Tracking Chessboard Corners Using Projective Transformation for Augmented Reality

S. Malek s_malek@cdta.dz
Advanced Technologies Development Center (CDTA)
Algiers, 16005. Algeria

Zenati-Henda nzenati@cdta.dz
Advanced Technologies Development Center (CDTA)
Algiers, 16005. Algeria

M.Belhocine mbelhocine@cdta.dz
Advanced Technologies Development Center (CDTA)
Algiers, 16005. Algeria

S.Benbelkacem sbenbelkacem@cdta.dz
Advanced Technologies Development Center (CDTA)
Algiers, 16005. Algeria

Abstract

Augmented reality has been a topic of intense research for several years for many applications. It consists of inserting a virtual object into a real scene. The virtual object must be accurately positioned in a desired place. Some measurements (calibration) are thus required and a set of correspondences between points on the calibration target and the camera images must be found. In this paper, we present a tracking technique based on both detection of Chessboard corners and a least squares method; the objective is to estimate the perspective transformation matrix for the current view of the camera. This technique does not require any information or computation of the camera parameters; it can used in real time without any initialization and the user can change the camera focal without any fear of losing alignment between real and virtual object.

Keywords: Pinhole Model, Least Squares Method, Augmented Reality, Chessboard Corners Detection.

1. INTRODUCTION

The term Augmented Reality (AR) is used to describe systems that blend computer generated virtual objects with real environments. In real-time systems, AR is a result of mixing live video from a camera with computer-generated graphical objects that are recorded in a user's three-dimensional environment [1]. This augmentation may include labels (text), 3D rendered models, or shading and illumination variations.
In order for AR to be effective, the real and computer-generated objects must be accurately positioned relative to each other. This implies that certain measurements or calibrations need to be made initially.

Camera calibration is the first step toward computational computer vision. The process of camera calibration determines the intrinsic parameters (internal characteristics of camera) and/or extrinsic parameters of the camera (camera position related to a fixed 3D frame) from correspondences between points in the real world (3D) and their projection points (2D) in one or more images.

There are different methods used to estimate the parameters of the camera model. They are classified in three groups of techniques:

* *Non linear optimization techniques*: the camera parameters are obtained through iteration with the constraint of minimizing a determined function. These techniques are used in many works [2][3]. There advantage is that almost any model can be calibrated and accuracy usually increases by increasing the number of iterations. However, these techniques require a good initial guess in order to guarantee convergence. Surf

* *Linear techniques which compute the transformation matrix*: due to the slowness and computational burden of the first techniques, closed-form solutions have also been suggested. These techniques use the least squares method to obtain a transformation matrix which relates 3D points with their projections [4][5].

* *Two-steps techniques*: these approaches consider a linear estimation of some parameters while others are iteratively estimated.

To ensure tracking of the virtual object, the camera position must be estimated for each view. Several techniques can be used. They are classified in two groups: vision-based tracking techniques and sensor-based tracking techniques [6].

Most of the available vision-based tracking techniques are divided into two classes: feature-based and model-based. The principle of the feature-based methods is to find a correspondence between 2D image features and their 3D world frame coordinates. The camera pose can then be found from projecting the 3D coordinates of the feature into the observed 2D image coordinates and minimizing the distance to their corresponding 2D features [7]. The features extracted are often used to construct models and use them in model-based methods; edges are often the most used features as they are computationally efficient to find and robust to changes in lighting [6].
The sensor-based techniques are based on measurements provided by different types of sensors: magnetic, acoustic, inertial, GPS ... In augmented reality, they are mainly combined with techniques of the first group (hybrid tracking). These methods have the advantage of being robust to abrupt movements of the camera, but they have the disadvantage of being sensitive to environmental disturbances or a limited range in a small volume.

Recently, there is another classification more used to distinguish between vision-based tracking techniques: fiducial marker tracking and markerless tracking [8]. In the first one, the fiducial marker is surrounded by a black rectangle or circle shape boundary for easy detection. Markerless techniques are the other vision-based tracking techniques which don’t use fiducial marker.

The common thing between augmented reality applications is the necessity to calibrate the camera at first; if the user wants to change the camera, the resolution or the focal length (zoom lens), he must then calibrate his camera again before starting working with it.

This paper introduces a technique to model a camera using a robust method to find the correspondences without any need of calibration. We use the least squares method to estimate the Perspective Transformation Matrix. For tracking, we use chessboard corners as features to track; these corners are extracted from each video image and used to estimate the Perspective Transformation Matrix. This system does not need any camera parameters information (intrinsic and extrinsic parameters which are included in the Perspective Transformation Matrix). The only requirement is the ability to track the chessboard corners across video images.

