Visual servoing with indirect image control and a predictable camera trajectory

Enric Cervera, Philippe Martinet

Visual Servoing with Indirect Image Control and a Predictable Camera Trajectory

E. Cervera

Robotic Intelligence Lab.
Jaume-I University
12071 Castelló, Spain

P. Martinet

LASMEA
Blaise Pascal University of Clermont-Ferrand
63177 Aubière - Cedex, France

Abstract

Neither of the classical visual servoing approaches, position-based and image-based, are completely satisfactory. This paper presents a different approach with some advantages of both, i.e. the trajectory of the camera motion is predictable and the image features remain in the field of view of the camera. Our new approach is based on the computation of the pose of the object from the image, thus an appropriate calibration of the camera and a geometric model of the object are required. Experimental results on a real robotic platform are presented.

1 Introduction

Classically, visual servoing systems are divided in two categories [8, 4], position-based and image-based, which differ in the nature of the inputs used in their respective control schemes: Cartesian frames and image features respectively.

However, neither of them is completely satisfactory. As explained in [1], in the position-based scheme it is impossible to ensure that the object will always remain in the camera field of view during the servoing task. On the other hand, image-based control may imply unpredictable inadequate camera motion, leading to possible local minima and the nearing of task singularities.

The motivation of our work is the development of a new visual servoing method with the advantages of both schemes: Guarantee that the object remains in the field of view of the camera, Prediction of the motion of the camera, and Inexistence of local minima or singularities.

Our approach is based on the extraction of the object pose from the image. Thus, the main requirement is the existence of a model of the object. Pose computation is no longer a disadvantage, since fast algorithms [2] are available.

Since our model is pose-based, it is free of local minima or singularities at control level, such difficulties being reported on the pose estimation algorithm. Additionally, the input is the pose of the object, resulting in an *indirect* image control. As we will demonstrate, the object will remain in the camera field of view, in the absence of calibration errors. The trajectory of the object in the camera frame can be easily determined, which in turn allows the off-line computation of the trajectory of the camera, as long as that the object remains static.

The difference between the presented approach and other pose-based schemes [9, 6] is the coordinate frame with respect to which the task is defined. The proposed frame will allow an easier computation of the trajectories, as shown in Section 2.

The following development is founded on the *task function approach*, which was applied to visual servoing in [3]. For extensive information about this approach, the interested reader may refer to [7].

In the following, the reader is assumed to be familiar with coordinate frames and transformations. A brief overview of these concepts in the context of visual servoing is given in [4]. Since our notation slightly differs from the one proposed in the mentioned paper, the main concepts are remarked below.

1.1 Notation

A frame a is represented by $\mathcal{F}_a$ and the transformation between frames a and b is ${}^a\mathbf{T}_b$, which also specifies the *pose*, i.e. the position and orientation, of frame b with respect to frame a.

Therefore, a transformation consists of a rotation component ${}^a\mathbf{R}_b$ which denotes the orientation between

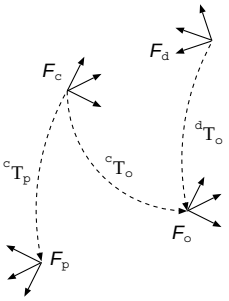

Figure 1: Frames defined in the visual servoing task.

frames and a translation component ${}^a\mathbf{t}_b$ which denotes the position of the origin of one frame with respect to the other.

In the development, we parametrize a rotation by the product $\mathbf{u}\theta$ of the corresponding axis and angle. Sometimes, we need to come back to a rotation matrix which will be represented as $\mathbf{R}(\mathbf{u}\theta)$.

A *velocity screw* in frame a consists of a translational and an angular velocity in the referred frame, and it is represented by ${}^a\mathbf{v} = ({}^a\boldsymbol{\nu}^t, {}^a\boldsymbol{\omega}^t)^t$.

