of the advantages of the Random Forest is its combinational behavior. If a random tree can be weak itself, i.e. its recognition rate is low, then the combination of such weak tree can generate a strong classifier [4].

3.3.2 Training

In a typical supervised learning scenario, a training set is given and with the goal to form a description that can be used to predict previously unseen examples and recognize known examples. Each class must be defined and described before the training process itself. In supervised classification a class C_i is defined as a set of attributes a_i, known as features $C_i = \{a_1, a_2, ..., a_n\}$.

In order to define the classes that will be recognized by the classifier, we use a point extractor [17] to get the candidate points and their surrounding patches, as shown in Figure 5.

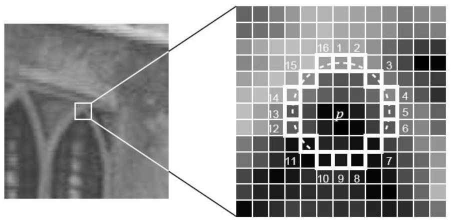

Fig. 5 (a) Interest point p, (b) Pixels surrounding the interest point p [17]

Then, the classifier assigns a class number to each point, and their class descriptor is defined. The descriptor of each class is constructed as the pixel intensity values of the extracted patch centered at the interest key-point. Once the classes to be recognized by the classifier are defined, a training set must be generated. As described in [12], we can exploit the fact that the patches belong to a planar surface. Therefore, we can then synthesize different new views of the patches using warping techniques as affine deformations. These affine transformations are needed to allow the classifier to identify or recognize the same class seen from different points of view and at different scales. This step is particularly important, when the camera will be freely moving around the object.

Once the training set is ready, the training task can be started. During this task, a number of examples are randomly selected from the available ones. These examples are pushed down in the trees. In order to decrease the correlation between trees, and thus increase the strength of the classifier, different examples from the training set must be pushed down in each tree. This randomness injection favors the minimization of trees correlation and avoids overfitting as well. This term refers to the situation in which the training algorithm generates a classifier which perfectly fits the training data but has lost the capacity of generalizing instances not presented during training.

While building up the tree, each non-terminal node of every tree is treated as follows:

1. N training examples from the training reach the current node.
2. A random set of n pixel positions within the image are selected and written in that node.
3. The examples are tested with the selected set of pixels. Depending on the result of this test, they are pushed down to their corresponding child node.
4. The above process is recursively applied to the children nodes, whether until there is only one example, or only one class is represented in the remaining examples or the maximal predefined depth is reached. As shown in Figure 6 when the examples reach a leaf node, the posterior probability distribution are updated with those examples.

Once the descriptors reach the bottom (maximal depth) of the tree, it is said that they have reached a terminal node or a leaf node, and the recursion stops. In leaf nodes the class posterior probability distributions are stored. These distributions represent the ratio class examples from the training set that has reached that node, with respect to the total number of examples in the training set. When an example of a given class has reached a leaf node, the posterior probability distribution stored in that node must be updated accordingly.

The tests to be performed in each node j in every tree k can be, for example, binary tests based on the comparison of the intensity values of two pixels as:

$$n_{k,j} = \begin{cases} GoLeftChild & if(p_{j,1} - p_{j,2}) \geq t \\ GoRightChild & \text{otherwise} \end{cases} \quad (8)$$

Where $v(p_{j,1})$ and $v(p_{j,2})$ represent the intensity values of two pixels located respectively at positions $p_{j,1}$ and $p_{j,2}$ stored in node j. The values of these positions were randomly selected during the training step. The value of t represents a threshold that can also be randomly selected during training. We have also experimented that, given the weakness of the tests, smoothing every patch before training and classification, significantly increases the final reliability of the classification.

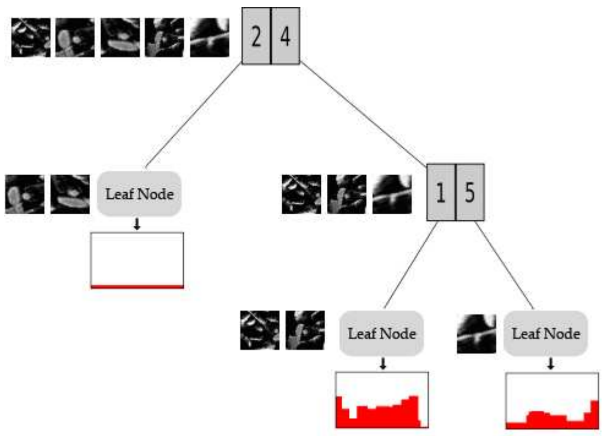

Fig. 6 Random Tree construction: When the examples reach leaf nodes, then the posterior probability distributions are updated.