The rest of the paper is organized as follows. Section 2 describes related work. Section 3 discusses and evaluates the proposed approach of tracking using a chessboard pattern. In section 4, we present how virtual objects are inserted into real scenes and we give some results. Finally, in the last section, conclusions and discuss of results are given.

2. RELATED WORKS

There are many works which use fiducial marker for tracking. ARToolKit [9][10] is one of the most popular library for augmented reality, it use a rectangular shape which contains an arbitrary grayscale patterns. When the marker is detected, its pattern is extracted and cross-correlated with all known patterns stocked on its data-bases. the camera pose is estimated by using the projective relation between the vertices of the fiducial marker in the scene and the world. ArtoolKit has the inconvenient to be more slowly when using several pattern and markers. However, later research suggested to apply digital communication coding techniques to improve the system's performance, at the cost of customization: QR-code [11], TRIP [12], Artag [13], Cantag [14].

In markerless tracking, the principle is to match features extracted from scenes taken from different viewpoints. Lowe [15] presents a new method named SIFT (Scale Invariant Feature Transform) for extracting distinctive invariant features from images. These features can be used to perform reliable matching between different viewpoints; they are invariant to image scale and rotation, and present robust matching when changing in illumination or adding of noise. Bay et al. [16] present a novel scale- and rotation-invariant interest point detector and descriptor, named SURF (Speeded Up Robust Features). It approximates and also outperforms previously proposed methods to get more robustness and to be much faster.

Recently, Lee and Höllerer [17] extract distinctive image features of the scene and track them frame-by-frame by computing optical flow. To avoid problem of consuming time of processing and achieve real-time performance, multiple operations are processed in a synchronized multithreaded manner.

3. IMPLEMENTATION

We will present here the camera model and the tracking method used for augmented reality.

3.1 Camera Model

The model is a mathematical formulation which approximates the behavior of any physical device, e.g. a camera. There are several camera models depending on the desired accuracy. The simplest is the pinhole Model.

In an AR system, it is necessary to know the relationship between the 3D object coordinates and the image coordinates. The pinhole model defines the basic projective imaging geometry with which 3D objects are projected onto a 2D image plane.
This is an ideal model commonly used in computer graphics and computer vision to capture the imaging geometry. The transformation that maps the 3D world points into the 2D image coordinates is characterized by the next expression:

$$\begin{pmatrix} su \\ sv \\ s \end{pmatrix} = \begin{pmatrix} \alpha_u & \gamma & u_0 & 1 \\ 0 & \alpha_v & v_0 & 1 \\ 0 & 0 & 1 & 1 \end{pmatrix} \begin{pmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \qquad (1)$$

Equation (1) can be simplified to:

$$s\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = A.R.t\begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \qquad (2)$$

With: $[u,v,1]^T$ represents a 2D point position in Pixel coordinates. $[X,Y,Z]^T$ represents a 3D point position in World coordinates. t describes the translation between the two frames (camera frame and world frame), and R is a 3x3 orthonormal rotation matrix which can be defined by the three Euler angles and A is the intrinsic matrix containing 5 intrinsic parameters.
The product *A.R.t* represents the "Perspective Transformation Matrix" M, with:

$$M = A.R.t = \begin{pmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \end{pmatrix} \qquad (3)$$

From equations (2) and (3), we get:

$$\begin{cases} u = \dfrac{m_{11}X + m_{12}Y + m_{13}Z + m_{14}}{m_{31}X + m_{32}Y + m_{33}Z + m_{34}} \\ v = \dfrac{m_{21}X + m_{22}Y + m_{23}Z + m_{24}}{m_{31}X + m_{32}Y + m_{33}Z + m_{34}} \end{cases} \qquad (4)$$

Each 3D point gives two equations. Six points are then sufficient to estimate the 12 coefficients of the matrix M. But it is possible to use more than 6 points to get better precision. To calculate the different parameters, the constraint $m_{34} = 1$ is used.
To solve the system (4), we first transform it into a linear system as described by (5):

$$\begin{cases} u_1 = m_{11}X_1 + m_{12}Y_1 + m_{13}Z_1 + m_{14} - m_{31}X_1.u_1 - m_{32}Y_1.u_1 - m_{33}Z_1.u_1 \\ v_1 = m_{21}X_1 + m_{22}Y_1 + m_{23}Z_1 + m_{24} - m_{31}X_1.v_1 - m_{32}Y_1.v_1 - m_{33}Z_1.v_1 \\ . \\ . \\ . \\ u_N = m_{11}X_N + m_{12}Y_N + m_{13}Z_N + m_{14} - m_{31}X_N.u_N - m_{32}Y_N.u_N - m_{33}Z_N.u_N \\ v_N = m_{21}X_N + m_{22}Y_N + m_{23}Z_N + m_{24} - m_{31}X_N.v_N - m_{32}Y_N.v_N - m_{33}Z_N.v_N \end{cases} \qquad (5)$$