Finally, cross products are represented in matrix form,

$$\mathbf{a} \times \mathbf{b} = [\mathbf{a}]_\times \mathbf{b} \ , \ [\mathbf{a}]_\times = \begin{bmatrix} 0 & -a_z & a_y \\ a_z & 0 & -a_x \\ -a_y & a_x & 0 \end{bmatrix} \quad (1)$$

with $[\mathbf{a}]_\times$ the so-called *skew-symmetric* matrix

2 Indirect image control with a predictable camera trajectory

Let $\mathcal{F}_o$ be the coordinate frame attached to the observed object, $\mathcal{F}_c$ and $\mathcal{F}_d$ the coordinate frames associated to the current and desired camera poses, as depicted in Figure 1. Let ${}^c\mathbf{T}_o$ and ${}^d\mathbf{T}_o$ be the transformations between the camera and object frames, for the current and desired camera poses respectively. Let us define a new frame $\mathcal{F}_p$, rigidly attached to the current camera frame, as:

$${}^c\mathbf{T}_p = {}^d\mathbf{T}_o \quad (2)$$

that is, the transformation between the current camera frame and the defined one is the same as that of the desired camera frame and the object frame.

The key is to define the error vector in the new frame, instead of defining it in the camera frame as usual. The pose of the object is then:

$${}^p\mathbf{T}_o = ({}^c\mathbf{T}_p)^{-1}\,{}^c\mathbf{T}_o = \left({}^d\mathbf{T}_o\right)^{-1}\,{}^c\mathbf{T}_o \quad (3)$$

which can be calculated since ${}^d\mathbf{T}_o$ is given, and ${}^c\mathbf{T}_o$ is reconstructed from the current image and the model of the object.

Let

$${}^p\mathbf{T}_o = \begin{bmatrix} \mathbf{R}(\mathbf{u}\theta) & {}^p\mathbf{t}_o \\ 0 & 1 \end{bmatrix} \quad (4)$$

be the decomposition of the transformation matrix in its rotation and translation components. Then let us define the feature vector as

$$\mathbf{s} = \begin{bmatrix} {}^p\mathbf{t}_o \\ \mathbf{u}\theta \end{bmatrix} \quad (5)$$

and the error vector as

$$\mathbf{e} = \mathbf{s} - \mathbf{s}^* \quad (6)$$

where $\mathbf{s}^*$ is the desired feature vector. However, when the camera achieves the desired pose, frames $\mathcal{F}_p$ and $\mathcal{F}_o$ are coincident, thus $\mathbf{s}^* = \mathbf{0}$ and

$$\mathbf{e} = \mathbf{s} = \begin{bmatrix} {}^p\mathbf{t}_o \\ \mathbf{u}\theta \end{bmatrix} \quad (7)$$

If an exponential decoupled convergence of the task function is imposed then

$$\dot{\mathbf{e}} = -\lambda \mathbf{e} \quad (8)$$

which is a vectorial differential equation whose solution is

$$\mathbf{e}(t) = \mathbf{e}(0) e^{-\lambda t} \quad (9)$$

and, following the definition of the error,

$$\begin{aligned} {}^p\mathbf{t}_o(t) &= {}^p\mathbf{t}_o(0) e^{-\lambda t} \\ \mathbf{u}\theta(t) &= \mathbf{u}\theta(0) e^{-\lambda t} \end{aligned} \quad (10)$$

where

$$\begin{aligned} {}^p\mathbf{t}_o(0) &= {}^p\mathbf{R}_c\,{}^c\mathbf{t}_o(0) + {}^p\mathbf{t}_c \\ &= ({}^d\mathbf{R}_o)^{-1}\,{}^c\mathbf{t}_o(0) + {}^o\mathbf{t}_d \\ &= ({}^d\mathbf{R}_o)^{-1}\,{}^c\mathbf{t}_o(0) - ({}^d\mathbf{R}_o)^{-1}\,{}^d\mathbf{t}_o \\ &= ({}^d\mathbf{R}_o)^{-1} \left({}^c\mathbf{t}_o(0) - {}^d\mathbf{t}_o\right) \end{aligned} \quad (11)$$

and where $\mathbf{u}\theta(0)$ are the axis and the angle which correspond to the rotation matrix:

$${}^p\mathbf{R}_o(0) = ({}^c\mathbf{R}_p)^{-1}\,{}^c\mathbf{R}_o(0) = ({}^d\mathbf{R}_o)^{-1}\,{}^c\mathbf{R}_o(0) \quad (12)$$

2.1 The trajectory of the object in the camera frame

The trajectory of the object in the camera frame ${}^c\mathbf{t}_o(t)$ is very important, since such trajectory

determines whether the object remains within the vision field of the camera or not. Thus,

$$ {}^c\mathbf{t}_o(t) = {}^c\mathbf{R}_p\, {}^p\mathbf{t}_o(t) + {}^c\mathbf{t}_p = {}^c\mathbf{R}_p\, {}^p\mathbf{t}_o(0) e^{-\lambda t} + {}^c\mathbf{t}_p \quad (13) $$

where it should be noted that both ${}^c\mathbf{R}_p$ and ${}^c\mathbf{t}_p$ are constant since the frame $\mathcal{F}_p$ is rigidly attached to the camera frame.

