Model Based Markerless 3D Tracking applied to Augmented Reality

João Paulo Lima, Francisco Simões, Lucas Figueiredo, Judith Kelner
Virtual Reality and Multimedia Research Group (GRVM) - Informatics Center (CIn)
Federal University of Pernambuco (UFPE)
Recife/PE, Brazil
{jpsml, fpms, lsf, jk}@cin.ufpe.br

Abstract— **This paper presents the implementation of 3D tracking techniques based on natural features for augmented reality. The contemplated techniques are from the model based category, comprising recursive and non-recursive methods, as well as edge and texture based techniques. An evaluation of the implemented 3D trackers was performed regarding performance and accuracy under different scenarios.**

markerless tracking; augmented reality; computer vision

I. Introduction

AR systems support the coexistence of real elements (that are part of users' world) and synthetic ones (computer generated) in the same environment [1]. Nowadays, this kind of user interface has obtained more attention due to the fact that it allows users performing tasks in a more intuitive, efficient and effective way. AR interfaces superimpose virtual information – 2D or 3D, textual or pictorial – onto real world scenes in real-time, registered in 3D, and allow users interaction with real and virtual elements simultaneously. In this kind of interface the real environment takes part of the application context. In AR the technical challenges lie in determining, in real-time, what should be shown where, and how. The latter problem is especially important when the visual appeal of the result is crucial. Then substantial effort must go into seamlessly fitting the information into the scene, according to the objectives of the system [2]. Ideally, AR proposes that the user must not be able to distinguish between real and virtual information, demanding that the virtual elements show both geometric (correct placement, correct size, occlusions identification) and photometric (shadowing, mutual reflections, chromatic adaptation to scene illumination) consistency. Even under simplified conditions these problems cannot be trivially solved.

The problem related to correctly positioning virtual information relative to the real environment, called registration, is solved by tracking the environment so that the synthetic elements can be adequately registered with the real scene. There are diverse tracking technologies available, such as optical sensors, movement sensors, thermal imaging, ultrasound, magnetic sensors, GPSs, among others [3]. They capture features from the real world, and based on this information the AR system determines when, where and how the virtual scene should be exhibited.

Optical tracking is often used for this purpose due to cost, accuracy and robustness requirements. Two types of optical tracking can be cited: marker based and markerless. Marker based tracking is a more well established approach for registration. It makes use of known artificial patterns placed along the environment in order to perform camera pose estimation. On the other hand, markerless tracking differs from the former one by the method used to place virtual objects in the real scene. In markerless AR any part of the real environment may be used as a marker, since the system exploits natural features present in the real scene to perform tracking. Markerless AR has received more attention from researchers in the latest years, and presents important challenges to be overcome.

Markerless AR systems use natural features instead of fiducial markers in order to perform tracking. Therefore, there are no ambient intrusive markers that are not really part of the world. Furthermore, markerless AR counts on specialized and robust trackers. Another advantage is the possibility of extracting from the surroundings characteristic information that may later be used by the AR system for other purposes.

In this paper, we address an online monocular markerless AR approach. Optical tracking presents some advantages when compared to its counterparts, such as higher precision and less sensibility to interference. Besides that, the use of a single camera allows lower cost and more compact systems. Calibration issues are also easier to be managed. Nonetheless, it is important to mention that tracking and registration techniques are more complex in markerless AR systems.

In several AR application scenarios, markerless tracking is mandatory or at least desirable [4]. An example of such application is an AR system for equipment maintenance [5]. Using markers for tracking the equipment presents many disadvantages: tracking failures can occur due to occlusion of markers by the user's body and tools; the markers can hide important parts of the equipment; the equipment pose has to be calibrated with each marker present at the scene. Therefore, a markerless tracking approach is strongly advised in such scenario.

This paper details the development of some markerless 3D tracking techniques applied to AR. The techniques addressed in this work belong to the model based category of the markerless

AR taxonomy. This taxonomy is further explained in Section III. Three different markerless tracking methods were developed: point sampling [6], interest point based [7] and keypoint based [8]. They differ by the features of the real objects that are exploited for tracking purposes: point sampling uses edges, while interest point based and keypoint based use textures. They also differ by their tracking nature: point sampling and interest point based are recursive, which means that they use the last calculated pose as an estimate for the current pose; keypoint based is non-recursive, being capable of using just the information from the current frame in order to estimate the pose.

The main contributions of this work are: (1) Survey and taxonomy of existing markerless tracking techniques for AR; (2) Modifications performed in some phases of the implemented techniques, such as using the Moving Edges method in the point sampling technique, as well as the SURF keypoint and the Lu algorithm in the keypoint based technique; (3) The Edge-ID algorithm, which is a novel method for visible edges detection; (4) Evaluation of the implemented techniques and comparison under different configurations, which can be used as a reference by other AR researchers and practitioners.

This paper is organized as follows. Section II describes the main concepts of camera representation and robust pose estimation, required for performing markerless 3D tracking for AR, and more specifically model based tracking. Section III explains how markerless 3D tracking methods for AR can be categorized and its main concepts. Section IV describes the model based techniques developed in this work. Section V discusses the results obtained with each method. The conclusions and future work are shown in Section VI.

II. Mathematical Background

Camera tracking, which is a fundamental aspect in tracking and register phases, comes from recovering information that correctly describes a virtual camera used to position virtual objects in the real scene and to render these objects in the image. There are many models for projecting 3D objects onto 2D images, varying between simple pinhole (perspective) camera models to complex lenses models that simulate human eyes [9]. In this work, it was considered the pinhole camera model without distortion factors (lenses), which is a well known simple model that correctly approximates a virtual camera in terms of geometry.

In all camera models, virtual objects are defined in a general coordinate system, also called world coordinate system (w_x,w_y,w_z), in a way to have a generic description that does not depend on the camera system used (c_x,c_y,c_z). The camera system corresponds to the world coordinate system after applying a rotation and translation transform and, because of that, it is necessary to get object coordinates from the world coordinate system to the camera coordinate system before projecting it onto the image plane (see Fig. 1). This affine transform is described by the composition of the rotation R_{3x3} and translation t_{3x1} matrices, resulting in a $[R|t]_{3x4}$ matrix. When applied to the homogeneous coordinates of the 3D point, the composed matrix leads to the same 3D point in the camera coordinate system. This matrix is called extrinsic parameters matrix because of its relation with the virtual camera model movement.

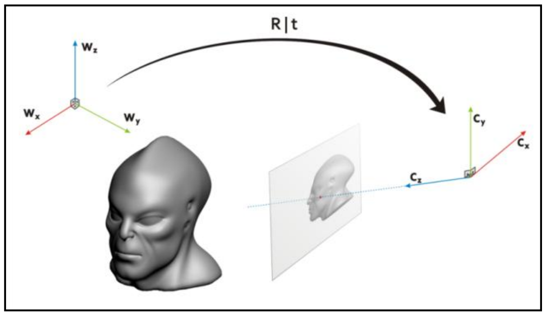


Figure 1. The 3D object, its projection onto the image plane and the relation between world (w_x,w_y,w_z) and camera (c_x,c_y,c_z) coordinate systems.

It is also important to observe that, for other purposes like pose estimation, there are many ways to represent the rotation transform. One of them is the axes-angle representation, which corresponds to a vector representing a fixed rotation axis $(w_x,w_y,w_z)^t$, and its norm referring to a rotation angle θ. This representation has an one-to-one correspondence to the R_{3x3} form by using the Rodrigues and inverse Rodrigues formula [10].

In the pinhole camera model, a point in image plane $m=[u,v,f]^t$ is obtained by projecting the 3D point $M=[x,y,z]^t$, written in camera coordinate system, onto the image plane by obeying to perspective projection conditions (see Fig. 2). By similarity of triangles,

$$u=\frac{X}{Z}f+u_0 \text{ and } v=\frac{Y}{Z}f+v_0. \quad (1)$$

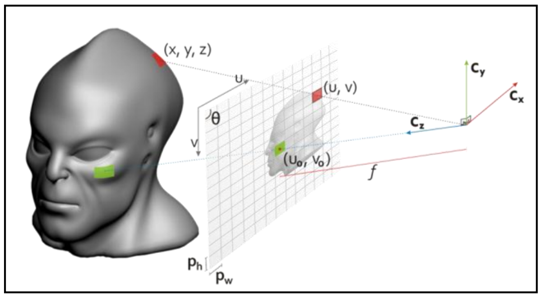


Figure 2. Perspective transform from 3D points to image points.

