EUROGRAPHICS 2002 / G. Drettakis and H.-P. Seidel
(Guest Editors)

Volume 21 (2002), Number 3

Virtual Visual Servoing: a framework for real-time augmented reality

Éric Marchand and François Chaumette

IRISA - INRIA Rennes
Campus de Beaulieu, 35042 Rennes France

Abstract

This paper presents a framework to achieve real-time augmented reality applications. We propose a framework based on the visual servoing approach well known in robotics. We consider pose or viewpoint computation as a similar problem to visual servoing. It allows one to take advantage of all the research that has been carried out in this domain in the past. The proposed method features simplicity, accuracy, efficiency, and scalability wrt. to the camera model as well as wrt. the features extracted from the image. We illustrate the efficiency of our approach on augmented reality applications with various real image sequences.

1. Introduction

We consider in this paper the problem of real-time augmented reality (AR) [1, 2]. We do not restrict ourself to a particular display technology and we restrict the problem to the use of a unique vision sensor: a camera. Now that little cameras such as webcams are available at low cost for any personal computer, a vision-based augmented reality system is an attractive interface for various applications such as video games, architecture, interior design, etc. We will therefore focus on the registration techniques that allow alignment of real and virtual worlds using images acquired in real-time by a moving camera. In such systems AR is mainly a pose (or viewpoint) computation issue. In this paper we will address the pose computation problem as a virtual visual servoing problem. Though some new interesting approaches avoid considering that camera position and parameters are available [15], most of vision-based AR systems rely on the availability of this information.

Most approaches consider the pose computation as a registration problem that consists of determining the relationship between 3D coordinates of points (or other features: lines, ellipses, ...) and their 2D projections onto the image plane. The position of these 3D features in a world frame have to be known with a good accuracy. They can be part of a known pattern, but may also result in the knowledge of the environment blueprints. Computing pose leads to the estimation of the position and orientation of the camera with respect to a particular world or object frame.

Many approaches have been developed to estimate the position of a camera with respect to an object by considering its projection in the image plane. The geometric primitives considered for the estimation of the pose are often points [11, 6], segments [8], contours [16, 9], conics [23, 4], or cylindrical objects [7]. However, even though combining different types of primitives is fundamental to compute the viewpoint in a real environment, very few methods propose such combination (see [22] for the joint use of points and straight lines). We will address this issue in the present paper. Another important issue is the registration problem. *Purely geometric* (eg, [8]), or *numerical and iterative* [6] approaches may be considered. *Linear approaches* use a least-squares method to estimate the pose. *Full-scale non-linear optimization techniques* (e.g., [16, 17, 24]) consists of minimizing the error between the observation and the back-projection of the model. Minimization is handled using numerical iterative algorithms such as Newton-Raphson or Levenberg-Marquartd. The main advantage of these approaches are their accuracy. The main drawback is that they may be subject to local minima and, worse, divergence. Therefore they usually require a good guess of the solution to ensure correct convergence. However, since the camera displacement between two successive images is small, dealing with augmented reality applications the risk of divergence is quite nonexistent. Another draw-

back is that these approaches require the estimation of the explicit computation of a Jacobian. An analytical derivation of this Jacobian may be a complex and error prone task while its on-line estimation may lead to a less efficient and longer minimization. We show in this paper that the presented formulation allows easily consideration of a large number of Jacobians (called here Interaction matrices) available for various visual features. Partial 3D models of the scene are necessary in most of these approaches. Though they can be obtained using the same approaches (eg [5]) we do not address this issue in this paper since we seek a sequential and real-time computation of the camera viewpoint. Indeed approaches that allow simultaneous estimation of the pose and the structure of the scene require more that one image and are therefore more dedicated to post production applications than to real-time applications.

In this paper we propose a formulation of pose computation involving a full scale non-linear optimization: Virtual Visual Servoing (VVS). We consider the pose computation problem as similar to 2D visual servoing [25]. Visual servoing or image-based camera control [13, 10, 12] allows to control a camera wrt. to its environment. More precisely it consists in specifying a task (mainly positioning or target tracking tasks) as the regulation in the image of a set of visual features. A set of constraints are defined in the image space. A control law that minimizes the error between the current and desired position of these visual features can then be automatically built. This approach has proven to be an efficient solution to camera positioning task within the robotics context (see papers in [12]) and more recently in computer graphics [21]. Considering pose as an image-based visual servoing problem takes advantage of all the background knowledge and the results in this research area. It allows us to propose a very simple and versatile formulation of this important problem. One of the main advantages of this approach is that it allows consideration of different geometrical features within the same process. We show how this framework is easily scaled when the camera parameters are unknown or modified.

In the remainder of this paper, we present in Section 2 the principle of the approach and its application to pose computation (Section 3.1). In Section 4 we show how this approach scale to the case when camera parameters are unknown or modified. In Section 5, we present several experimental results.

2. Principle

As already stated, the basic idea of our approach is to define the pose computation problem as the dual problem of 2D visual servoing [10, 13]. In visual servoing, the goal is to move the camera in order to observe an object at a given position in the image. This is achieved by minimizing the error between a desired state of the image features $\mathbf{p_d}$ and the current state $\mathbf{p}$. If the vector of visual features is well chosen, there is only one final position of the camera that allows this minimization to be achieved. We now explain why the pose computation problem is very similar.