When $t = 0$,

$$ {}^p\mathbf{t}_o(0) = {}^p\mathbf{R}_c \left({}^c\mathbf{t}_o(0) - {}^c\mathbf{t}_p \right) \quad (14) $$

Then,

$$ \begin{aligned} {}^c\mathbf{t}_o(t) &= {}^c\mathbf{R}_p\, {}^p\mathbf{R}_c \left({}^c\mathbf{t}_o(0) - {}^c\mathbf{t}_p \right) e^{-\lambda t} + {}^c\mathbf{t}_p \\ &= \left({}^c\mathbf{t}_o(0) - {}^d\mathbf{t}_o \right) e^{-\lambda t} + {}^d\mathbf{t}_o \end{aligned} \quad (15) $$

which is simply a straight line trajectory from the initial position ${}^c\mathbf{t}_o(0)$ to the desired one ${}^d\mathbf{t}_o$, the exponential factor only affecting the velocity along such line.

The orientation of the object with respect to the camera frame can also be obtained. From Equation 10,

$$ {}^p\mathbf{R}_o(t) = \mathbf{R}\left(\mathbf{u}\theta(0)e^{-\lambda t}\right) \quad (16) $$

and by changing the coordinate frame,

$$ {}^c\mathbf{R}_o(t) = {}^c\mathbf{R}_p\, {}^p\mathbf{R}_o(t) = {}^d\mathbf{R}_o \mathbf{R}\left(\mathbf{u}\theta(0)e^{-\lambda t}\right) \quad (17) $$

which is a composition of the constant rotation ${}^d\mathbf{R}_o$ and the variable rotation around the constant axis $\mathbf{u}$, as given in Equation 12. In all of the above development, no calibration have been taken into account.

2.2 Image point trajectories

In the proposed framework, image features (points) are not directly controlled. However, since the trajectory of the object in the camera frame is known, it is possible to calculate the trajectory of each point in the camera frame and even its trajectory in the image plane.

If the angle of rotation θ between the current and desired poses of the object with respect to the camera is small, then we will demonstrate that each object point follows a straight line trajectory in 3D space from its initial to its final position. Thus, the projection of each point describes a straight line from its initial to its final position in the image plane too.

The trajectory of a point i, which is rigidly attached to the object, is

$$ {}^c\mathbf{t}_i(t) = {}^c\mathbf{R}_o(t)\, {}^o\mathbf{t}_i + {}^c\mathbf{t}_o(t) \quad (18) $$

where ${}^o\mathbf{t}_i$ is the position of such point in the object frame. Since the point is rigidly attached, this position is constant.

From Equations 15 and 17

$$ \begin{aligned} {}^c\mathbf{t}_i(t) &= {}^d\mathbf{R}_o \mathbf{R}(\mathbf{u}\theta(0)e^{-\lambda t})\, {}^o\mathbf{t}_i \\ &\quad + ({}^c\mathbf{t}_o(0) - {}^d\mathbf{t}_o)e^{-\lambda t} + {}^d\mathbf{t}_o \end{aligned} \quad (19) $$

A rotation matrix with axis $\mathbf{u}$ and angle θ can be expressed as

$$ \mathbf{R}(\mathbf{u}\theta) = \mathbf{I}_3 + [\mathbf{u}]_\times^2 \left(1 - \cos\theta\right) + [\mathbf{u}]_\times \sin\theta \quad (20) $$

which, if θ is small, leads to,

$$ \mathbf{R}(\mathbf{u}\theta) \approx \mathbf{I}_3 + [\mathbf{u}]_\times\, \theta \quad (21) $$

Assuming such approximation, Equation 17 becomes

$$ {}^c\mathbf{R}_o(t) \approx {}^d\mathbf{R}_o + {}^d\mathbf{R}_o\, [\mathbf{u}]_\times\, \theta(0)e^{-\lambda t} \quad (22) $$