However, the image plane is divided in pixels units that correspond in the real world to well defined areas with dimensions written in millimeters. They are, by default, called pixel width (p_w) and pixel height (p_h)[1]. Considering that, the dimensions of u and v are not written in millimeters but in pixels, and by this the equation (1) must be rewritten as

$$u=\frac{X}{Z}\frac{f}{p_w}+u_0 \text{ and } v=\frac{Y}{Z}\frac{f}{p_h}+v_0. \quad (2)$$

[1] In most of real camera specifications, p_w and p_h parameters are not given. Instead, their relationship is provided, known as aspect ratio (a_r), where $a_r={}^{p_w}/_{p_h}$.

By looking at the problem of projecting the 3D point again, a first version of the k transformation matrix comes up, which takes a 3D point in camera coordinates and returns its 2D image representation in homogeneous coordinates:

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f/p_w & 0 & u_0 \\ 0 & f/p_h & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x/z \\ y/z \\ 1 \end{bmatrix}. \quad (3)$$

If the pixels of the camera are not squared, it is added in the equation a new parameter, also called skew factor [2], that correlates the θ angle between u and v dimensions with its 3D point, turning the affine transformation (3) into:

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f/p_w & -cot(\theta)/p_w & u_0 \\ 0 & f*cosec(\theta)/p_h & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x/z \\ y/z \\ 1 \end{bmatrix}. \quad (4)$$

The final k matrix presented in equation (4) is called intrinsic parameters or calibration matrix because of its dependence on the real camera used to display the scene. By combining the intrinsic and extrinsic parameters matrices we have the camera projection matrix P that is responsible for getting 3D points from the world coordinate system and projecting them onto the camera image plane[3]:

$$\begin{bmatrix} su \\ sv \\ s \end{bmatrix} = \underbrace{\underbrace{\begin{bmatrix} f/p_w & -cot(\theta)/p_w & u_0 \\ 0 & f*cosec(\theta)/p_h & v_0 \\ 0 & 0 & 1 \end{bmatrix}}_{k} * \underbrace{\begin{bmatrix} R_{11} & R_{12} & R_{13} & T_1 \\ R_{21} & R_{22} & R_{23} & T_2 \\ R_{31} & R_{32} & R_{33} & T_3 \end{bmatrix}}_{R|t}}_{P=k[R|t]} * \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}. \quad (5)$$

In order to estimate camera extrinsic parameters for a given frame, some correspondences between 2D points from the image and 3D points from the model are needed. In the following subsections, two classes of methods for pose estimation are described: Pespective-n-Point (PnP) and minimization of reprojection error.

PnP is basically the problem of estimating the camera pose *[R|t]* given n 2D-3D correspondences. The first intuitive approach for solving this problem is to apply the equation $PM_i=sm_i$ to each correspondence i and then solve a linear system. This method is called Direct Linear Transformation (DLT) [11] and can estimate all parameters of P (even if the intrinsic ones are not known). However, when using DLT to calculate P, in most cases, the number of correspondences must be higher than 15, which is more than the necessary when applying other methods and for some techniques is not an acceptable number. Furthermore, the DLT method minimizes an algebraic error, but for the pose estimation problem it is preferable to minimize a geometric error.

[2] Since θ is generally near to 90º, the skew factor *sw* is generally only referenced as *-cot(θ)* and the influence of the *cosec(θ)* term is discarded.

[3] In order to finish the transformation from 3D to 2D points, it is also necessary to normalize the answer in terms of the scale factor s: $[su,sv,s]^t$ to $[u,v,1]^t$.

In many AR applications the intrinsic parameters do not change during the frame sequence since the same camera configuration is used the whole time. So it is preferable to obtain them separately, reducing in a considerable way the number of correspondences needed to estimate the current pose and probably also the estimation error. Encouraged by this context, the PnP problem explicitly uses the intrinsic parameters, which must be previously obtained, and estimates only the extrinsic parameters.

This way, when trying to solve the P3P problem, four solutions are reached. This means that it is not possible to find out a unique solution having only 3 correspondences. An approach to find the correct pose is adding a correspondence and solving the P3P problem for each subset of 3 correspondences; then, a common pose will emerge from the results. Solving P4P and P5P problems usually reaches a unique solution, unless the correspondences are aligned. For $n \geq 6$ the solution is almost always unique.

Several solutions have been proposed for the PnP problem in the Computer Vision and AR communities. In general they attempt to represent the n 3D points in camera coordinates trying to find their depths (which is the distance between the camera optical center C and the point M_i). In most cases this is done using the constraints given by the triangles formed from the 3D points and C. Then *[R|t]* is retrieved by the Euclidean motion (that is an affine transformation whose linear part is an orthogonal transformation) that aligns the coordinates. Reference [12] proposed an iterative, accurate and fast solution that minimizes an error based on collinearity in the object space. Later, the EPnP [13] solution showed a $O(n)$ method for PnP if $n \geq 4$. It represents all points as a weighted sum of four virtual control points. Then the problem is reduced to estimate these control points in the camera coordinate system.

In despite of being able to estimate the pose based solely on the 2D-3D correspondences, PnP methods are sensitive to noise in the measurements, resulting in loss of accuracy. In this scenario, a more adequate approach for calculating the pose is by minimization of the reprojection error. This consists in a non-linear least squares minimization defined by the following equation:

$$[R|t] = \underset{[R|t]}{argmin} \sum_{i=0}^{n} dist^2(\theta(P,M_i),m_i), \quad (6)$$

where: M_i and m_i are correspondent 3D and 2D points in homogeneous coordinates, respectively; θ is the projection function, which takes as arguments the projection matrix P and the 3D point M_i and returns the 2D projected point; *dist* is the Euclidean distance function between 2D points, which is called residual; and *[R|t]* are the extrinsic parameters to be estimated.

Due to the fact that the θ function is non-linear, there is not a closed form solution to equation (6). In this case, an optimization method should be used, such as Gauss-Newton or Levenberg-Marquardt [14]. These methods iteratively refine an estimate of the pose until an optimal result is obtained. The pose increment between consecutive iterations is calculated using the Jacobian matrix of θ. This matrix can be calculated analytically or using differentiation. A requirement for such kind of iterative method is a good initial estimate. Since the difference between consecutive poses is often small, the pose calculated for the previous frame can be used as an estimate for the current frame.

When calculating the pose, few spurious 2D-3D correspondences (named outliers) can ruin estimation even when there are many correct correspondences (named inliers). There are two common methods to decrease the influence of these outliers: RANdom SAmple Concensus (RANSAC) [15] and M-estimators [16].

The RANSAC method is an iterative algorithm that tries to obtain the best pose using a sequence of random small samples of 2D-3D correspondences. The idea is that the probability of having an outlier in a small sample is much lower than when the entire correspondence set is considered.

The algorithm receives basically 4 inputs:

- A set D of 2D-3D correspondences;
- A sample size n, which is a small value (e.g. 6);
- A threshold t, used to classify the correspondences as inliers or outliers. It consists in the maximum value allowed to the return of the *dist* function from equation (6). A commonly used value for t is 2.0.
- A probability Γ of finding a set that generates a good pose. This probability is utilized for calculating the iteration count of the algorithm. This value is usually set to *95%* or *99%*.

RANSAC works in the following way: initially, it is determined a number m of iterations to be executed by the algorithm, e.g. 500. The number of iterations can be decreased during algorithm execution, depending on how good is the pose by that time.

After this, algorithm execution begins. From the D set provided, n correspondences are randomly chosen. From this sample, a pose is calculated using any of the methods previously presented. Next, the other correspondences that were not included in the sample are utilized to verify how good the found pose is. In order to do this, the $dist$ function from equation (6) is applied to the correspondence. If the distance is lower than the t threshold, the correspondence is an inlier. Otherwise, it is an outlier. After all the correspondences are tested, it is verified the percentage w of the correspondences in D that were tagged as inliers. If the current value of w is bigger than any previously obtained percentage, the calculated pose is stored, since it is the most refined by that time.