To simply illustrate the principle, let us consider the case of an object made of points. Let us define a virtual camera with intrinsic parameters ξ located at a position such that the object frame is related to the camera frame by the homogeneous 4×4 matrix ${}^c\mathbf{M}_o$. ${}^c\mathbf{M}_o$ defines the pose whose parameters are called extrinsic parameters. The position of the object point ${}^c\mathbf{P}$ in the camera frame is defined by:

$$ {}^c\mathbf{P} = {}^c\mathbf{M}_o\, {}^o\mathbf{P} \tag{1} $$

and its projection in the digitized image by:

$$ \mathbf{p} = pr_\xi({}^c\mathbf{P}) = pr_\xi({}^c\mathbf{M}_o\, {}^o\mathbf{P}) \tag{2} $$

where $pr_\xi(.)$ is the projection model according to the intrinsic parameters ξ. The goal of the pose computation problem is to estimate the extrinsic parameters by minimizing the error between the observed data denoted $\mathbf{p_d}$ (usually the position of a set of features in the image) and the position $\mathbf{p}$ of the same features computed by back-projection according to the current extrinsic and intrinsic parameters (as defined in Equation 2). In order to ensure this minimization we move the virtual camera (initially in ${}^{ci}\mathbf{M}_o$) using a visual servoing control law. When the minimization is achieved, the parameters of the virtual camera will be ${}^{cf}\mathbf{M}_o$. We have illustrated this example with points. For other geometrical features, equations (1) and (2) are obviously different but the principle remains identical. This process is illustrated in Figure 1 where straight lines are considered.

3. Virtual visual servoing

3.1. Visual servoing and pose

For pose computation, intrinsic camera parameters must be known, therefore, we consider that the position of the features are expressed in the metric space. We denote $\mathbf{p_{m_d}}$ the features extracted from the real image and $\mathbf{p_m}$ the same features computed by back-projection.

The goal is to minimize the error $\|\mathbf{p_m} - \mathbf{p_{m_d}}\|$. As in classical visual servoing, we define a task function $\mathbf{e}$ to be achieved by the relation:

$$ \mathbf{e} = \mathbf{C}\,(\mathbf{p_m}(\mathbf{r}) - \mathbf{p_{m_d}}) \tag{3} $$

where $\mathbf{r}$ are the camera extrinsic parameters matrix (i.e., the camera viewpoint). Matrix$\mathbf{C}$ called the combination matrix is chosen such that $\mathbf{C}\mathbf{L}_{\mathbf{p_m}}$ is full rank. It allows to take into account more visual features in $\mathbf{p}$ than the number of controlled degrees of freedom (6 in this case). We have:

$$ \dot{\mathbf{e}} = \frac{\partial \mathbf{e}}{\partial \mathbf{r}} \frac{\partial \mathbf{r}}{\partial t} = \mathbf{C}\mathbf{L}_{\mathbf{p_m}} \mathbf{T_c} \tag{4} $$

Matrix $\mathbf{L}_{\mathbf{p_m}}$ is classically called the interaction matrix or image Jacobian in the visual servoing community [13, 10]. It links

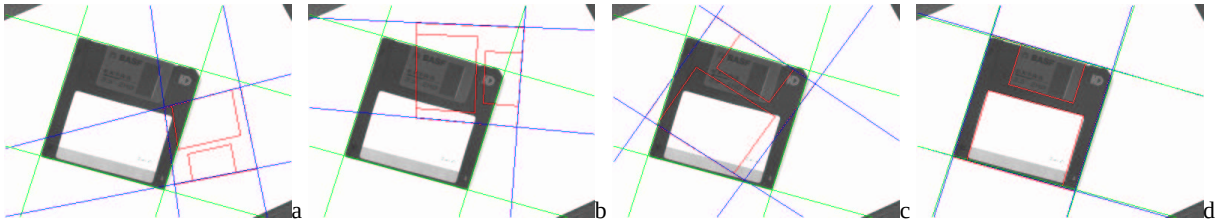

Figure 1: *Pose computation by virtual visual servoing. The principle of our algorithm is to iteratively modify using a visual servoing control law the position of a virtual camera in order to register the desired features extracted from the image (in green) and the current one obtained by back-projection of the object model (in blue) for a given pose. Image (a) corresponds to the initialization while in image (d) registration has been achieved and the pose is computed.. This figure illustrates the registration convergence for one image. It also illustrates the minor influence of the initialization, indeed the initial position/orientation of the camera is very different from the final computed one. In this example, straight lines are considered to compute pose.*

the motion of the features in the image to the camera velocity $\mathbf{T_c}$:

$$\dot{\mathbf{p_m}} = \frac{\partial \mathbf{p_m}}{\partial \mathbf{r}} \frac{d\mathbf{r}}{dt} = \mathbf{L_{p_m}} \mathbf{T_c} \tag{5}$$

If we specify an exponentially decoupled decrease of the error $\mathbf{e}$, that is:

$$\dot{\mathbf{e}} = -\lambda \mathbf{e} \tag{6}$$

where λ is a proportional coefficient that tunes the decay rate, we can derive the control law. Indeed, using (6) and (4) we obtain:

$$\mathbf{C} \mathbf{L_{p_m}} \mathbf{T_c} = -\lambda \mathbf{e} \tag{7}$$

which leads to the ideal control law:

$$\mathbf{T_c} = -\lambda \left(\mathbf{C} \mathbf{L_{p_m}}\right)^{-1} \mathbf{e} \tag{8}$$

We will see in the next section that the interaction matrix depends on the pose between the camera and the target and on the value of the visual feature: $\mathbf{L_{p_m}} = \mathbf{L_{p_m}}(\mathbf{p_r}, \mathbf{r})$. In practice, a model $\bar{\mathbf{L}}_{\mathbf{p_m}}$ of $\mathbf{L_{p_m}}$ is used, and we obtain:

$$\mathbf{T_c} = -\lambda \left(\mathbf{C} \bar{\mathbf{L}}_{\mathbf{p_m}}\right)^{-1} \mathbf{e} \tag{9}$$

We will see later on the different possible choices for $\bar{\mathbf{L}}_{\mathbf{p_m}}$ and $\mathbf{C}$.