3.3.3 Tracking

Once the classifier is built, i.e. the pixels to be tested in each node and the class posterior distributions of all classes are calculated, it is ready to identify keypoints. During the classification task any example (image patch) is dropped down in every tree that constitutes the forest. These examples will be dropped down the tree until they reach a leaf(terminal) node. The node they reach will depend on the results of the tests (Equ. EQU:travers) obtained in the previous non-terminal nodes they visit, as depicted in Figure (7).

As depicted in Figure 7 the example to be classified traverses the tree until it reaches a leaf node. When the example reaches a leaf node, the tree will return the posterior probability distribution vector stored in that node. This probability vector represents, for each class, the probability of the example to be an instance of one of the trained class. This is $P(Y = C_i|T_i, n = \eta)$ where T_i is a given tree of the forest and η is the reached node by the example (image patch) Y and C_i represent every class that was previously trained, during the training step. The size of the posterior distributions vector equals the number of different classes trained by the classifier.

As explained in Section 3.3.1, a Random Forest is a multi-classifier, i.e. is a set M of classifiers T_i $M = T_1, T_2, ..., T_n$. The main idea of a combination methodology is to combine a set of models (*classifiers*), each of them solving the same original task, in order to obtain a better composite global model, with more accurate and reliable estimates or decisions than those obtained from a single model. Like any other multi-classifier, the Random Forest needs to combine the independently generated output by each tree in the forest in order to assign a final class label to the examples to be classified.

In our approach we are using a distribution summation combining method [1]. This method sums up the conditional probability vector obtained independently by each tree in the forest. The selected class is chosen according to the highest

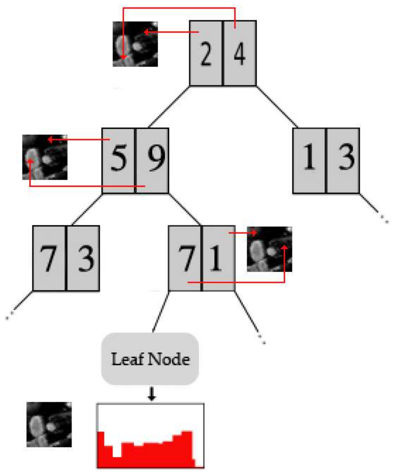

Fig. 7 Example of image patch classification: The image patch traverse the tree until a terminal node is reached.

value in the total vector:

$$Class(x) = argmax_{c_i} \sum_k P_k(Y = c_i|x) \tag{9}$$

During tracking, the Random Forest classifier is applied to interest point matching between points m extracted from images and points M of the model. With the set of putative matches $M_i \Leftrightarrow m_i$ the homography estimation can be obtained as explained in Section 3.1. After the classification step, wrong classified examples (outliers) can be removed by using robust estimation techniques such as RANSAC [7] in order to obtain a more accurate homography estimation. Furthermore, the final estimation can be refined by using Levenberg Marquardt non-linear minimization starting from the estimation obtained by RANSAC, and using all the inlier points. This non-linear minimization favors the reduction of the jitter problem (see Section 2), obtaining more accurate estimations. More details are given in [1].

4 Application to markerless tracking

The approaches for markerless tracking described previously were applied within an innovative system for collaborative mobile mixed reality design indoor and outdoor review. In the next section we describe this application more in detail.

4.1 Implementation Details

As described before, our tracking module is based on markerless techniques using natural features to estimate the position and orientation of the digital camera. The tracking module can be used either in indoor or outdoor scenarios, where a well textured plane is present. Figure 8 shows the tracking module working on indoor and outdoor scenarios.

The application involves the integration of different modules such as visualization device (HMD, display wall), rendering, image transmission and tracking. All these modules

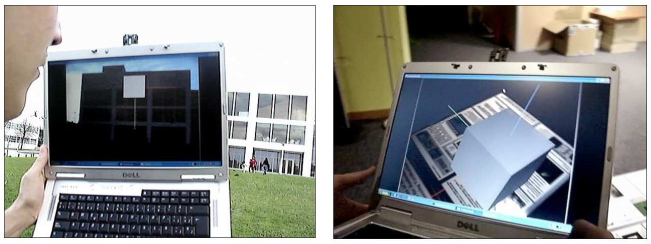

Fig. 8 (a) Outdoor tracking of a building facade, (b) indoor tracking of a textured floor.

can interact and share information through a communication module or communication backbone. Similarly the other subsystems proposed in the design, the tracking subsystem needs to be connected to the communication backbone in order to deliver tracking information to other modules, like the rendering module. The rendering module will uses tracking information (camera pose) to update the virtual camera accordingly and therefore renders the virtual objects correctly aligned with real ones. The connection of the markerless tracking module with the communication backbone is realized by using OpenTracker [16]. OpenTracker is an open software architecture that allows the interaction between different tracking approaches and tracking input devices.