Then, we transform this system in a matrix form:

$$U = P.V_m \text{ (eq 6)}$$

$$\begin{pmatrix} u_1 \\ v_1 \\ \\ . \\ \\ . \\ \\ . \\ \\ u_N \\ v_N \end{pmatrix} = \begin{pmatrix} X_1 & Y_1 & Z_1 & 1 & 0 & 0 & 0 & 0 & X_1 & Y_1 & Z_1 \\ 0 & 0 & 0 & 0 & X_1 & Y_1 & Z_1 & 1 & X_1 & Y_1 & Z_1 \\ \\ . & . & . & . & . & . & . & . & . & . & . \\ \\ . & . & . & . & . & . & . & . & . & . & . \\ \\ . & . & . & . & . & . & . & . & . & . & . \\ \\ X_6 & Y_6 & Z_6 & 1 & 0 & 0 & 0 & 0 & X_N & Y_N & Z_N \\ 0 & 0 & 0 & 0 & X_N & Y_N & Z_N & 1 & X_N & Y_N & Z_N \end{pmatrix} \begin{pmatrix} m_{11} \\ m_{12} \\ m_{13} \\ m_{14} \\ m_{21} \\ m_{22} \\ m_{23} \\ m_{24} \\ m_{31} \\ m_{32} \\ m_{33} \end{pmatrix} \qquad (6)$$

To find the m_{ij} parameters, the least squares method is used. The following relation is obtained:

$$V_m = (P^T.P)^{-1}.\ P^T.U \qquad (7)$$

Equation (7) represents a system of equations which can be solved by using a numerical technique such as the Gauss-Jacobi technique.

3.2 Checkpoints Auto Detection and Tracking

We use chessboard corners as features to locate the position of virtual objects in the world coordinate; these corners are extracted from each video frame (Webcam) and used to compute the perspective transformation matrix for the current view. The popular open source library for computer vision OpenCV [18] is used. It provides a function for automatically finding grids from chessboard patterns (*cvFindChessboardCorners()* and *cvFindCornerSubPix()*).

To ensure good tracking and detection, some precautions must be taken which are summarized in the following points:

- The function cvFindChessboardCorners() lists the detected corners line-by-line from left to right and bottom to top according to the first detected corner. There thus are four possibilities of classification which produce errors in the localization of the 3D world reference. In order to avoid this, a rectangular chessboard (it contains MxN corners with M≠N) is used instead of a square one and the corner where the square on its top/right direction is white is considered as the first detected corner. The center of the 3D world coordinate is fixed on the first corner of the second line (Figure1).

- The two faces of the 3D object which contain the chessboard are inclined instead of being perpendicular (Figure1); this configuration facilitates the detection of the checkpoints (chessboard corners) by minimizing the shadow effects and also gives the camera more possibilities of moving.

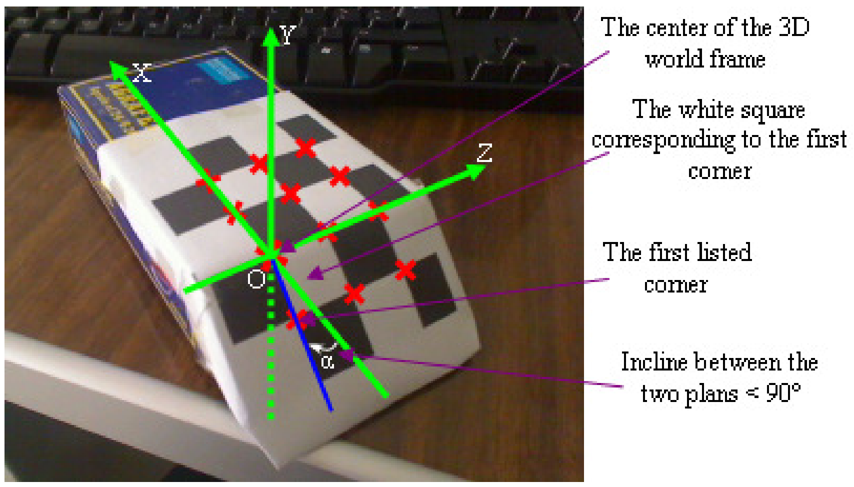
Y
X
Z
O
α
The center of the 3D world frame
The white square corresponding to the first corner
The first listed corner
Incline between the two plans < 90°

FIGURE1: Example of a 3D object with a 3x4 chessboard

3.3 The Performance Evaluation

To evaluate our method, several measures are performed using three cameras. All tests are performed on the rate frame of 20 Hz with the resolution of 640x480.