Convergence and stability are important issues in dealing with such control law. Using (9) in (4) the behavior of the closed loop system is obtained:

$$\dot{\mathbf{e}} = -\lambda \left(\mathbf{C} \mathbf{L_{p_m}}\right) \left(\mathbf{C} \bar{\mathbf{L}}_{\mathbf{p_m}}\right)^{-1} \mathbf{e} \tag{10}$$

The positivity condition:

$$\left(\mathbf{C} \mathbf{L_{p_m}}\right) \left(\mathbf{C} \bar{\mathbf{L}}_{\mathbf{p_m}}\right)^{-1} > 0 \tag{11}$$

is thus sufficient to ensure the decay of $\|\mathbf{e}\|$ which implies the global asymptotic stability and the convergence of the system. Let us now consider the different possible choices of $\mathbf{C}$ and $\bar{\mathbf{L}}_{\mathbf{p_m}}$.

In all the experiments reported here, the dimension k of the visual feature vector $\mathbf{p}$ is greater than 6 (i.e., the chosen visual features are redundant). Since the combination matrix has to be of dimension $6 \times k$ and of rank 6, the simplest choice is to define $\mathbf{C}$ as the pseudo inverse of the interaction matrix used:

$$\mathbf{C} = \bar{\mathbf{L}}^{+}_{\mathbf{p_m}} \tag{12}$$

where $\mathbf{L}^{+} = (\mathbf{L}^T \mathbf{L})^{-1} \mathbf{L}^T$. In that case, where $\mathbf{C} \bar{\mathbf{L}}_{\mathbf{p_m}} = \mathbf{I}_6$, the stability condition is given by:

$$\bar{\mathbf{L}}^{+}_{\mathbf{p_m}} \mathbf{L_{p_m}} > 0 \tag{13}$$

Let us note that according to this choice for $\mathbf{C}$, the control is simplified and is given by:

$$\mathbf{T_c} = -\lambda \bar{\mathbf{L}}^{+}_{\mathbf{p_m}} (\mathbf{p_m} - \mathbf{p_{m_d}}) \tag{14}$$

As already stated the choice of $\bar{\mathbf{L}}_{\mathbf{p_m}}$ is important. Many choices are theoretically possible:

- $\bar{\mathbf{L}}_{\mathbf{p_m}} = \mathbf{L_{p_m}}(\mathbf{p_{m_d}}, \mathbf{r_d})$: the interaction matrix is computed only once with the final value of the pose and of the visual features. This choice is the most classical in robotics. It ensures the local asymptotic stability of the system since the positivity condition is ensured in the neighborhood of the desired position. That means that, if the error $\mathbf{p_m} - \mathbf{p_{m_d}}$ is small enough, the convergence of $\mathbf{p_m}$ to $\mathbf{p_{m_d}}$ will be obtained. However in our case, though $\mathbf{p_d}$ is known, $\mathbf{r_d}$ is what try to estimate and is then unknown. This choice is thus impossible for AR applications.
- $\bar{\mathbf{L}}_{\mathbf{p_m}} = \mathbf{L_{p_m}}(\mathbf{p_m}, \mathbf{r})$, the interaction matrix is computed at each iteration with the current value of the pose and of the visual features. We may think that the global stability is demonstrated since $\mathbf{L}^{+}_{\mathbf{p_m}} \mathbf{L_{p_m}} = \mathbf{I}_6 > 0$ whatever the value of $\mathbf{p_m}$. However in that case matrix $\mathbf{C}$ is not constant and equation (4) should thus take into account the variation of $\mathbf{C}$. This leads to inextricable computations, and thus, once again, only the local stability can be obtained.
- $\bar{\mathbf{L}}_{\mathbf{p_m}} = \mathbf{L_{p_m}}(\mathbf{p_{m_i}}, \mathbf{r_i})$ where $\mathbf{r_i}$ is the initial pose of the virtual camera and $\mathbf{p_i}$ the initial value of the visual features. This choice is interesting since $\bar{\mathbf{L}}^{+}_{\mathbf{p_m}}$ is computed

only once. Once again, positivity condition (13) will be satisfied only if $\mathbf{p_{m_i}} - \mathbf{p_{m_d}}$ is small.

This last choice can be used in AR application since $\mathbf{p_{m_i}}$ and $\mathbf{r_i}$ are available. However since the presented results show that convergence is ensured in few iterations, we have used $\bar{\mathbf{L}}_{\mathbf{p_m}} = \mathbf{L}_{\mathbf{p_m}}(\mathbf{p_m}, \mathbf{r})$ this choice being not time consuming when the number of visual features is small.

We have seen that only local stability can be demonstrated. It means that the convergence may not be reached if the error $\mathbf{p_m} - \mathbf{p_{m_d}}$ is too large. However in AR application, the motion between two successive images acquired at video rate is small enough to ensure the convergence of the control law if we use as initialization of $\mathbf{p_m}$ and $\mathbf{r}$ the result obtained from the previous image. Indeed in practice, it has been shown with experimental results in visual servoing that the convergence is always obtained when the camera displacement has an orientation error less that 30^o on each axis. Potential problems may thus appear only for the very first image where the initialization has not to be too coarse.