When a refined pose is found, the algorithm tries to decrease the number of iterations m needed. The idea behind this calculation is very straightforward. Since the n correspondences are sampled independently, the probability that all n correspondences are inliers is w^n. Then, the probability that there is any outlier correspondence is $1\text{-}w^n$. The probability that all the m samples contain an outlier is $(1\text{-}w^n)^m$ and this should be equal to $1\text{-}\Gamma$, resulting in:

$$1-\Gamma=(1-w^n)^m. \tag{7}$$

After taking the logarithm of both sides, the following equation can be obtained:

$$m=\frac{\log(1-\Gamma)}{\log(1-w^n)}. \tag{8}$$

M-estimators are often used together with minimization of reprojection error in order to decrease the influence of outliers. M-estimators apply a function to the residuals that has a Gaussian behavior for small values and a linear or flat behavior for higher values. This way, only the residuals that are lower than a c threshold have an impact on the minimization. A modified version of equation (6) is then used:

$$[R|t]=\underset{[R|t]}{\operatorname{argmin}} \sum_{i=0}^{n}\rho(dist(\theta(P,M_i),m_i)), \tag{9}$$

where ρ is the M-estimator function. Two of the most used M-estimators are Huber and Tukey [16]. The Huber M-estimator is defined by:

$$\rho_{Hub}(x)=\begin{cases}\frac{x^2}{2}, & |x|\le c\\ c\left(|x|-\frac{c}{2}\right), & |x|>c\end{cases}, \tag{10}$$

where c is a threshold that depends on the standard deviation of the estimation error.

The Tukey M-estimator can be computed using the following function:

$$\rho_{Tuk}(x)=\begin{cases}\frac{c^2}{6}\left[1-\left(1-\left(\frac{x}{c}\right)^2\right)^3\right], & |x|\le c\\ \frac{c^2}{6}, & |x|>c\end{cases}. \tag{11}$$

The graphics of the Huber and Tukey M-estimator functions, which can be seen in Fig. 3, highlight how the residuals are weighted according to their magnitude.

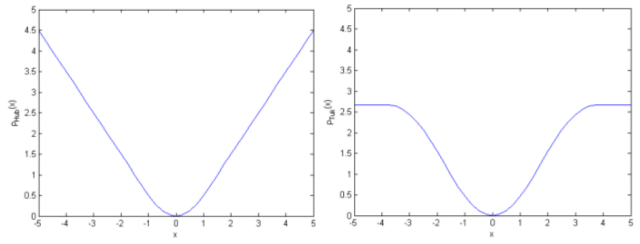


Figure 3. Huber M-estimator function with *c=1* (left) and Tukey M-estimator with *c=4* (right).

III. MARKERLESS AUGMENTED REALITY

Markerless AR systems integrate virtual objects into a 3D real environment in real-time, enhancing user's perception of, and interaction with, the real world. Its basic difference from marker based AR systems is the method used to place virtual objects in the user's view. The markerless approach is not based on the use of traditional artificial markers, which are placed in the real world to support position and orientation tracking by the system. In markerless AR, any part of the real environment may be used as a marker that can be tracked in order to place virtual objects. Therefore, there are no ambient intrusive markers that are not really part of the world. Another advantage is the possibility of extracting from the surroundings characteristic information that may later be used by the markerless AR system for other purposes. Nonetheless, tracking and registration techniques are more complex in markerless AR systems. Another disadvantage emerges in online markerless AR applications since it presents more restrictions.

Techniques developed for online monocular markerless AR can be classified in two major types: model based and Structure from Motion (SfM) based, as described in [17]. With model based techniques, knowledge about the real world is obtained before tracking occurs and is stored in a 3D model that is used for estimating camera pose. In SfM based approaches, camera movement throughout the frames is estimated without any previous knowledge about the scene, being acquired during tracking [18].

Considering their tracking nature, model based techniques can be classified in two categories (Fig. 4): recursive tracking, where the previous pose is utilized as an estimate to calculate the current pose [6][7][19][20]; and tracking by detection, where it is possible to calculate the pose without any previous estimate, allowing automatic initialization and recovery from failures [8][21].

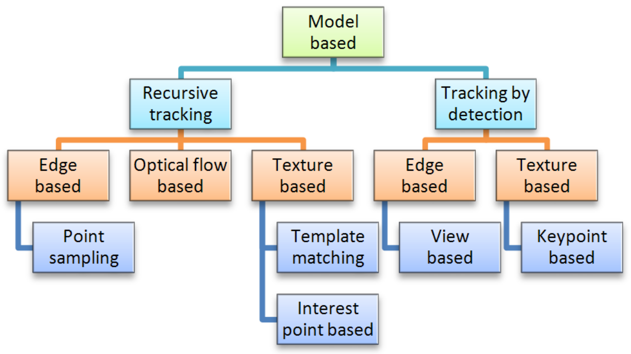


Figure 4. Model based online monocular markerless AR taxonomy.

By taking into account the type of feature used for tracking, model based techniques can also be classified in three other categories: edge based, where camera pose is estimated by matching a wireframe 3D model of an object with the real world image edge information [6][21]; optical flow based, which exploits temporal information extracted from the relative movement of the object projection onto the image in order to track it [19]; and texture based, which takes into account texture information presented in images for tracking [7][8][20].

The edge based recursive tracking category comprises point sampling methods, which sample some control points along the edges of the wireframe 3D model and compare their projections with strong gradients present in the image [6]. Texture based recursive techniques are also classified in two subcategories: template matching, which applies a distortion model to a reference image to recover rigid object movement [20]; and interest point based, which takes into account localized features in the camera pose estimation [7].

Edge based tracking by detection techniques are called view based, since the current frame is matched with 2D views of the target object previously obtained from different positions and orientations [21]. Texture based tracking by detection methods are named keypoint based [8]. Keypoints are features invariant to scale, viewpoint and illumination changes. They are extracted from the object image at every frame, providing 2D-3D correspondences needed for pose estimation.

The presented approaches for model based markerless AR can be analyzed taking into account some relevant metrics. One of the most important metrics is the presence of automatic detection, where user interaction is not required to determine the initial camera pose. When evaluating an AR application, the processing load needed to perform tracking has to be quantified. If the time slice used to estimate camera pose is short, the remaining processing time can be dedicated to other tasks. Accuracy and robustness are the last two metrics considered in the methods analysis. While accuracy is related to the correctness of pose estimation throughout the frames, robustness is about how resistant is the tracker to noise sources. Table 1 compares the model based markerless AR methods introduced in this section, according to the presented criteria. The comparison considers the features that are common to most of the techniques of a given category.

Model based markerless AR approaches may be also analyzed according to their applicability to a specific scenario. Edge based methods are more suitable when tracked objects are polygonal or have strong contours. If objects are textured, optical flow based techniques should be used (in case of constant lighting and not very large camera displacement). If optical flow is not a good option, texture based methods may be the best solution. If the textured object is planar, template matching presents good results with low CPU load; if not, interest point based methods should be used. Tracking by detection techniques suffer from jitter when they estimate each pose based only on current frame information. Taking temporal information into account reduces this problem, but tracking by detection tends to be less accurate than recursive tracking, due to lack of precision on matching. View based techniques are highly accurate, but can cover only a restricted range of rotations and scales of the target object with low detection rates.

IV. MODEL BASED TECHNIQUES

In the following subsections, the model based tracking techniques implemented in this work are detailed.

TABLE I. MODEL BASED METHODS ANALYSIS

Category	Method	Detection	Processing	Accuracy	Robustness
Recursive tracking	Edge based	No	Low	Jitter	Sensible to: • Fast camera movement • Cluttered background
	Optical flow based	No	Low	Cumulative errors	Sensible to: • Fast camera movement • Lighting changes
	Template matching	No	Low	Highly accurate	Sensible to: • Fast camera movement • Lighting changes • Occlusion
	Interest point based	No	High	Accurate	Sensible to: • Fast camera movement
Tracking by detection	View based	Yes	High	Accurate	Restricted range of poses
	Keypoint based	Yes	High	Jitter and drift	No restrictions

Three techniques were contemplated, two of them belonging to the recursive tracking category (point sampling and interest point based) and one of them belonging to the tracking by detection category (keypoint based). Two of the methods are texture based (interest point based and keypoint based), while one is edge based (point sampling). The choice of these techniques for implementation and evaluation in this work is due to the desire of allowing the development of markerless AR systems for different application scenarios. Regarding texture based recursive techniques, the interest point based method was preferred over template matching because the former is capable of tracking fully three dimensional objects, while the later is more suitable for planar objects. Concerning tracking by detection, keypoint based was chosen due to the restriction on the pose range presented by the view based technique.

A. *Point Sampling*

The point sampling (PS) edge based technique described in this paper is based on Wuest et al. [6]. It consists in a recursive technique that uses points sampled from model edges to estimate camera's pose.