During the tracking process, every new camera pose estimation must be converted to an OpenTracker state structure and delivered through the communication backbone to be accessible for other clients.

As described earlier, tracking by detection techniques require an off-line process when the classifier is trained. For this task, one image of a highly textured plane, such as a building facade or a picture over a table, must be acquired. After the acquisition, some feature points and their surrounding texture patches are extracted from the image, and synthetic views of the plane are generated. Afterwards, the training step starts automatically using the generated views as the training set. Once the training period is finished, the system is ready for tracking as shown in Figure (9).

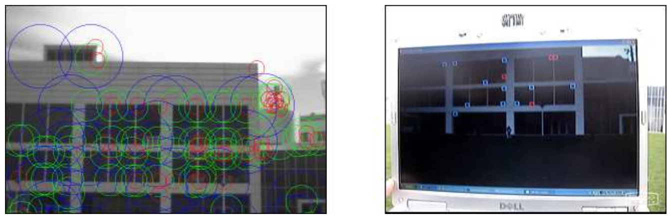

Fig. 9 (a) Keypoints extracted from a building facade. (b) Classifier training.

The markerless tracking module uses OpenCV for image acquisition and processing, and our own C++ implementation of the Random Forest classifier for the keypoint recognition and matching.

4.2 Results

The approach based on recursive tracking is very unstable, tending to fail easily. Moreover, it does not allow to move rapidly the camera, as of image blur generates tracking error. In comparison with the recursive tracking implementation, the tracking by detection module allows the tracking to run faster, being more robust against partial object occlusion, or fast camera movement.

The classifier integrated in the tracking by detection module is trained to be able to recognize about 150 different classes (image patches). The Random Forest classifier is constructed with 15-20 trees. As mentioned before, every tree that forms the forest is independent from the rest and will generate an individual output. The forest is trained with a training set of 1000 synthetically generated new examples. This training step, i.e. the set up preparation takes less than 5 minutes. This size of the training set is a good compromise between training time and final accuracy of the classifier. Training time is a very important factor in practical situations such as outdoor.

In order to evaluate the computational costs of the method, we conducted some tests using different hardware with the same memory amount but different CPUs. The obtained frame rate on different CPUs is given on Figure 10. The obtained

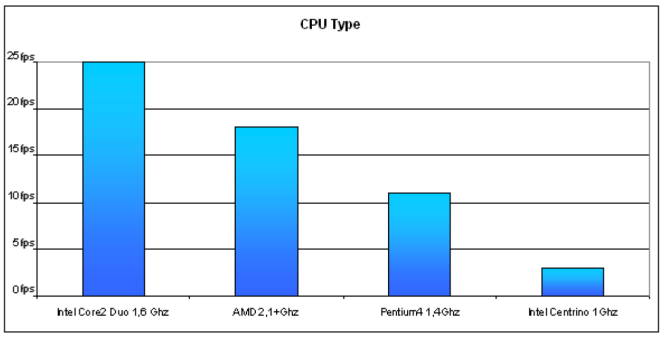

Fig. 10 Frame Rate Tracking Results

frame rate with SVGA image resolution is about 20-25 frames per second (near real-time) on a 1.6Ghz dual core CPU by using the Random Forest based classification technique. This frame rate may vary depending on the accuracy of the tracker, i.e. depending on the number of different points to be recognized. More details can be found in [2].

For about 150 points the tracker obtains good accuracy and the frame rate is near real-time. The drift and jitter are well controlled, so no severe displacements of the objects occur. On older CPUs, the obtained frame rate is lower, for the same number of points and trees. Better frame rates can be obtained by switching off the jitter filtering module or reducing the maximum number of identifiable points, but the accuracy decreases, increasing the object jittering. Regarding robustness and practicality, the tracker can run indefinitely without requiring a new initialization.

5 Conclusions and Future Work

In this work we have presented two approaches to solve the camera pose estimation problem in uncontrolled environment. While the recursive approach is computationally light, it is also very unstable and tends to fail, losing tracking information easily. The approach based on tracking by detection is more robust.It does not accumulate tracking errors over time and can obtain real-time frame rate.

We selected the Random Forest based classifier, as being fast, accurate enough and supporting a high number of identifiable classes, which makes it more robust against partial object occlusions. As a drawback, this approach requires a pre-processing to train the classifier with images of the plane to be tracked.