3.2. The interaction matrices

Any kind of feature can be considered within this control law as soon as we are able to compute the corresponding interaction matrix $\mathbf{L}_{\mathbf{p_m}}$. In [10], a general framework to compute $\mathbf{L}_{\mathbf{p_m}}$ is proposed. This is one of the advantages of this approach with respect to other non-linear pose computation approaches. Indeed we are able to perform pose computation from a large set of image information (points, lines, circles, quadrics, distances, etc...) within the same framework. We can also very easily mix different features by adding features to vector $\mathbf{p}$ and by "stacking" the corresponding interaction matrices. Furthermore if the number or the nature of visual features is modified over time, the interaction matrix $\mathbf{L}_{\mathbf{p_m}}$ and the vector error $\mathbf{p}$ is modified consequently. We now consider classical geometrical features (point, straight line, circle and cylinder) which will be examined in the result section of this paper.

Case of points. Let us define $\mathbf{M} = (X, Y, Z)^T$ the coordinates of a point in the camera frame. The coordinates of the perspective projection of this point in the image plane is given by $\mathbf{m} = (x, y)^T$ with:

$$\begin{cases} x = X/Z \\ y = Y/Z \end{cases} \tag{15}$$

We have to use the interaction matrix $\mathbf{L}_{\mathbf{p_m}}$ that links the motion $\dot{\mathbf{p}} = (\dot{x}, \dot{y})$ of a point $\mathbf{p} = (x, y)$ in the image to $\mathbf{T_c}$. For one point $\mathbf{L}_{\mathbf{p_m}}$ is a 2×6 matrix. The interaction matrix $\mathbf{L}_{\mathbf{p_m}}$ that relates the motion of a point in the image to the camera motion is well known [13, 10] and is given by:

$$\mathbf{L}_{\mathbf{p_m}} = \begin{pmatrix} -\frac{1}{Z} & 0 & \frac{x}{Z} & xy & -(1+x^2) & y \\ 0 & -\frac{1}{Z} & \frac{y}{Z} & 1+y^2 & -xy & -x \end{pmatrix} \tag{16}$$

Let us note here that dealing with points, our approach is very similar to the Lowe's approach [16].

Case of straight line. A straight line can be defined as the intersection of two planes:

$$\begin{cases} A_1 X + B_1 Y + C_1 Z = 0 \\ A_2 X + B_2 Y + C_2 Z + D_2 = 0 \end{cases} \tag{17}$$

The equation of the projected line in the image plane is given by:

$$x\cos\theta + y\sin\theta - \rho = 0. \tag{18}$$

It is possible to compute the interaction matrix related to $\mathbf{p_m} = (\theta, \rho)$. It is given by [10]:

$$\mathbf{L}_\theta = \begin{pmatrix} \lambda_\theta \cos\theta & \lambda_\theta \sin\theta & -\lambda_\theta \rho & \rho\cos\theta & -\rho\sin\theta & -1 \end{pmatrix}$$

$$\mathbf{L}_\rho = \begin{pmatrix} \lambda_\rho \cos\theta & \lambda_\rho \sin\theta & -\lambda_\rho \rho & (1+\rho^2)\sin\theta & -(1+\rho^2)\cos\theta & 0 \end{pmatrix} \tag{19}$$

where $\lambda_\theta = (A_2 \sin\theta - B_2 \cos\theta)/D_2$ and $\lambda_\rho = (A_2 \rho \cos\theta + B_2 \rho \sin\theta + C_2)/D_2$.

Case of circle and cylinder. A circle is defined as the intersection of a sphere and a plane.The projection in the image plane of a circle is an ellipse whose parameters are $\mathbf{p_m} = (x_c, y_c, \mu_{02}, \mu_{20}, \mu_{11})$. x_c, y_c is the center of the ellipse while μ_{02}, μ_{20} and μ_{11} are the order 2 centered moments of the ellipse in [10].

We also consider in the results section the case of the cylinder. Here again we refer the reader to [10] for its parameterization and the related interaction matrix.

4. Generalization to unknown camera parameters

When the camera intrinsic parameters are unknown (or change during an experiment), computing the pose is not sufficient to allow realistic insertion of virtual objects. These parameters have then to be estimated. This calibration process may be achieved off line using a calibration pattern using classical approach or on-line using the approach proposed in this Section.

4.1. Principle

For the pose problem, the visual feature $\mathbf{p}$ can be expressed directly in the metric space (and were denoted $\mathbf{p_m}$). Now, since the camera parameters are unknown, $\mathbf{p}$ can be only computed in the digitized space and the image features are now denoted $\mathbf{p_p}$.

Within this context, the current position of the feature in the image, obtained by back-projection, is also function of the intrinsic parameters ξ. The error to be minimized, expressed in digitized space, is then defined by $\|\mathbf{p_p}(\mathbf{r}, \xi) - \mathbf{p_{p_d}}\|$ and the corresponding task function define by:

$$\mathbf{e} = \mathbf{C}(\mathbf{p_p}(\mathbf{r}, \xi) - \mathbf{p_{p_d}}) \tag{20}$$