This technique starts with an initial pose estimate that will be used in several phases of the algorithm. After that, points are sampled in a balanced way from edges and only visible edges remain to be used in the pipeline. In sequence, sampled visible points are matched with strong gradient image points and this information is used to estimate camera's pose. The pipeline of the PS technique is depicted in Fig. 5.

As an edge based technique, PS is used for polygonal objects tracking but, because of its point sampling aspect, it can also be used for curved objects tracking with small changes. It is also robust to illumination changes, partial occlusion and self-occlusion problems. The following subsections detail each phase of the PS algorithm.

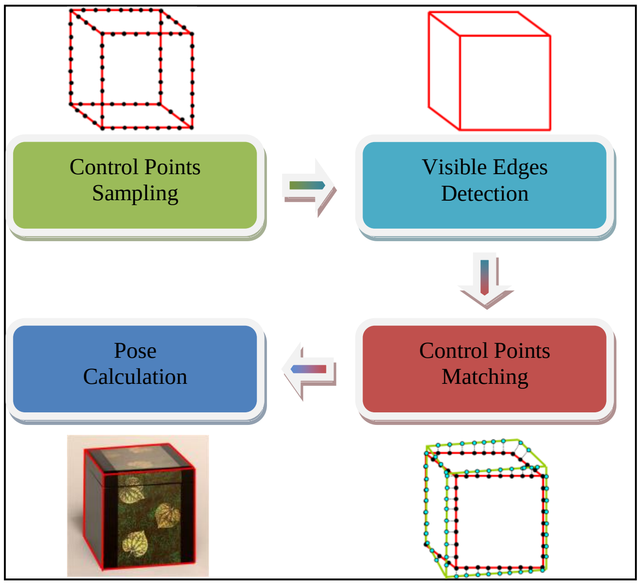


Figure 5. PS technique workflow diagram.

1) Control points sampling: As sampling is a key point of this algorithm, it is important to use a sampling coefficient applied to the projected edges in a way to balance the sampling process. This factor is responsible for allowing sampling from edges while maintaining a fixed amount of points per length unit of projected edges, according to the equation:

$$nPoints = edgeProjSize * sampCoef. \qquad (12)$$

Held this way, sampling points appears uniform, balancing edge's influence according to its projected size. Thus, it is considered the real importance of the points in the projected image and not just the virtual 3D model (see Fig. 6).

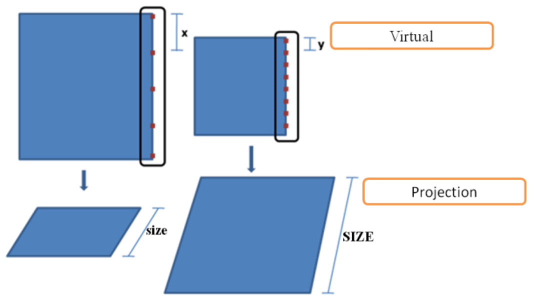


Figure 6. Sampled points distance (x, y) in edges varies to maintain number of points proportional to edges size (size, SIZE).

2) Visible edges detection: Self-occlusion, which occurs when parts of the object occlude parts of itself, is an important aspect in tracking quality. Since some sampled points do not have correspondents in the image because of self-occlusion, they become outliers that will negatively influence the pose estimation.

In order to eliminate outliers originated by self-occlusion, a visibility test can be done in many ways. The approach adopted in [6] for determining the visible parts of the edges at a given frame makes use of an OpenGL extension which is not available at some platforms. Reading from the depth buffer, which could be used with the same purpose, is also not allowed in all platforms. Due to this, an alternative method was developed to perform visibility testing. Inspired by the Facet-ID method described in [7], its goal is to identify edges, and is called the Edge-ID method. In Facet-ID, the index of each polygon is encoded in its color value, and after the model is rendered, it is possible to discover the facet that generated a given pixel when projected. Edge-ID exploits the same idea for edges, but for a different purpose: while Facet-ID is used for finding the 3D back-projection of a pixel and its normal at the model, Edge-ID aims to determine if a control point sampled from an edge is visible or not. Another difference between the methods is that in Facet-ID the model is drawn with filled faces, while in Edge-ID a wireframe model with hidden line removal is rendered. This way, only the visible model edges will have a color value different from the background color. It is then possible to find out if a control point $p(x,y)$ is visible by comparing the index of its edge with the index decoded from the color stored at the position (x,y) in the color buffer. The use of unique IDs for each edge is justified by the fact that points from different edges can be projected to the same position in image space. If no ID checking is performed, a hidden control point could be considered visible. Fig. 7 illustrates the proposed visibility testing approach.

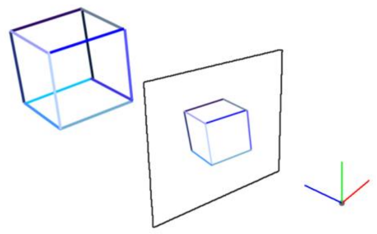


Figure 7. Edge-ID method.

In summary, the outline of the Edge-ID method is as follows:

- Map the color value of each model edge to its index
- Render the model edges with hidden line removal
- For each model edge i
  - Sample the edge, obtaining control points
  - For each sampled point $p(x,y)$
    - If $ID(x,y) = i$, then the point is visible

The default coding scheme adopted for mapping the IDs to RGB color components was rather simple. The color black (R=0, G=0, B=0) is reserved for representing the background. Then, each edge index is incremented by one and, considering its 24-bit binary representation, the most significant byte is stored at the red channel, the next byte is stored at the green channel and the least significant byte is stored at the blue channel. The inverse process is done for decoding. With this representation, the maximum number of model edges is $2^{32} - 1 = 4{,}294{,}967{,}295$. The average edge count of the models commonly used for tracking does not even approach this value.

3) Control points matching: Correlation between object image's points and virtual points is made with the Moving Edges (ME) algorithm [22]. This algorithm makes a search adopting the pipeline described below.

Initially, the edge is projected onto the scene using the previously estimated pose and control points are sampled. After that, the ME algorithm performs a search in the line that passes through the sampled point and is perpendicular to the projected edge in order to find points of strong gradient as matching points (see Fig. 8).

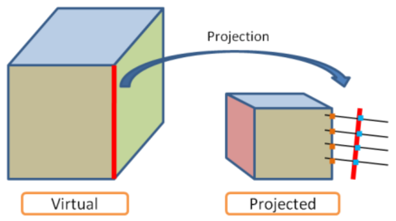


Figure 8. The edge is projected and ME looks for strong gradient points (orange) in a line perpendicular to the edge that passes through the sampled point (blue).

There are two possible ways to compute the point matching result: single hypotheses (SH) model, where the algorithm determines the point of strongest gradient close to the edge as the match to be used in the process of estimating the pose; and multiple hypotheses (MH) model, where a fixed number of strong gradient points is stored to be used as a possible match.

4) Pose calculation: Considering matches found by the ME it is possible to estimate camera's pose by using the LM algorithm with two possible approaches: SH and MH.

In the SH model, reprojection error is minimized by using the control point found in the matching step. MH uses a more balanced approach that employs, at each iteration of LM, the strong gradient point that has minimal distance to the projected virtual point, as can be seen in (13). This addresses the problem of objects with strong gradient contours near the tracked object, minimizing their influence:

$$[R|t] = \underset{[R|t]}{argmin} \sum_{i=0}^{n} dist^2\left(\theta(P, M'_i), mindist\left(m_i^j, P * M'_i\right)\right), \quad (13)$$

where: j strong gradient points are the return from the ME algorithm and mindist chooses the point that is closer to the projected virtual point at each iteration.

B. *Interest Point Based*

The interest point based (IPB) technique described in this paper is based on the work of Vacchetti et al. [7]. It makes use of keyframes in order to perform drift reduction, which are generated prior to the tracking procedure in an offline manner. They are created from images of the target object that have a known camera pose, as illustrated in Fig. 9. In the online tracking phase, local features are extracted from the current frame and matched against the keyframe with a pose that is closer to the current frame. Based on the obtained matches, the current pose can be calculated. The workflow of the online phase of the IPB method is illustrated in Fig. 10. IPB procedures are detailed next.