The quality of each measurement is estimated by computing the re-projection error, which is the Euclidean distance between the identified corner and the projection of its corresponding 3D coordinates onto the image; about 100 calculations are performed for each case. The obtained results, which represent the average of each case, are represented on the Table1 and compared with results obtained by Fiala and Shu [19].

Our camera model method		Results obtained by Fiala and Shu	
Camera	**Reproj.error (pixel) Std.Dev/Max**	**Camera**	**Reproj.error (pixel) Std.Dev/Max**
Sony VGP-VCC3 0.265	0.14/0.57	**Creative webcam live ultra**	0.16/1.97
Logitech QuickCam Pro 9000	0.15/0.53	**SONY 999 NTSC camera**	0.13/1.25
VRmUsbCam	0.18/0.62	**Logitech QuickCam Pro 4000**	0.29/2.62

TABLE1: Comparison of our camera model method with results obtained by Fiala and Shu [19]

According to Table1, we note that our obtained results are very close from those obtained by Fiala and Shu.

The last important factor of this tracking technique is its robustness against the problem of lighting change; actually corners have the propriety to be detected in different lighting condition. Figure 2 shows an example of corners detection in a very low lighting environment and Figure 3 shows another example in a very high lighting environment.

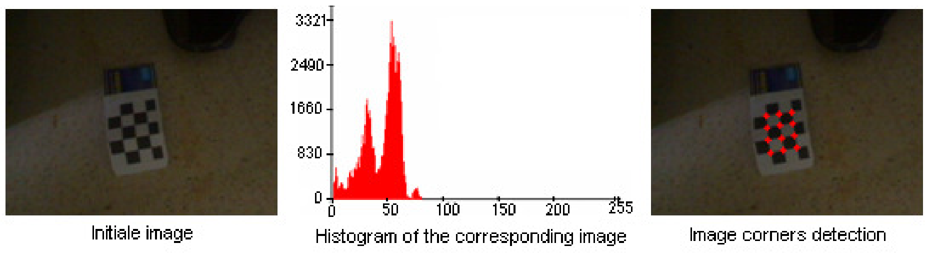

FIGURE 2: corners detection in a very low lighting environment

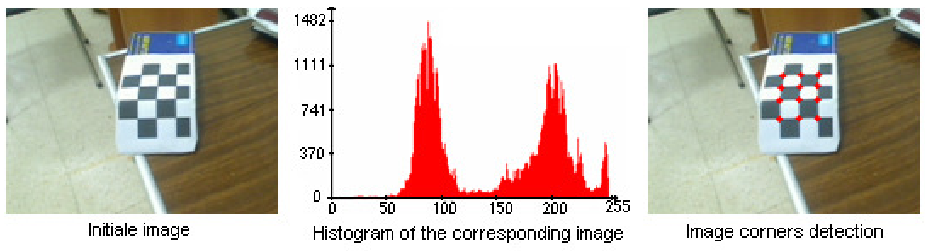

FIGURE 3: corners detection in a very high lighting environment

4. AUGMENTATION TECHNIQUE AND RESULTS

4.1 Principle of Virtual Object Insertion

With the proposal technique, the extrinsic parameters are not required to be calculated. We draw the virtual object directly on the current image using the standard graphic libraries provided by the C++Builder environment.

The principle of insertion is simple; equation (4) is used to compute the projection of any 3D point on the current image. To insert a segment, a straight line is drawn between the projections of its two extremities. For a polygon, it can be inserted using the projection of its apexes. The case of a 3D object is somehow more complicated; this object can be inserted on the current image by drawing its faces one by one. An example of a cube is presented in figure4. In this example, a virtual cube is drawn and positioned as mentioned in figure5. At first, the projections of its apexes (P_i : i = 1..8) are calculated, then its six faces are drawn one by one in a respective order which is described in the next sub-section.