In this new case, the motion of the features in the image are related to the camera velocity $\mathbf{T}_c$ and to the time variation of the intrinsic parameters by:

$$\dot{\mathbf{p}}_\mathbf{p} = \mathbf{L}_{\mathbf{p}_\mathbf{p}}\mathbf{T_c} + \frac{\partial \mathbf{p_p}}{\partial \xi}\frac{d\xi}{dt} \quad \text{where} \quad \mathbf{L}_{\mathbf{p}_\mathbf{p}} = \frac{\partial \mathbf{p_p}}{\partial \mathbf{p_m}}\mathbf{L}_{\mathbf{p_m}} \tag{21}$$

that can be rewritten as:

$$\dot{\mathbf{p}}_\mathbf{p} = \mathbf{H_p V} \qquad \text{with} \qquad \mathbf{V} = \begin{pmatrix} \mathbf{T_c} \\ \dot{\xi} \end{pmatrix} \tag{22}$$

and

$$\mathbf{H_p} = \begin{pmatrix} \frac{\partial \mathbf{p_p}}{\partial \mathbf{p_m}}\mathbf{L}_{\mathbf{p_m}} & \frac{\partial \mathbf{p}}{\partial \xi} \end{pmatrix} \tag{23}$$

If $\mathbf{C}$ is chosen such as $\mathbf{C} = \mathbf{H}^+$, we obtain as "control law":

$$\mathbf{V} = -\lambda \mathbf{H}_p^+ \mathbf{e} \tag{24}$$

Thus, a sufficient number of primitives have to be selected in order for $\mathbf{H}$ be of full rank.This number depends directly on the number of parameters considered in ξ (usually 4 or 5) and in $\mathbf{r}$ (6).

4.2. New Interaction matrices

We now determine the analytical form of $\mathbf{H_p}$ matrices when points and circles are considered. Straight lines or any other geometrical primitive can be handled using the same approach. We consider in this paper the most classical camera model (in [20], more complex models involving lens distortion are considered).

Case of point If we denote (u,v) the position of the corresponding pixel in the digitized image, this position is related to the coordinates (x,y) in the normalized space by:

$$\begin{cases} u = u_0 + p_x x \\ v = v_0 + p_y y \end{cases} \tag{25}$$

The four parameters to be estimated are thus $\xi = \{p_x, p_y, u_0, v_0\}$ where (u_0, v_0) are the coordinates of the principal point and p_x, p_y are the ratio between the focal length and the size of a pixel.

Let us note that more simple model may be considered as well. Indeed, in this context, considering u_0 and v_0 as the center of the image and assuming $p_x = \alpha p_y$, where α is constant given by the camera manufacturer, is usually sufficient.

We have to compute the Jacobian matrix $\mathbf{H_p}$ that links the motion $\dot{\mathbf{p}} = (\dot{u}, \dot{v})$ of a point $\mathbf{p} = (u,v)$ in the image to $[\mathbf{T_c} \quad \dot{\xi}]^T$. For one point $\mathbf{L}_{\mathbf{p_p}}$ is a 2×6 matrix and $\frac{\partial \mathbf{p}}{\partial \xi}$ is a 2×4 matrix.

The interaction matrix $\mathbf{L}_{\mathbf{p_p}}$ is given by:

$$\mathbf{L}_{\mathbf{p_p}} = \begin{pmatrix} p_x & 0 \\ 0 & p_y \end{pmatrix} \mathbf{L}_{\mathbf{p_m}} \tag{26}$$

where $\mathbf{L}_{\mathbf{p_m}}$ is given by equation (16)

Furthermore, from (25), differentiating u and v for ξ leads very easily to:

$$\frac{\partial \mathbf{p}}{\partial \xi} = \begin{pmatrix} x & 0 & 1 & 0 \\ 0 & y & 0 & 1 \end{pmatrix} \tag{27}$$

Case of a circle. Let us note $(u_c, v_c, m_{20}, m_{02}, m_{11})$ the parameters that describe an ellipse in the digitized image. It can be shown that they are linked to the metric representation $(x_c, y_c, \mu_{20}, \mu_{02}, \mu_{11})$ by:

$$\begin{array}{llll} u_c = u_0 + p_x x_c & v_c = v_0 + p_y y_c & & \\ m_{20} = \mu_{20} p_x^2 & m_{02} = \mu_{02} p_y^2 & m_{11} = \mu_{11} p_x p_y & \end{array} \tag{28}$$

$\frac{\partial \mathbf{p_p}}{\partial \mathbf{p_m}}$ is thus a 5×5 diagonal matrix whose diagonal is given by: $\left(p_x, p_y, p_x^2, p_y^2, p_x p_y\right)$. Dealing with the camera intrinsic parameters, we get $\frac{\partial \mathbf{p_p}}{\partial \xi}$ as:

$$\frac{\partial \mathbf{p}_p}{\partial \xi} = \begin{pmatrix} (u-u_0)/p_x & 0 & 1 & 0 \\ 0 & (v-v_0)/p_y & 0 & 1 \\ 2m_{20}/p_x & 0 & 0 & 0 \\ 0 & 2m_{02}/p_y & 0 & 0 \\ m_{11}/p_x & m_{11}/p_y & 0 & 0 \end{pmatrix} \tag{29}$$

5. Experimental Results

5.1. Software architecture and Implementation

The architecture of our system is similar to those proposed in [3] or [14]. Since our approach is based on the visual servoing framework we rely on a library dedicated to such systems and called VISP (Visual servoing platform) [18]. This library is written in C++ and proposes both the feature tracking algorithms and the pose computation and calibration algorithms. A new software component based on Open Inventor has been written to allow the insertion of virtual objects in the images. All the experiments presented in the next paragraph have been carried out on a simple PC with an Nvidia 3D board and an Imaging technology IC-Comp framegrabber.