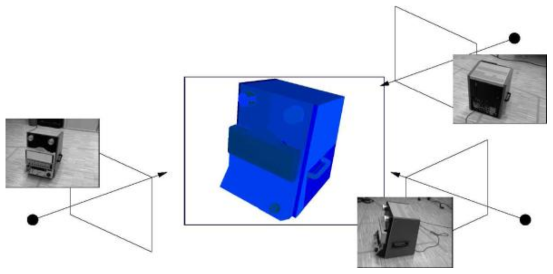
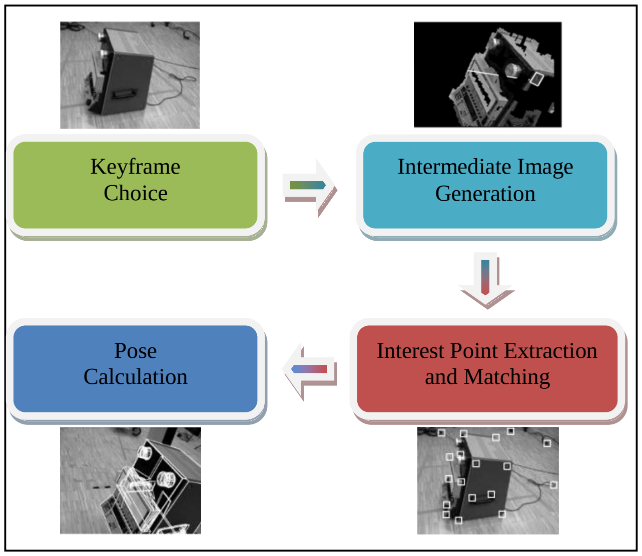


Figure 9. Keyframes generated from three different camera poses.


Keyframe Choice
Intermediate Image Generation
Pose Calculation
Interest Point Extraction and Matching
Figure 10. IPB technique workflow diagram.

1) *Keyframe generation:* Each generated keyframe stores the following information: the image of the object; the corresponding 3D pose for this image; a collection of 2D interest points extracted from the image; the 3D points on the model that correspond to the extracted 2D features; and the normals at the object surface for each 3D point.

The first step in order to generate the keyframe consists in obtaining a frame containing the target object with a known pose. After this, 2D interest points are extracted from the frame using an approach that will be described later. These 2D features are then backprojected in order to obtain the corresponding 3D points and normals in the model. It is possible to discover the facet that generated a given pixel when projected by using the Facet-ID method [7]. Given an interest point $m=(u, v, 1)^t$, the corresponding 3D point M can be obtained by calculating the intersection between the projector line and the generating triangle. The projector line is represented by a ray with origin at the camera optical center C_{opt} and that passes through the interest point m in the projection plane. Ray origin C_{opt} and direction $\vec{r}$ are computed as follows:

$$C_{opt} = -R^t T, \quad (14)$$

$$\vec{r} = (AR)^{-1} m. \quad (15)$$

2) *Keyframe choice:* In the IPB pipeline, it is necessary to discover which keyframe is closer to the current frame. In this work, it was implemented two different approaches to keyframe choice: one using Mahalanobis distance between keyframes poses and current pose; and another using histograms from keyframes and current frame.

In the Mahalanobis approach, it is calculated the distance between all keyframes poses and the last frame pose, since it is a good approximation to the current pose, and the keyframe that has the minimal Mahalanobis distance to the last frame pose is chosen. The Mahalanobis distance D_M between the last frame pose p_f and the keyframe pose p_k is calculated as follows:

$$D_M\left(p_f, p_k\right) = \left(p_f - p_k\right)^t C^{-1} \left(p_f - p_k\right), \quad (16)$$

where C is the covariance matrix of the keyframes poses, which is given by:

$$C = \frac{1}{n-1} \sum_{i=1}^{n} \left(p_{k_i} - \overline{p_k}\right)\left(p_{k_i} - \overline{p_k}\right)^t, \quad (17)$$

where $\overline{p_k}$ is the average pose of the keyframes.

In the histogram approach, the concept of using the last pose as an approximation of the current pose is also exploited. All keyframes poses and the last calculated pose are used to backproject the model using the Facet-ID algorithm. After that, histograms are generated from each of these images and the closest keyframe is the one whose histogram has the least difference relative to the histogram of the current frame.

3) Intermediate image generation: The poses of the chosen keyframe and the current frame may be not close enough to allow the matching of their interest points. Due to this, an intermediate synthetic image is generated from the keyframe image with a pose near to the one of the current frame, as can be seen in Fig. 11. In order to perform this, a patch around each interest point of the keyframe image is transferred to the intermediate image by applying a homography. Given an estimation of the current projection matrix $P=A(R|T)$, a keyframe with projection matrix $P_K=A(R_K|T_K)$ and a plane π approximated by the patch in the object surface with normal $\vec{n}$ and distance to the origin d, the homography H is obtained by:

$$H=A(\delta R-\delta T\vec{n}'^{t}/d')A^{-1}, \tag{18}$$

$$\text{where } \delta R=RR_K^t, \tag{19}$$

$$\delta T=-RR_K^tT_K+T, \tag{20}$$

$$\vec{n}'=R_K\vec{n}, \tag{21}$$

$$\text{and } d'=d-T_K^t(R_K\vec{n}). \tag{22}$$

A point m in the keyframe image is then transferred to a point m' in the intermediate image by $m'=Hm$. A patch size of 10x10 pixels was empirically chosen.

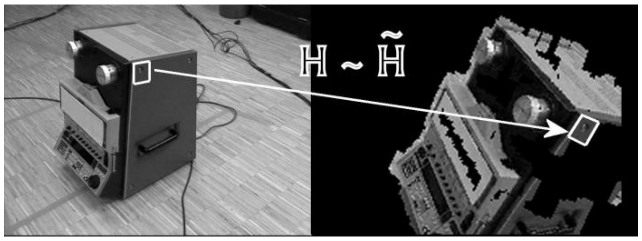


Figure 11. Intermediate image (right) generated from a keyframe (left).

4) Interest point extraction and matching: The Harris corner detector [23] was applied to extract interest points from the images. Next, the interest points from the current frame are matched against the ones from the intermediate image using the method proposed by Zhang et al. [24]. In this method, the similarity level between an interest point $m_1(x_1,y_1)$ from image I_1 and an interest point $m_2(x_2,y_2)$ from image I_2 is determined by their normalized cross correlation, which is given by:

$$S(m_1,m_2)=\frac{\sum_{i=-n}^{n}\sum_{j=-n}^{n} I_1(x_1+i,y_1+j)\cdot I_2(x_2+i,y_2+j)}{\sqrt{\sum_{i=-n}^{n}\sum_{j=-n}^{n} I_1(x_1+i,y_1+j)^2\cdot I_2(x_2+i,y_2+j)^2}}. \tag{23}$$

According to our experiments, a value of 7 for the window size n was found to be sufficient. For each interest point m_1 of I_1, it is calculated the similarity level with the nearby interest points of I_2. A neighborhood size of 50x50 was used. The interest point from I_2 with the highest similarity level (m_2^{max}) is kept as a match candidate for m_1. The procedure is then repeated with the roles of I_1 and I_2 inversed. After this, the interest points that are mutually pointed out as match candidates are retained as matches.

5) Pose calculation: Once matching points between intermediate image and current frame were found, correspondences between 3D keyframe's points and 2D image's points are transitively obtained. The system is then allowed to estimate camera pose correctly by using these correspondences together with the LM algorithm, as discussed in Section II. In order to minimize outliers influence, it is also used the Tukey M-estimator.

C. Keypoint Based

The keypoint based (KB) technique described in this paper is based on the work of Skrypnyk et al. [8]. As IPB, an offline training phase is needed in order to acquire knowledge about the object to be tracked. A set of 2D object features that are invariant to scale, illumination and viewpoint are obtained, together with their corresponding 3D position in the object model. At runtime, invariant features are extracted from the current frame and matched with the acquired knowledge base, resulting in 2D-3D correspondences that enable the computation of the current pose without any previous estimate. The pipeline of the online phase of the KB technique is shown in Fig. 12. Each step involved in the KB technique is described next.

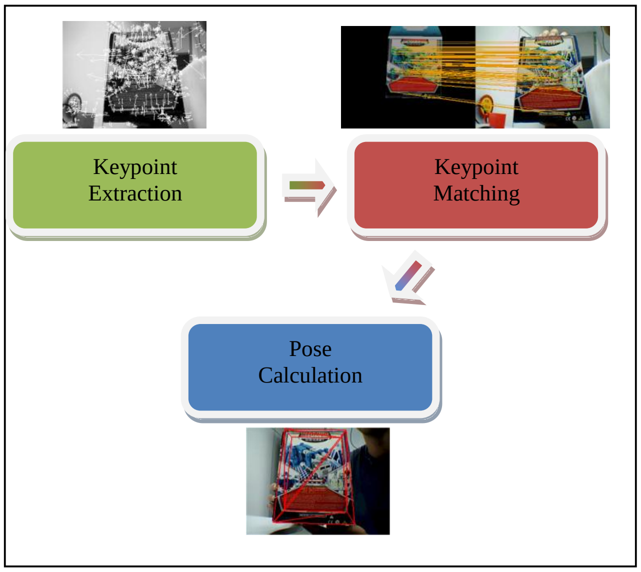


Figure 12. KB technique workflow diagram.