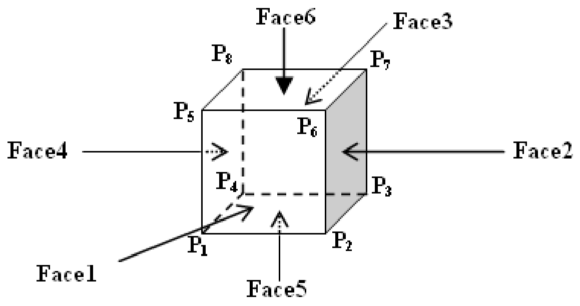

FIGURE4: the virtual object to insert (cube)

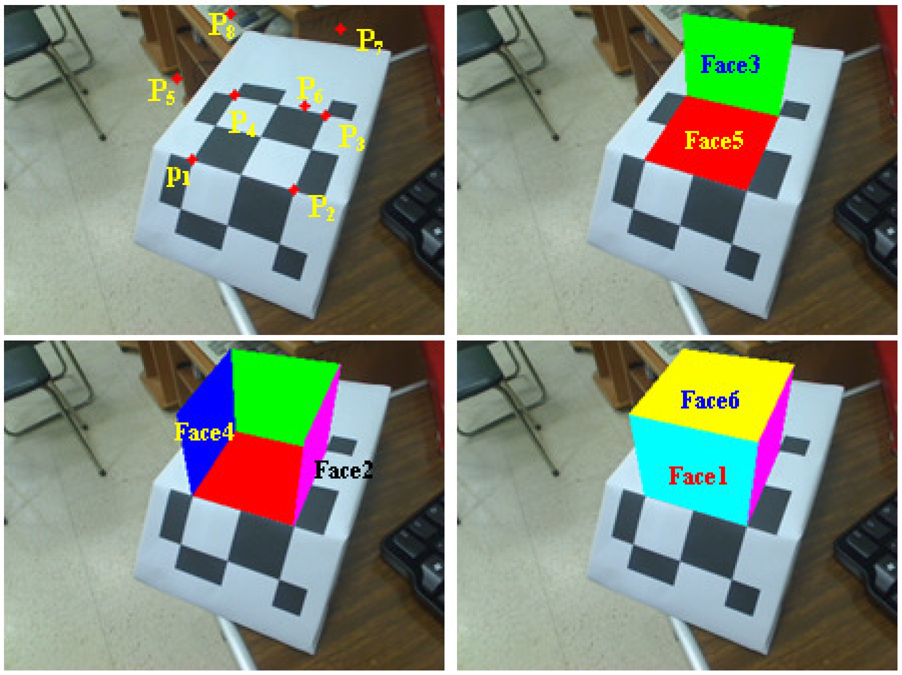

FIGURE5: The technique used to insert a 3D object

4.2 Technique Used to Avoid Occultation Problem

Occultation problem between real and virtual objects are not treated here, but only the occultation between the virtual object components, i.e. the different faces which form the augmented object. The virtual cube shown in figure5 is inserted in a special order (Face5, Face3, Face4, Face2, Face1 and Face6). If this order changes, the inserted object does not appear as a cube. Figure6 depicts a wrong presentation of a virtual object (house) when its faces are inserted in another order; the front faces are occulted by others since they are inserted in advance.

We note that this technique is not necessary when using the wire frame technique for virtual object insertion.

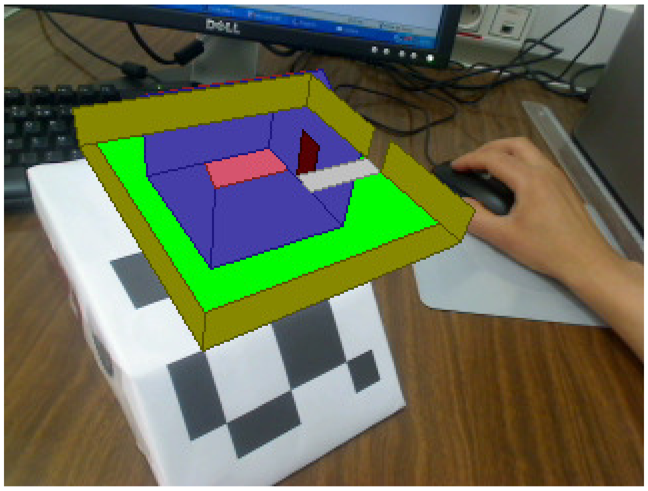
FIGURE6: Example of wrong augmentation

This problem can be avoided by estimating the camera viewpoint. This viewpoint consists of the estimation of the camera orientation according to the direction of the 3D world reference axes. We note that this technique is not required if the virtual object is drawn by the wire frame technique.