Most of the images considered in this paper are quite simple. Our goal was indeed to illustrate the possibility to compute a precise camera viewpoint and, if required, camera intrinsic parameters with various image features (points, lines, circle, cylinders, ...) in real-time with low cost hardware. It is obvious that considering well contrasted images and precise pattern is helpful to achieve this goal, but considering a more complex image processing algorithm is always possible depending on the application as reported in [19, 26].

5.2. Augmented reality experiments

Comparison with other approaches. In this paragraph we present results related to the application of this framework to augmented reality. We first compare our method with classical algorithms proposed in the literature: linear estimation,

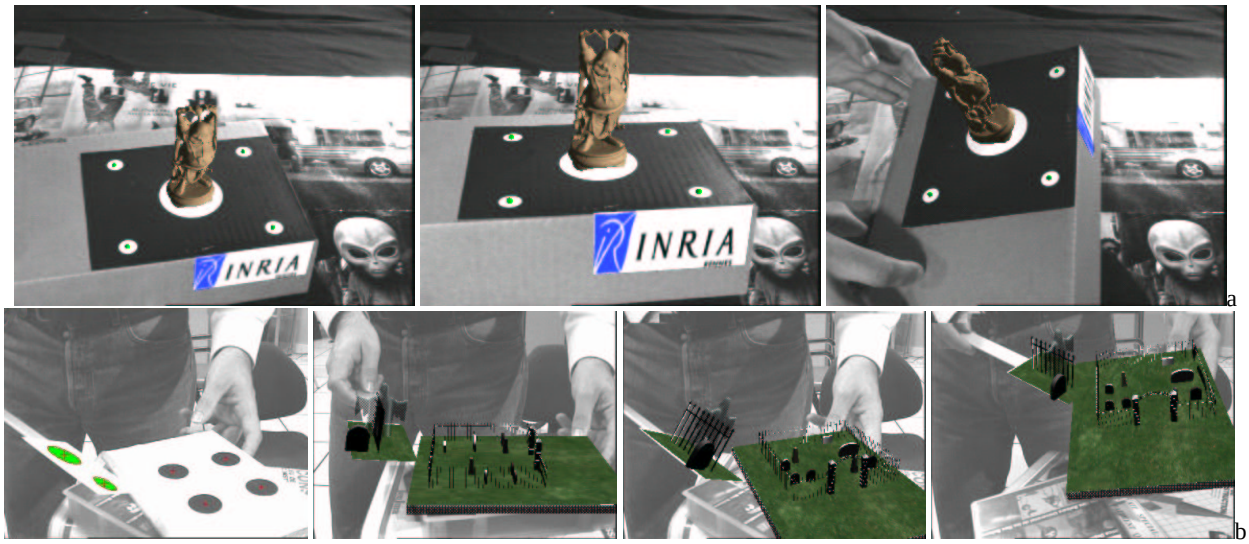

Figure 2: *We report here two augmented reality experiments that use precise pattern to compute the viewpoint. In (a) four points and a circle are consider while in (b) two pose are computed for two different objects: the former is computed using a point and a circle and is related with the insertion of the "ghost" while the later is computed using four points and is related to the insertion of the "graveyard" (the first image of row (b) shows the fisrt image prior to its "augmentation" and the result of the image processing algorithm). In both experiments poses and rendering are computed at video rate (25Hz) with a one frame delay. In application such as interactive video games, by moving and orienting a simple pattern the player may modify on line and in real time its perception of the game.*

method	residual error (mm) image 1	image 2
linear	7.3441	1.1402
Dementhon	1.189	5.6942
LM	1.1425	0.7641
VVS	1.1316	0.7463

Table 1: *Pose (viewpoint) computation: comparison between the virtual visual servoing approach and three other classical methods (linear pose computation, Dementhon [6], and a non linear minimization by the Levenberg Marquartd approach (LM)). We show in this table the residual error computed for each method in two different images. The pattern was made of four coplanar points.*

numerical iterative estimation by Dementhon's Algorithm [6], non-linear estimation using a Levenberg-Marquartd minimization scheme (as proposed in [16] but restricted to point features). The target is made of four coplanar points. We consider in this experiment two images. For the second one, the image plane is nearly parallel to the target plane. Table 1 displays the mean error ($\|\mathbf{p} - \mathbf{p_d}\|$) for these various methods. As expected, the VVS approach is similar to the non-linear minimization by Levenberg-Marquart [16] but is far more efficient than the linear method or the Dementhon's algorithm. A more complete analysis of the algorithm behavior is proposed in [20] for the calibration problem.

Augmented reality using precise patterns. To begin with we report augmented reality experiments that use precise patterns to estimate the camera viewpoint. Such experiments show that this approach may be efficiently used in interactive application such as (but not restricted to) a collaborative immersive workplace, cultural heritage or architecture interactive visualization interactive video game (by moving and orienting a simple pattern the player may modify on line its perception of the game).

- Figure 2.a shows the efficiency of the algorithm along a 1600 image sequence acquired at video rate. Pose is computed from four points (that appear in green) and one circle (at the middle of the four points). Image processing and pose computation are handled in real-time that is 25Hz (pose itself is computed in 3 ms). The statue and the textured polygon that "augment" the initial images are very stable along the entire sequence.
- in the next experiment (see Figure 2.b), we show that a limited number of features are required to compute the pose. Indeed, the pose related to *"ghost"* is computed using two circles (the first image depicts the result of the image processing where the related circles appear in green). An other object, a "virtual graveyard", is also inserted. Therefore, an other pose is computed for this object using four points (the center of each black dot). The two poses and the rendering of the scene is handled at video rate.