1) Keypoint extraction: Some points on objects textures can have a number of associated characteristics that make them, in some way, unique. These special points are called keypoints and for each keypoint these associated characteristics are stored into a high dimensional descriptor (e.g. the Scale-Invariant Feature Transform (SIFT) [8] keypoint descriptor has a length of 128).

There are some algorithms for extracting keypoints from an image. Two of them have been used in this work: in a first moment SIFT was utilized and as a second approach Speeded Up Robust Features (SURF) [25] was used attempting to obtain better results.

2) Offline training phase: First of all, it is needed some information about the object to be tracked. This offline phase may be basically summarized in the following steps:

- Various keyframes of the model are obtained from a synthetic scene. These frames must show each face of the object; if a face is not represented by a keyframe, this face cannot be recognized in the tracking phase.
- For each keyframe, a camera pose is associated. As the keyframes have been obtained from a synthetic scene, it is easy to obtain the camera pose and there is no doubt about its correctness.
- In each keyframe, keypoints are extracted using one of the algorithms cited above. A relation between keypoint and keyframe pose is stored to be used in future initializations.
- For each keypoint, its 3D correspondence in the model is found using the camera pose and the Facet-ID algorithm previously described.
- A kd-tree of keypoints is constructed using their descriptors [8]. This tree will reduce a lot the searching time of keypoints matches.

At the end of this pipeline a set of keypoints with their 3D correspondences are arranged into a kd-tree and now the new keypoints obtained in the tracking phase can be matched with the offline data.

3) Keypoint matching: Starting the online tracking pipeline, the first step is to extract keypoints from the current frame. This extraction must use the same method chosen in the offline phase (e.g. SIFT).

Then, for each extracted keypoint, it is executed a Best Bin First search [8] in the kd-tree to find its match (if it has one). This search returns the two nearest neighbours of the keypoint.

By verifying the Euclidian distance between the neighbours and the extracted feature descriptors it can be checked if there is an error in the search. The closer are the distances the more probable is that it is an error case, because given the cardinality of the descriptors it is very improbable that they are similar. Then, a ratio threshold is used to discard these cases. If the distances ratio is lower than the threshold, the nearest neighbour and the extracted keypoint are considered as a match case and establish a 3D correspondence to the new keypoint. The ratio threshold used was 0.5. Fig. 13 shows the current frame keypoints matched with some keypoints of a keyframe.

After analyzing all extracted keypoints there will be a given number of matches, but if this number is lower than a predefined threshold, it is considered that a tracking failure has occurred. It has been empirically verified that when the number of matches falls below 5 the results showed to be not acceptable, so 5 has been used as the threshold that detects the failures.

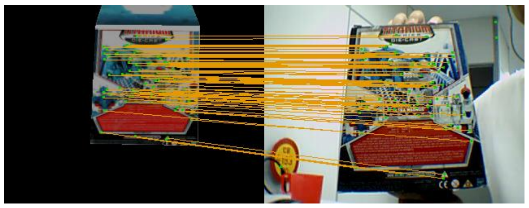


Figure 13. Keypoint matching.

4) Pose calculation: Once the set of matches has been obtained, the next step is to estimate a camera pose for the current frame. For this it was utilized RANSAC together with two methods for pose hypothesis generation: minimization of reprojection error using LM and the Lu PnP algorithm (see Section II).

The reprojection is done using equation (6), with the addition of a confidence factor that is inversely proportional to the keypoint scale. When using LM, an initial estimation is needed that will be converted into a new pose. During the tracking, the last pose obtained is used to solve this, but if the current frame is the first one there is no previous pose to be used. The same occurs when there is a tracking failure, because the last pose obtained may not be reliable. In these cases, the initial estimation used is the pose of the keyframe that contributes with the most number of matches of the current frame.

V. Results

The implemented model based techniques have been evaluated taking into account frame rate and accuracy metrics. As explained in Section III, the implemented techniques differ by the type of object that is more suitable of being tracked using them. Due to this, different objects and scene sequences were used in the evaluation of each method. As the techniques have different purposes, a direct comparison between them is not always possible or desirable.

The desktop computer used to perform the tests has an AMD Athlon 64 3200+ processor, 1 GB of RAM, a NVIDIA GeForce 8800 GTX graphics board with 768 MB of memory and a screen resolution of 1280 x 1024 pixels. The A4Tech ViewCam PK-635 camera was used, with a resolution of 320x240 pixels and a frame rate of 30 fps. The operating system is Microsoft Windows XP Professional SP3. The development tool utilized was Microsoft Visual Studio .NET 2005 Professional Edition. The VXL [26] library provided most of the math and computer vision support required. The ViSP library [27] was also used, since it contains an implementation of the ME algorithm, which had to be modified to support MH. Feature extraction for KB was done using SIFT GPU [28] and the official SURF implementation [25]. Keypoint matching was done using Rob Hess' implementation [29]. It was also used the implementation of the Lu PnP technique available on the ARToolKitPlus library [30]. OpenGL was utilized for 3D graphics rendering. All the images used in the test sequences have QVGA resolution (320 x 240 pixels).

A. Point Sampling

The PS technique was evaluated using SH and MH approaches with real (cube) and synthetic (building) sequences. Fig. 14 shows examples of real and synthetic sequences tracking using a wireframe model to illustrate the correctness of this technique.

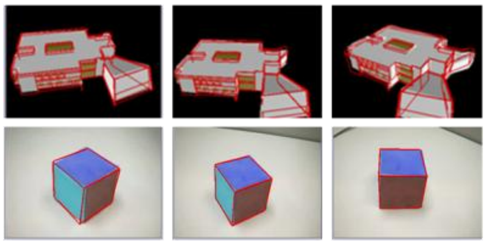


Figure 14. PS tracking of synthetic (top) and real (bottom) sequences.

Fig. 15 illustrates tracking instability generated from high gradient object's approximation in the SH approach against MH stability using 5 hypotheses in the pose calculation step.

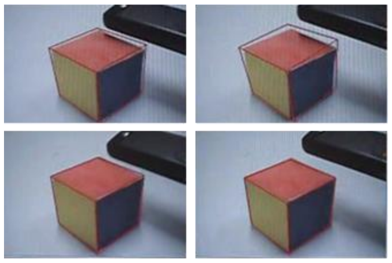


Figure 15. SH instability (top) and MH stability (bottom).

Table 2 and Fig. 16 present a performance evaluation using both synthetic and real sequences with SH and MH approaches. The table presents timing for all computational relevant steps of the tracking algorithm and the figure shows the total computational time against average times for all tests. As expected, MH has presented more stability than SH in despite of a worse average speed of 133.1 ms and 30.19 ms (MH) against 61.44 ms and 18.57 ms (SH). The obtained frame rates are suitable to AR applications. The performance difference between synthetic and real data sequences is basically because of the complexity of the tracked objects, greater in the synthetic sequence (703 faces) than in the real one (6 faces). The bottleneck was the pose calculation step.

TABLE II. COMPARISON TABLE OF TIMES AND # OF MATCHES FOR THE PS TRACKING ALGORITHM

		Synth SH	Synth MH	Real SH	Real MH
Time (ms)	***Visibility test***	17.72	18.73	10.71	11.01
	ME	21.07	22.68	5.70	7.75
	Pose calculation	22.65	91.69	2.16	11.43
# of matches		1537	6964	268	1077

Fig. 17 shows tracking precision by comparing camera's centers calculated by PS using SH and MH with ground truth camera's centers obtained in the synthetic sequence generation. The MH approach had an average error of ~2 mm while SH showed an error of ~3 mm, evidencing quality and robustness improvement due to the addition of MH. Compared to the object length and distance to camera (~70 mm both), SH and MH errors were small and acceptable according to AR techniques needs.