The technique is based on the use of a virtual Cube put on the horizontal plan (OXZ) where its vertices are $S_1(0,0,0)$, $S_2(0,0,L)$, $S_3(L,0,L)$, $S_4(L,0,0)$, $S_5(0,L,0)$, $S_6(0,L,L)$, $S_7(L,L,L)$ and $S_8(L,L,0)$. L can take any positive value. Projections of these 3D points on the current image ($s1(u_1,v_1)$, $s_2(u_2,v_2)$, $s_3(u_3,v_3)$, $s_4(u_4,v_4)$, $s_5(u_5,v_5)$, $s_6(u_6,v_6)$, $s_7(u_7,v_7)$, $s_8(u_8,v_8)$) are then calculated using equation 5 (see figure7 and figure8). Once calculated, a set of tests are applied using these projections to estimate the camera viewpoint.

These tests are divided into two groups; in the first group, we use the four vertices of the bottom-square to estimate their positions according to the camera. For the second one, we need to calculate the projection of "s_5" on the lines (s_1s_2) and (s_1s_4) and the projection of "s_7" on the lines (s_2s_3) and (s_3s_4); these projections are respectively $p_{5\_12}(u_{5\_12},v_{5\_12})$, $p_{5\_14}(u_{5\_14},v_{5\_14})$, $p_{7\_32}(u_{7\_32},v_{7\_32})$ and $p_{7\_34}(u_{7\_34},v_{7\_34})$. Then, using these projections we estimate the faces which are opposed to the camera.

In the example shown in figure7, the verified tests are:

- ($u_4 < u_1 < u_2$) and ($v_5 > v_{5\_12}$) and ($v_5 > v_{5\_14}$) which means that the camera is oriented according to $\overrightarrow{OX}$ and $\overrightarrow{OZ}$ axes.

In the example shown in figure8, the verified tests are:

- ($u_1 < u_2 < u_3$) and ($v_5 > v_{5\_12}$) and ($v_7 > v_{7\_32}$) which means that the camera is oriented according to $\overrightarrow{OX}$ and $-\overrightarrow{OZ}$ axes.