Figure 3: *Pose computation and augmented reality in real environment without any landmarks. Since reliable points features are difficult to track in real environment, it is often useful to consider various features such as circles, lines or cylinder. In these experiments features are tracked using the moving edges algorithm in less than 5 ms. The whole process, tracking, pose computation, and rendering is achieved in real-time (i.e., 25Hz). In (a) circles and lines are considered in while in (b), (c) and (d) cylinders and lines are used. Figures (d) illustrates various cases of partial occlusions. This illustrates the versatility of the virtual visual servoing process wrt. the choice of the features extracted from the image.*

Augmented reality in non controlled situations. We then consider "real" images acquired using a commercial recorder. In such experiments, the image processing may be very complex. Indeed extracting and tracking reliable points in real environment is a real issue. Therefore it is important to consider image features other than simple points. We demonstrate the use of circles, lines, and cylinders. In various experiments, the features are tracked using the Moving edges algorithm [18] at video rate.

- In the first experiment (see results in Figure 3.a), the visual features considered in the pose computation process are lines and circles. Specificaly, four non-coplanar lines and three circles (on the door) are considered to compute the pose. The model of the scene has been roughly hand made. The features are tracked along a 60 image sequence and the images have been successfully augmented with various objects. The left image of Figure 3.a displaies the tracked lines and cylinder limbs as well as 3D information inserted after the pose computation.
- In Figure 3.b a context that can be encountered in an industrial environment is considered. A camera is mounted on the end-effector of a 6 d.o.f robot. The primitives considered in the virtual servoing process are two cylinders and a straight line. Pose computation is performed in less than 3 ms (except in the very first image). Note that some features disappear over time. The size of the interaction matrix and of the error vector is modified consequently.
- In the third experiment, an outdoor scene is considered. Here again, a cylinder and two lines are used to compute the pose. Despite very noisy images (wind in the trees, etc.) tracking is achieved along a 700 and 1400 image sequences. The left image on Figure 3.c displaies the tracked lines and cylinder limbs as well as the 3D information inserted after the pose computation (the reference frame and the projection of the cylinder). The other images are extracted from the sequence after the insertion of virtual objects.
- The fourth experiment features is a similar experiment but features are only partially visible from the camera (left and right image). The algorithm still generates (up to a certain limit) acceptable results. On the right image note that explicit occlusions is not handled (this was not considered in the scope of this paper).

Augmented reality with unknown focal length. In the sequence presented in Figure 4, important variations are introduced in the focal length of the camera (focal length has been divided by nearly two and then increased again, see Figure 4.b. Furthermore the initial camera parameters are unknown. Our goal is to compute the camera parameters u_0, v_0, p_x, p_y considering both points and ellipses as image features. Three points and three circles are tracked along an image sequence. Due to the low number of features considered in the experiment, the quality of the estimation of the camera parameters is of course less accurate than considering a real off-line calibration approach (although this can be achieved as shown in [20]. However, the quality of the obtained results is far sufficient for AR.

6. Conclusion

We proposed in this paper an original formulation of pose computation and its application to augmented reality. It consisted of modifying the parameters of a virtual camera (position, orientation, and if necessary intrinsic parameters) using the visual servoing paradigm in order to register the back projection of the model with the data extracted from the images. We presented experimental results obtained using several cameras, lens, and environments. This approach features many advantages:

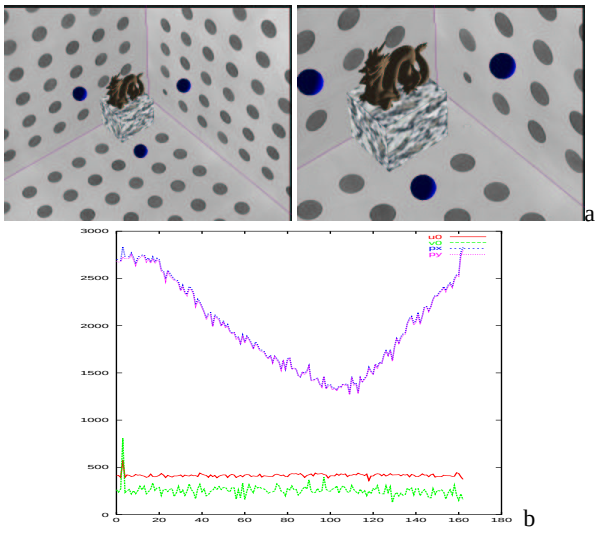

Figure 4: *AR with on-line calibration from points and circles: (a) augmented scene, (b) value of the camera parameters (note that p_x and p_y, that reflect the focal length, are modified)*

- First of all, this algorithm is really simple. In [20], we give a 20 lines source code of the virtual servoing closed-loop that computes the pose proposed in this paper;
- Even with this simplicity, the obtained results may be favorably compared to the best algorithms currently available. In particular the computed pose is far better (considering the residual error) than linear algorithm.
- Although it is an iterative minimization process, pose computation may be handled in real-time.
- Finally, it allows consideration of various geometrical features within the same registration process. The resulting interaction matrix (or image Jacobian) is straightforward to derive thanks to the wide visual servoing literature. The choice of adequate features may allow reduction of the number of landmarks to be tracked in the image.

Demonstration on-line. Most of the demostrations presented in this paper can be found as mpeg film on the WWW page (`http://www.irisa.fr/prive/marchand` then follow the "demo" link). A pseudo code of pose computation algorithm is also given. **Contact:** `Eric.marchand@irisa.fr`.