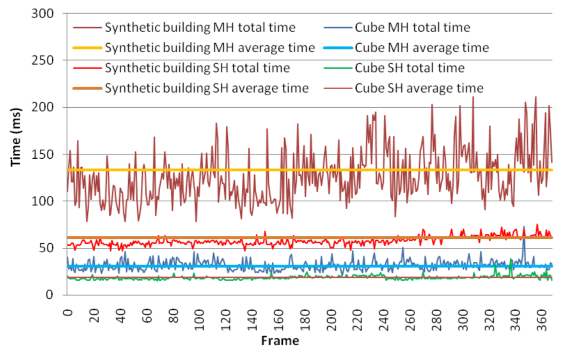


Figure 16. PS computation times using synthetic/real sequences and SH/MH.

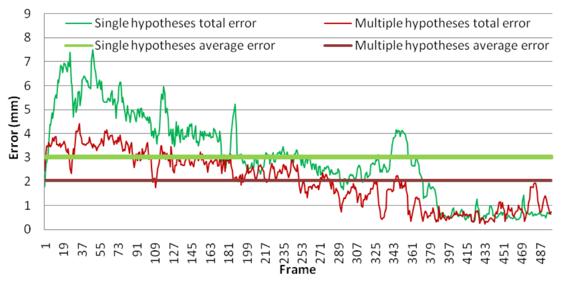


Figure 17. PS tracking precision for the synthetic sequence using SH and MH.

B. Interest Point Based

The IPB technique was evaluated using synthetic and real data. Fig. 18 shows some pose estimation results for the "cube" synthetic sequence and the "coffee box" real sequence. In the cube sequence, 11 keyframes were used, while 8 keyframes were used in the coffee box sequence. In both cases, the objects were augmented with their wireframe model, in order to show if the tracking results are visually acceptable.

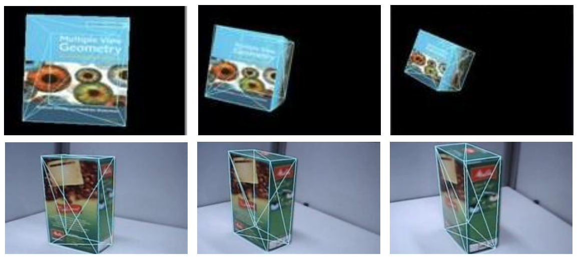


Figure 18. IPB tracking results for a synthetic sequence (top) and a real sequence (bottom).

Table 3 presents the average time required by each step of the tracking algorithm using both sequences mentioned above as input. The Mahalanobis keyframe choice method was faster than the histogram one. Considering the worst case for keyframe choice (histogram), the average total times spent for tracking a frame were 50 ms for the cube sequence (resulting in a 20 fps rate) and 90 ms for the coffee box sequence (resulting in a 11 fps rate). Figure 19 shows the total times spent for tracking each of the first 250 frames of both sequences. The obtained frame rate is adequate to AR applications. Nevertheless, some optimization can still be done, especially regarding the feature extraction and matching phase, which has shown to be the bottleneck of the technique.

TABLE III. COMPARISON TABLE OF TIMES AND # OF MATCHES FOR THE IPB TRACKING ALGORITHM

			Cube	Coffee box
Time (ms)	***Keyframe choice***	***Mahalanobis***	0.46	0.46
		Histogram	2.11	1.66
	Intermediate image generation		2.60	4.46
	Feature extraction and matching		43.18	79.54
	Pose calculation		2.20	3.45
# of matches			52	96

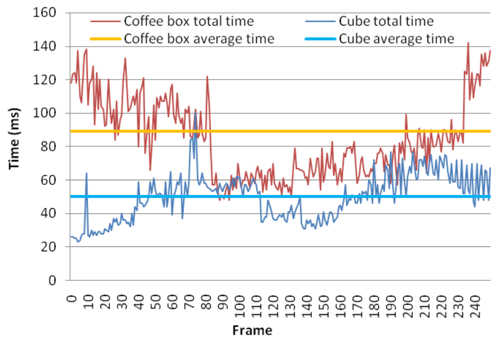


Figure 19. IPB total computation times for each of the first 250 frames of the sequences.

The tracking error for the cube synthetic sequence is presented in Fig. 20. The distance between the tracked object and the camera ranged between 200 and 600 mm. The side length of the cube was 100 mm. Different keyframe choice methods were used. The average errors were 2.80 mm when using Mahalanobis and 3.44 mm when using histogram. The histogram tracking error presented some peaks that influenced the total average error, but showed to be more stable than Mahalanobis during most of the sequence.

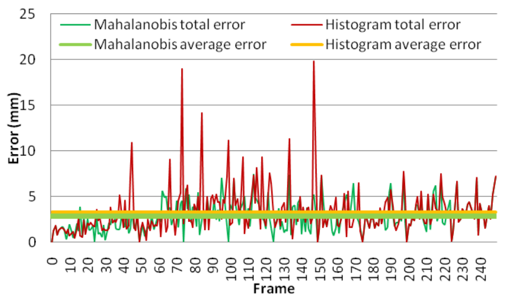


Figure 20. IPB estimation accuracy for the cube synthetic sequence.

Fig. 21 shows tracking accuracy results for the coffee box real sequence considering both keyframe choice methods. The camera positions calculated by the tracking algorithm in the x axis are compared with ground truth values provided by the keyframes. When using the histogram method, poses calculated by the tracker followed the keyframes throughout the sequence, while using the Mahalanobis method resulted in a tracking failure around frame 70.

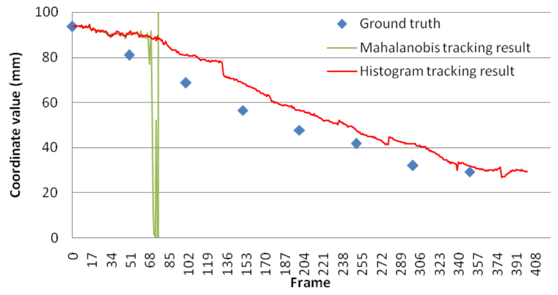


Figure 21. IPB tracking results of the camera x axis for the coffee box real sequence.

C. Keypoint Based

KB has been tested using a book (planar) and a box (non-planar) as target objects. In Fig. 22, some successfully recovered poses are presented for both, planar and non-planar objects. For most of the cases the result showed to be acceptable.

The subsequent results are related to the box object. In the offline training phase, 14 keyframes have been taken from distinct angles covering all box faces. It was obtained 2,026 SIFT keypoints resulted from these 14 images. Using the same keyframes, 1,393 SURF keypoints were extracted. The test was done with both a synthetic and a real scene. The synthetic scene was a sequence of 250 images of the box in various angles and distances. The real sequence had 1 minute of duration and also covers all box faces.

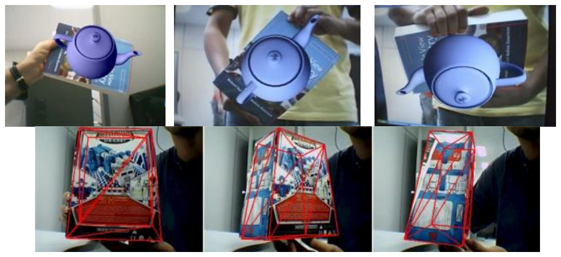


Figure 22. KB results using SIFT and LM with planar (top) and non-planar (bottom) objects.

Using SIFT there was less than 1% of tracking failures in both sequences. With SURF (and LM), in 48% of the real and 22% of the synthetic sequence frames tracking failures occurred. This can be easily understood by looking at Table 4 and seeing the average number of matches obtained with SURF.

The real sequence showed similar results, however some behaviors that were not present in the synthetic scene emerged in this case. For example, it was observed that when the specular component was predominant, a tracking failure occurred. This happened due to the fact that not enough keypoints could be extracted from the scene, since the surface of the box looses a lot of its texture characteristics.

TABLE IV. COMPARISON TABLE OF TIMES AND # OF MATCHES FOR THE KB ALGORITHM

		Synthetic		Real	
		SURF	***SIFT***	***SURF***	***SIFT***
Time (ms)	***Extraction***	39	39	77	51
	Matching	12	39	26	105
	Lu	6	10	7	8
	LM	122	106	113	127
# of matches		13	71	7	76

On the other hand, as an advantage SURF showed to be faster than SIFT (by analyzing extraction plus matching times). The same occurs with Lu in relation to LM. Indeed, the fastest fps rate obtained was 17 in the synthetic scene and 9 in the real one using SURF and Lu together.

The precision of the pose recovered by LM was better than with Lu. This occurred, in part, due to the fact that when only one face of the object detained the greatest part of the matched keypoints, the Lu method inverted the shown face (Fig. 23). LM however did not fall in this problem since the new pose results strongly depended of an initial estimation, so the real pose was always more probable to be obtained than the inverted one.