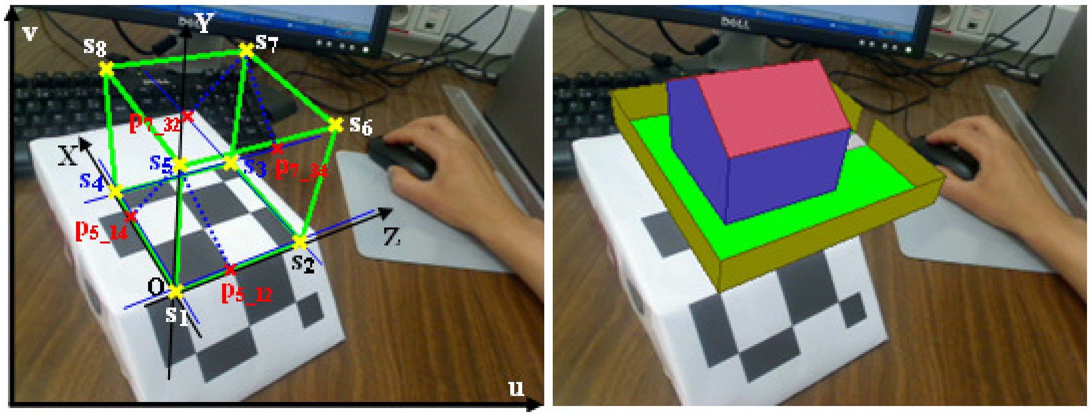

FIGURE7: Example 1 of the camera estimation viewpoint

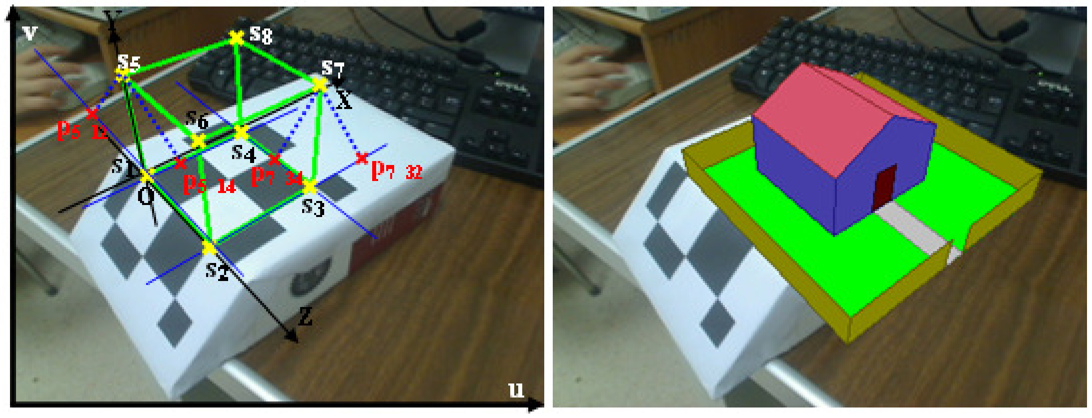

FIGURE8: Example 2 of the camera estimation viewpoint

4.3 Results

We present here two application examples of our approach. The scene captured by the camera (webcam) is augmented by virtual objects in real time. In the first example, the "Logitech QuickCam Pro 9000" webcam is used with the resolution 320x240, in the second one; a "Sony VGP-VCC3 0.265A" webcam is used with a resolution 640x480. Images presented in figure9 and figure10 are selected arbitrarily from the webcams.

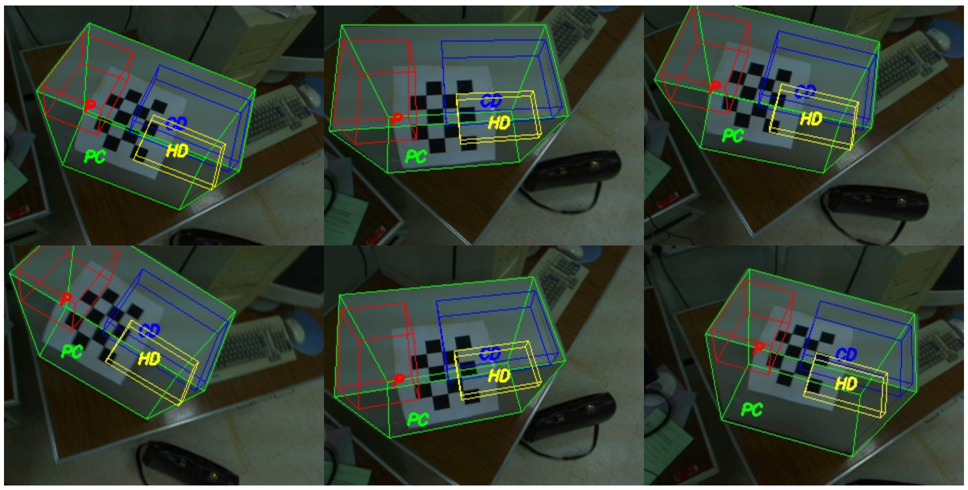

FIGURE9: Example of virtual object insertion into real scene with the wire frame technique

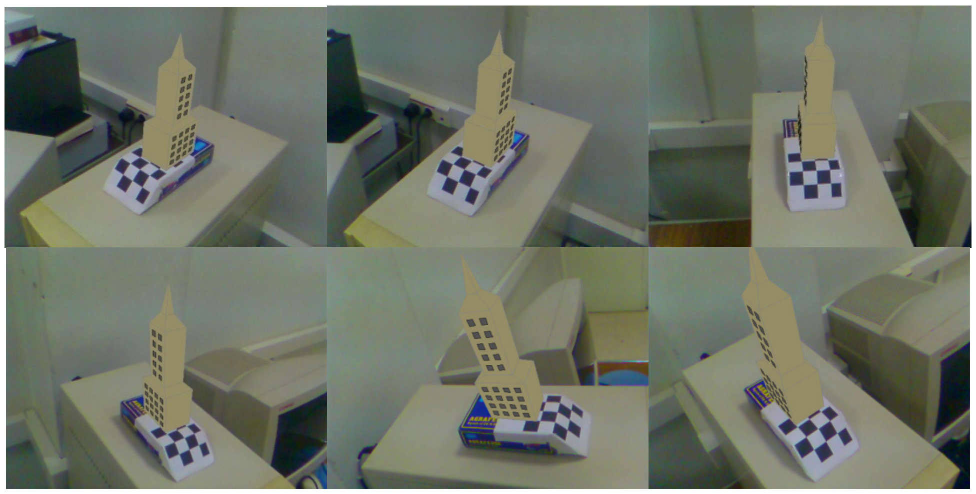

FIGURE10: Example of augmentation with a 3D virtual model.

According to these examples, we note that using 12 fiducial points (corners) is sufficient to model the camera using a pinhole model; the virtual object is positioned in the desired place with a good accuracy and it size depends of the camera depth. We also note that the used technique of tracking (tracking of chessboard corners) gives good and acceptable results and the virtual object is well tracked across camera frame in real time. The last point is the efficacy of the developed technique of virtual object insertion; the virtual object follows the motion of the camera and it is positioned and drawn correctly without any information about intrinsic or extrinsic parameters of the camera.