References

1. R. Azuma. A survey of augmented reality. *Presence: Teleoperators and Virtual Environments*, 6(4):355–385, August 1997. 1

2. R. Azuma, Y. Baillot, R. Behringer, S. Feiner, S. Julier, and B. MacIntyre. Recent advances in augmented reality. *IEEE Computer Graphics and Application*, 21(6):34–47, Novembre 2001. 1

3. M. Billinghurst, H. Kato, and I. Poupyrev. The magicbook: Moving seamlessly between reality and virtuality. *IEEE Computer Graphics and Applications*, 21(3):6–8, May 2001. 5

4. S. de Ma. Conics-based stereo, motion estimation and pose determination. *Int. Journal of Computer Vision*, 10(1):7–25, 1993. 1

5. P. Debevec, C.-J. Taylor, and J. Malik. Modeling and rendering architecture from photographs: A hybrid geometry- and image-based approach. In *Proceedings of SIGGRAPH 96*, Computer Graphics Proceedings, Annual Conference Series, pages 11–20, New Orleans, Louisiana, August 1996. 2

6. D. Dementhon and L. Davis. Model-based object pose in 25 lines of codes. *Int. J. of Computer Vision*, 15:123–141, 1995. 1, 6

7. M. Dhome, J.-T. Lapresté, G. Rives, and M. Richetin. Determination of the attitude of modelled objects of revolution in monocular perspective vision. In *European Conference on Computer Vision, ECCV'90*, volume LNCS 427, pages 475–485, Antibes, April 1990. 1

8. M. Dhome, M. Richetin, J.-T. Lapresté, and G. Rives. Determination of the attitude of 3-d objects from a single perspective view. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 11(12):1265–1278, December 1989. 1

9. T. Drummond and R. Cipolla. Real-time tracking of complex structures with on-line camera calibration. In *proc. of the British Machine Vision Conference, BMVC 99*, 1999. 1

10. B. Espiau, F. Chaumette, and P. Rives. A new approach to visual servoing in robotics. *IEEE Trans. on Robotics and Automation*, 8(3):313–326, June 1992. 2, 4

11. R. Haralick, H. Joo, C. Lee, X. Zhuang, V Vaidya, and M. Kim. Pose estimation from corresponding point data. *IEEE Trans on Systems, Man and Cybernetics*, 19(6):1426–1445, November 1989. 1

12. K. Hashimoto. *Visual Servoing : Real Time Control of Robot Manipulators Based on Visual Sensory Feedback*. World Scientific Series in Robotics and Automated Systems, Vol 7, World Scientific Press, Singapor, 1993. 2

13. S. Hutchinson, G. Hager, and P. Corke. A tutorial on visual servo control. *IEEE Trans. on Robotics and Automation*, 12(5):651–670, October 1996. 2, 4

14. H. Kato, M. Billinghurst, I. Poupyrev, K. Imamoto, and K. Tachibana. Virtual object manipulation on a table-top ar environment. In *Proceedings of Int. Symp. on Augmented Reality 2000*, October 2000. 5

15. K. Kutulakos and J. Vallino. Calibration-free augmented reality. *IEEE Transactions on Visualization and Computer Graphics*, 4(1):1–20, january 1998. 1

16. D.G. Lowe. Three-dimensional object recognition from single two-dimensional images. *Artificial Intelligence*, 31:355–394, 1987. 1, 4, 6

17. C.P. Lu, G.D. Hager, and E. Mjolsness. Fast and globally convergent pose estimation from video images. *PAMI*, 22(6):610–622, June 2000. 1

18. E. Marchand. Visp: A software environment for eye-in-hand visual servoing. In *IEEE Int. Conf. on Robotics and Automation, ICRA'99*, volume 4, pages 3224–3229, Detroit, Michigan, Mai 1999. 5, 7

19. E. Marchand, P. Bouthemy, F. Chaumette, and V. Moreau. Robust real-time visual tracking using a 2d-3d model-based approach. In *IEEE Int. Conf. on Computer Vision, ICCV'99*, volume 1, pages 262–268, Kerkira, Greece, September 1999. 5

20. E. Marchand and F. Chaumette. A new formulation for non-linear camera calibration using virtual visual servoing. Technical Report 4096, INRIA, January 2001. 5, 6, 8

21. E. Marchand and N. Courty. Image-based virtual camera motion strategies. In S. Fels and P. Poulin, editors, *Graphics Interface Conference, GI2000*, pages 69–76, Montreal, Quebec, May 2000. Morgan Kaufmann. 2

22. T.-Q. Phong, R. Horaud, A. Yassine, and P.-D. Tao. Object pose from a 2d to 3d point and line correspondance. *Int. J. of Computer Vision*, 15(3):225–243, July 1995. 1

23. R. Safaee-Rad, I. Tchoukanov, B. Benhabib, and K.C. Smith. 3d-pose estimation from a quadratic-curved feature in two perspective views. In *IAPR Int. Conf. on Pattern Recognition, ICPR'92*, volume 1, pages 341–344, La Haye, Pays Bas, August 1992. 1

24. G. Simon and M.-O. Berger. Registration with a zoom lens camera for augmented reality applications. In *Proceedings of the 2nd International Workshop on Augmented Reality*, San Francisco, 1998. 1

25. V. Sundareswaran and R. Behringer. Visual servoing-based augmented reality. In *IEEE Int. Workshop on Augmented Reality*, San Francisco, November 1998. 2

26. R. Torre, P. Fua, S. Balcisoy, M. Ponder, and D. Thalmann. Interaction between real and virtual humans: Playing checkers. In J.-D. Mulder and R. Van Liere, editors, *Proc. Eurographics Workshop On Virtual Environments 2000*, Amsterdam, The Netherlands, june 2000. 5