Thus, the trajectory of the object point is

$$ \begin{aligned} {}^c\mathbf{t}_i(t) &\approx {}^d\mathbf{R}_o \left(\mathbf{I}_3 + [\mathbf{u}]_\times\, \theta(0)e^{-\lambda t}\right) {}^o\mathbf{t}_i \\ &\quad + ({}^c\mathbf{t}_o(0) - {}^d\mathbf{t}_o)e^{-\lambda t} + {}^d\mathbf{t}_o \\ &\approx {}^d\mathbf{R}_o\, {}^o\mathbf{t}_i + {}^d\mathbf{t}_o \\ &\quad + \left(({}^c\mathbf{R}_o(0) - {}^d\mathbf{R}_o)\, {}^o\mathbf{t}_i + {}^c\mathbf{t}_o(0) - {}^d\mathbf{t}_o\right) e^{-\lambda t} \\ &\approx {}^d\mathbf{t}_i + \left({}^c\mathbf{t}_i(0) - {}^d\mathbf{t}_i\right) e^{-\lambda t} \end{aligned} \quad (23) $$

which is effectively a straight line from its initial to its final position in the camera frame. Consequently, its projection is a straight line joining its initial and final position in the image plane.

From the point of view of the image, such behavior is the same as the one produced by classic image-based visual servoing, where the task error is defined in terms of image points. In this last approach, image points are constrained to follow straight lines even in the pressence of high orientation variations, thus frequently causing inadequate camera motions [1].

2.3 Advantages of the proposed coordinate frame

The key advantage of the proposed framework is that the trajectory of the object in the camera frame is known, and the object is likely to remain within the camera field (in the absence of high calibration and/or model errors).

However, one might wonder about the need of introducing a new coordinate frame $\mathcal{F}_p$ instead of using directly the current camera frame $\mathcal{F}_c$.

Effectively, if the error vector is defined in $\mathcal{F}_c$, then the trajectory of the object is exactly the same as given in Equation 15.

The difference between choosing either of the two frames concerns only the orientation of the object. If the error vector is defined in the camera frame, then $\mathbf{s}^*$ is no longer null, and:

$$\mathbf{e} = \begin{bmatrix} {}^c\mathbf{t}_o - {}^d\mathbf{t}_o \\ \mathbf{u}_c\theta_c - \mathbf{u}_d\theta_d \end{bmatrix} \tag{24}$$

where $\mathbf{u}_c\theta_c$ and $\mathbf{u}_d\theta_d$ are respectively the axes and angles which correspond to the rotation matrices ${}^c\mathbf{R}_o(0)$ and ${}^d\mathbf{R}_o$.

Solving the differential equation leads to:

$$\mathbf{u}\theta(t) = \mathbf{u}_c\theta_c e^{-\lambda t} + (1 - e^{-\lambda t})\mathbf{u}_d\theta_d \tag{25}$$

The problem now is that, in the general case, the orientation of the axis of rotation changes, since it is a linear combination of the two not-necessarily-parallel vectors $\mathbf{u}_c$ and $\mathbf{u}_d$. Only if these vectors are parallel or any of them is null, then the axis of rotation is constant.

This is clearly a disadvantage of the camera frame, which motivates the introduction of frame $\mathcal{F}_p$. A constant rotation axis not only produces simpler trajectories, but it also allows the determination of the trajectories of object points, as described before, and an easier calculation of the velocity screw, as shown in next section.

2.4 Control law

Let us calculate the control law which achieves the desired exponential decoupled convergence, as imposed in Equation 8. Using the task function approach [7], the velocity screw is,

$$\mathbf{v} = -\lambda \mathbf{L}_s^{-1}\mathbf{e} \tag{26}$$

where $\mathbf{L}_s$ is the *interaction matrix* which relates the derivatives of the feature vector and the velocity screw:

$$\dot{\mathbf{s}} = \mathbf{L}_s\mathbf{v} \tag{27}$$

The interaction matrix for the defined feature vector is,

$$\mathbf{L}_s = \begin{bmatrix} -\mathbf{I}_3 & [{}^p\mathbf{t}_o]_\times \\ \mathbf{0} & -\mathbf{L}_\omega \end{bmatrix} \tag{28}$$

where $\mathbf{L}_\omega$ is the submatrix which relates the rotational components:

$$\frac{d(\mathbf{u}\theta)}{dt} = -\mathbf{L}_\omega\boldsymbol{\omega} \tag{29}$$