We think that machine learning techniques such as Random Forest is a very promising technique for optical markerless tracking. We project to extend our work to support online training classification, like in [15]. The advantage of online training is that it allows the tracking to update the model with new feature points not present in the original training set. This increases the robustness of the model and the overall accuracy of classification rate. As described in [19] online training can be exploited in several frameworks such as Simultaneous Localization and Mapping (SLAM), or other recursive techniques [5, 6] as a tracking initialization mechanism.

Acknowledgements This work has been partially funded under the 6th Framework Programme of the European Union (EU) within the project ”IMPROVE”: IST FP6- 004785.

References

1. Barandiaran, I., Cottez, C., Paloc, C., Graña, M.: Random Forest Classifier for Real-Time Optical Markerless Tracking. In Proc. of VISAPP, Vol:2, 559-564 (2008) ISBN:978-989-8111-21-0
2. Barandiaran, I., Cottez, C., Paloc, C., Graña, M.: Comparative Evaluation of Random Forest and Fern Classifiers for Real-Time Feature Matching. In Proc. of WSCG, 159-166 (2008) ISBN 978-80-86943-15-2
3. Boffy, A., Tsin, Y. Genc, Y.: Real-Time Feature Matching using Adaptative and Spatially Distributed Classification Trees. In Proc. of BMVC. Vol:2, 529-539 (2006)
4. Breiman, L.: Random Forests. Machine Learning Journal, Vol:45, 5-32 (2001)
5. Chandaria, J., Graham, T., Stricker, D.: The MATRIS project: real-time markerless camera tracking for Augmented Reality and broadcast applications. Journal of Real Time Image Processing, Vol:2 69-79 (2007) DOI:10.1007/s11554-007-0043-z
6. Davison, A.J., Mayol, W., Murray, D.W.: Real-time localisation and mapping with wearable active vision. In Proc. of ISMAR (2003)

7. Fischler, M. A., Bolles, R. C.: Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. Comm. of the ACM, Vol:24, 381-395 (1981)
8. Hartley, R., Zisserman, A.: Multiple View Geometry in Computer Vision. Cambridge University Press, 2nd edition (2004) ISBN: 0521-54051-8
9. Harris, C. Stephens, M. J.: A combined corner and edge detector. In Alvey Vision Conference, 147-152 (1988)
10. Lepetit, V., Pilet, J., Fua, P.: Point Matching as a Classification Problem for Fast and Robust Object Pose Estimation. In Proc. of CVPR, Vol:2, 244-250 (2004) ISSN: 1063-6919
11. Lepetit, V., Fua, P.: Monocular model-based 3D object tracking of rigid objects: A survey. Foundations and Trends in Computer Graphics and Vision, Vol:1, 1-89 (2005)
12. Lepetit, V., Fua, P.: Keypoint Recognition Using Randomized Trees. IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol:28(9), 1465-1479 (2006) ISSN: 0162-8828
13. Lowe, D.: Distinctive Image Features from Scale Invariants Keypoints. International Journal of Computer Vision, Vol: 20(2) 91-110 (2004)
14. Mikolajczyk, K., Tuytelaars, T., Schmid, C., Zisserman, A., Matas, J., Schaffalitzky, F., Kadir, T., Gool, L. V.: A Comparison of Affine Region Detectors. Int. Journal of Computer Vision, Vol: 65(1-2) 43-72 (2005) ISSN:0920-5691
15. Özuysal, M., Fua, P., Lepetit, V.: Feature Harvesting for Tracking-By-Detection. In Proc. of European Conference on Computer Vision, 592-605 (2006) ISBN:3-540-33836-5
16. Reitmayr, G., Schmalstieg, D.: OpenTracker: A Flexible Software Design for Three-Dimensional Interaction. Virtual Reality Journal, Vol:9, 79-92 (2005) DOI: 10.1007/s10055-005-0006-2
17. Rosten, E., Drummond, T.: Machine Learning for High-Speed Corner Detection. In Proc. of European Conference on Computer Vision, 430-443 (2006) ISBN 3540338322
18. Vacchetti, L., Lepetit, V., Fua, P.: Combining Edge and Texture Information for Real-Time Accurate 3D Camera Tracking. In Proc. of IEEE and AM International Symposium on Mixed and Augmented Reality, Vol:4, 48-57 (2004) ISBN:0-7695-2191-6
19. Williams, B., Klein, G., Reid, I.: Real-time SLAM Relocalisation. In Proc. of IEEE Interrnational Conference on Computer Vision, 1-8 (2007)
20. Zhang, Z.: A flexible new technique for camera calibration. IEEE Trans Pattern Anal. Mach. Intell. Vol:22(11), 13301334 (2000). DOI: 10.1109/34.888718
21. Lepetit, V., ”On Computer Vision for Augmented Reality,” in Pro-ceedings of the International Symposium on Ubiquitous Virtual Re- ality, pp 13-16, 2008.