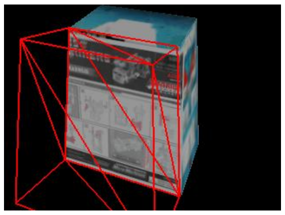


Figure 23. Lu method recovering an inverted pose.

The Lu error is reflected in Fig. 24: when the pose is inverted, the distance between the correct camera center and the recovered one is much higher than usual. SIFT average error with LM was 4.6 mm and with Lu was 27.6 mm, while using SURF combined with LM was 22.4 mm and with Lu was 82.8 mm.

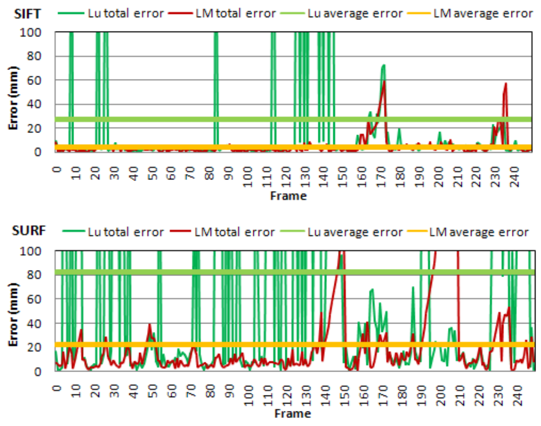


Figure 24. Accuracy of Lu and LM methods with SIFT (top) and SURF (bottom) keypoints.

VI. CONCLUSIONS AND FUTURE WORK

In this work, three model based tracking techniques have been implemented, allowing the development of markerless AR systems for different application scenarios. Analyzing the results, it was possible to confirm the features listed in Section III regarding the markerless tracking techniques surveyed in this work. Although the performance and accuracy results obtained were satisfactory in many of the tests, some improvements can still be done. Even though QVGA images were used in the evaluation of the implemented methods, it has been shown that computer vision algorithms can handle high definition images by using massively parallel approaches, such as GPGPU [31]. Direct comparisons between the results obtained in this work and those presented in related works were not done due to the fact that the implementations and datasets used by such works are not available for free access.

As a way to improve precision, the history of temporal information accumulated during tracking can be exploited in order to avoid jittering [7]. Another topic of future investigation resides in combining different techniques as a way to improve precision and robustness [32].

ACKNOWLEDGMENT

The authors would like to thank LCCV UFAL for kindly providing the building model for the tests.

REFERENCES

[1] M. Haller, M. Billinghurst, and T. Bruce, Emerging Technologies of Augmented Reality: Interfaces and Design, 1st ed., Hershey: Idea Group Publishing, 2007.

[2] J. Ferwerda "Three varieties of realism in computer graphics," Proc. SPIE Human Vision and Electronics Imaging, pp. 290-297, 2003.

[3] R. Azuma "A survey of augmented reality," Presence: Teleoperators and Virtual Environ., vol. 6, n. 4, pp. 355-385, 1997.

[4] J. Lima, S. Gomes Neto, M. Bueno, V. Teichrieb, J. Kelner, and I. Santos, "Applications in engineering using augmented reality technology," Proc. CILAMCE, 13 p., 2008.

[5] J. Platonov, H. Heibel, P. Meier, and B. Grollmann, "A mobile markerless AR system for maintenance and repair," Proc. ISMAR, pp. 105-108, 2006.

[6] H.Wuest, F. Vial, and D. Stricker, "Adaptive line tracking with multiple hypotheses for augmented reality," Proc. ISMAR, pp. 62-69, 2005.

[7] L. Vacchetti, V. Lepetit, and P. Fua, "Stable real-time 3D tracking using online and offline information," IEEE Trans. Pattern Anal. Mach. Intell., vol. 26, n. 10, pp. 1385-1391, 2004.

[8] I. Skrypnyk, and D. Lowe, "Scene modelling, recognition and tracking with invariant image features," Proc. ISMAR, pp. 110-119, 2004.

[9] D. Forsyth, and J. Ponce, Computer Vision - A Modern Approach, 1st ed., New Jersey: Prentice-Hall, 2002.

[10] R. Brockett, "Robotic manipulators and the product of exponentials formula," Proc. MTNS, pp. 120-127, 1984.

[11] O. Faugeras, Three-Dimensional Computer Vision: A Geometric Viewpoint, Cambridge: MIT Press, 1993.

[12] C. Lu, G. Hager, and E. Mjolsness, "Fast and globally convergent pose estimation from video images," IEEE Trans. Pattern Anal. Mach. Intell., vol. 22, n. 6, pp. 610-622, 2000.

[13] F. Moreno-Noguer, V. Lepetit, and P. Fua, "Accurate non-iterative O(n) solution to the PnP problem," Proc. ICCV, 8 p., 2007.

[14] B. Triggs, P. McLauchlan, R. Hartley, and A. Fitzgibbon, "Bundle Adjustment – A Modern Synthesis," in Vision Algorithms: Theory and Practice, B. Triggs, A. Zisserman and R. Szeliski, Eds. Berlim: Springer, 2000, pp. 298-372.

[15] M. Fischler, and R. Bolles, “Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography,” Commun. ACM, vol. 24, n. 6, pp. 381-395, 1981.

[16] V. Lepetit, and P. Fua, “Monocular model-based 3D tracking of rigid objects: a survey,” Found. Trends Comput. Graph. Vis., vol. 1, n. 1, pp. 1-89, 2004.

[17] V. Teichrieb, J. Lima. E. Apolinário, T. Farias, M. Bueno, J. Kelner, and I. Santos, “A survey of online monocular markerless augmented reality,” Int. J. Model. Simul. Pet. Ind., vol. 1, n. 1, pp. 1-7, 2007.

[18] S. Gomes Neto, M. Bueno, T. Farias, J. Lima, V. Teichrieb, J. Kelner, and I. Santos, “Experiences on the implementation of a 3D reconstruction pipeline,” Int. J. Model. Simul. Pet. Ind., vol. 2, n. 1, pp. 7-15, 2008.

[19] S. Basu, I. Essa, and A. Pentland, “Motion regularization for model-based head tracking,” Proc. ICPR, pp. 611-616, 1996.

[20] F. Jurie, and M. Dhome, “A simple and efficient template matching algorithm,” Proc. ICCV, pp. 544-549, 2001.

[21] C. Wiedemann, M. Ulrich, and C. Steger, “Recognition and tracking of 3D objects,” Lect. Notes Comput. Sci., vol. 5096, pp. 132-141, 2008.

[22] P. Bouthemy, “A maximum likelihood framework for determining moving edges,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 11, n. 5, pp. 499-511, 1989.

[23] C. Harris, and M. Stephens, “A combined corner and edge detector,” Proc. AVC, pp. 147-151, 1988.

[24] Z. Zhang, R. Deriche, O. Faugeras, and Q. Luong, “A robust technique for matching two uncalibrated images through the recovery of the unknown epipolar geometry,” Artif. Intell., vol. 78, n. 1, pp. 87-119, 1995.

[25] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, “SURF: Speeded Up Robust Features,” Comput. Vis. Image Underst., vol. 110, n. 3, pp. 346-359, 2008.

[26] VXL - C++ libraries for computer vision, http://vxl.sourceforge.net, 2009.

[27] E. Marchand, F. Spindler, and F. Chaumette, “ViSP for visual servoing: a generic software platform with a wide class of robot control skills,” IEEE Robot. Autom. Mag., vol. 12, n. 4, pp. 40-52, 2005.

[28] S. Sinha, J.-M. Frahm, M. Pollefeys and Y. Genc, “GPU-based video feature tracking and matching,” Proc. EDGE, 2 p, 2006.

[29] SIFT Feature Detector – RobHess, http://web.engr.oregonstate.edu/~hess, 2009.

[30] D. Wagner, and D. Schmalstieg, “ARToolKitPlus for pose tracking on mobile devices”, Proc. Comput. Vis. Winter Workshop, 8 p, 2007.

[31] T. Farias, J. Teixeira, G. Almeida, P. Leite, V. Teichrieb, and J. Kelner, “A CUDA-enabled KLT tracker for high definition images”, Proc. Symp. Virtual Augment. Real., 9 p, 2009.

[32] L. Vacchetti, V. Lepetit, and P. Fua, “Combining edge and texture information for real-time accurate 3d camera tracking,” Proc. ISMAR, pp. 48-57, 2004.