5. CONCLUSION

This paper introduces a technique of virtual object tracking in real time. This technique is developed as part of an augmented reality system and does not require any information or computation of the camera parameters (intrinsic and extrinsic parameters). It is based on both detection of Chessboard corners and a least squares method to estimate the perspective transformation matrix for the current view of the camera.

The use of this technique is simple and does not require initialization steps or manual intervention, the only requirement is the tracking of the marker (chessboard) across images provided by the camera.

The results show the efficiency of the tracking method based on detection of chessboard corners; the major advantages of tracking corners are their detection robustness at a large range of distances, their reliability under severe orientations, and their tolerance of lighting changes or shadows. The technique used for the insertion of virtual objects gives its proof when the camera parameters are not calculated.

6. REFERENCES

1. R. Azuma, Y. Baillot,R. Behringer, S. Feiner, and S. Julier, "*Recent advances in augmented reality*", IEEE Computer Graphics and Applications, 21(6), pp. 34-47, 2001.

2. R. Tsai, "*An efficient and accurate camera calibration technique for 3D machine vision*", IEEE Proceedings CVPR, pp. 364-374, June 1986.

3. M. Tuceryan, M., Greer, D., Whitaker, R., Breen, D., Crampton, C., Rose, E., and Ahlers, K., "*Calibration Requirements and Procedures for Augmented Reality*", IEEE Trans. On Visualization and Computer Graphics, pp. 255-273, September 1995.

4. O.D. Faugeras and G. Toscani. "*The calibration problem for stereo*", In Proceedings of CVPR, pp. 15-20, 1986.

5. Z. Zhang, "*Camera calibration with one-dimensional objects*", In Proc. 7th European Conference on Computer Vision, (IV), pp. 161–174, 2002.

6. Zhou, F., Duh, H.B.L., Billinghurst, M. " *Trends in Augmented Reality Tracking, Interaction and Display: A Review of Ten Years of ISMAR*". Cambridge, UK: 7th IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR 2008), pp. 15-18, Sep 2008.

7. H. Wuest, F. Vial and D. stricker. "*Adaptative line tracking with multiple hypotheses for augmented reality*". In ISMAR '05, pp. 62-69, 2005.

8. Hyun Seung Yang, Kyusung Cho, Jaemin Soh, Jinki Jung, Junseok Lee: "*Hybrid Visual Tracking for Augmented Books*". ICEC 2008: pp. 161-166, 2008.

9. H. Kato and M. Billinghurst. "*Marker tracking and HMD calibration for a video-based augmented reality conferencing system*". IWAR '99 : Proceedings of the 2nd IEEE and ACM International Workshop on Augmented Reality, Washington, DC, USA. IEEE Computer Society, pp. 85–92, 1999.

10. ARToolKit site: http://www.hitl.washington.edu/artoolkit.

11. ISO. International standard, iso/iec18004. ISO International Standard, Jun 2000.

12. Diego Lopez de Ipina, Paulo R. S. Mendonca, and Andy Hopper. "*Trip: A low-cost vision-based location system for ubiquitous computing*". Personal Ubiquitous Comput., 6(3), pp. 206–219, , 2005.

13. Fiala, M.: ARTag: "*a fiducial marker system using digital techniques*". In: CVPR'05, vol. 1, pp. 590 – 596, 2005.

14. Rice, A., Beresford, A., Harle, R.: "*Cantag: an open source software toolkit for designing and deploying marker-based vision systems*". In: 4th IEEE International Conference on Pervasive Computing and Communications, 2006.

15. Lowe, D.G.: "*Distinctive Image Features from Scale-Invariant Keypoints*". International Journal of Computer Vision 20(2), pp. 91–110, 2004.

16. Bay, H., Tuytelaars, T., VanGool, L.: "*SURF: Speeded Up Robust Features*". In: 9th European Conference on Computer Vision, pp. 404–417, 2006.

17. T. Lee and T. Höllerer. 2009. "*Multithreaded Hybrid Feature Tracking for Markerless Augmented Reality*". IEEE Transactions on Visualization and Computer Graphics 15, 3, pp. 355-368, May. 2009.

18. Open source computer vision library. In: http://www.intel.com/research/mrl/research/opencv.

19. Mark Fiala, Chang Shu: "*Self-identifying patterns for plane-based camera calibration*". Mach. Vis. Appl. 19(4), pp. 209-216, 2008.