The complete derivation of such submatrix can be found in [5], and the result is:

$$\mathbf{L}_\omega = \mathbf{I} - \frac{\theta}{2}[\mathbf{u}]_\times + (1 - \frac{\mathrm{sinc}(\theta)}{\mathrm{sinc}^2(\frac{\theta}{2})})[\mathbf{u}]_\times^2 \tag{30}$$

where the function $\mathrm{sinc}(\theta)$, named *sinus cardinal*, is defined as

$$\mathrm{sinc}(\theta) = \frac{\sin\theta}{\theta} \tag{31}$$

The inverse of the interaction matrix is

$$\mathbf{L}_s^{-1} = \begin{bmatrix} -\mathbf{I}_3 & -[{}^p\mathbf{t}_o]_\times \mathbf{L}_\omega^{-1} \\ \mathbf{0} & -\mathbf{L}_\omega^{-1} \end{bmatrix} \tag{32}$$

where

$$\mathbf{L}_\omega^{-1} = \mathbf{I} + \frac{\theta}{2}\mathrm{sinc}^2\left(\frac{\theta}{2}\right)[\mathbf{u}]_\times + (1 - \mathrm{sinc}(\theta))[\mathbf{u}]_\times^2 \tag{33}$$

Originally, the error vector has been defined in frame $\mathcal{F}_p$, thus the velocity screw is calculated in the same coordinate frame:

$$\begin{aligned} {}^p\mathbf{v} &= -\lambda \begin{bmatrix} -\mathbf{I}_3 & -[{}^p\mathbf{t}_o]_\times \mathbf{L}_\omega^{-1} \\ \mathbf{0} & -\mathbf{L}_\omega^{-1} \end{bmatrix} \begin{bmatrix} {}^p\mathbf{t}_o \\ \mathbf{u}\theta \end{bmatrix} \\ &= -\lambda \begin{bmatrix} -{}^p\mathbf{t}_o - [{}^p\mathbf{t}_o]_\times \mathbf{L}_\omega^{-1}\mathbf{u}\theta \\ -\mathbf{L}_\omega^{-1}\mathbf{u}\theta \end{bmatrix} \end{aligned} \tag{34}$$

Since $[\mathbf{u}]_\times \mathbf{u} = 0$ and after Equation 33,

$$\mathbf{L}_\omega^{-1}\mathbf{u}\theta = \mathbf{u}\theta \tag{35}$$

and, consequently,

$${}^p\mathbf{v} = -\lambda \begin{bmatrix} -{}^p\mathbf{t}_o - [{}^p\mathbf{t}_o]_\times \mathbf{u}\theta \\ -\mathbf{u}\theta \end{bmatrix} \tag{36}$$

One should note that the precedent simplification cannot be applied in the camera frame $\mathcal{F}_c$, as indicated in Section 2.3 since, in the general case, $[\mathbf{u}_c]_\times \mathbf{u}_d$ is not null.

Let us calculate now the velocity screw in the camera frame:

$$\begin{aligned} {}^c\mathbf{v} &= -\lambda \begin{bmatrix} {}^c\mathbf{R}_p\left(-{}^p\mathbf{t}_o - [{}^p\mathbf{t}_o]_\times \mathbf{u}\theta\right) - [{}^c\mathbf{t}_p]_\times {}^c\mathbf{R}_p\mathbf{u}\theta \\ -{}^c\mathbf{R}_p\mathbf{u}\theta \end{bmatrix} \\ &= -\lambda \begin{bmatrix} -{}^c\mathbf{t}_o + {}^c\mathbf{t}_p - [{}^c\mathbf{t}_o - {}^c\mathbf{t}_p + {}^c\mathbf{t}_p]_\times {}^c\mathbf{R}_p\mathbf{u}\theta \\ -{}^c\mathbf{R}_p\mathbf{u}\theta \end{bmatrix} \\ &= -\lambda \begin{bmatrix} -{}^c\mathbf{t}_o + {}^d\mathbf{t}_o - [{}^c\mathbf{t}_o]_\times {}^d\mathbf{R}_o\mathbf{u}\theta \\ -{}^d\mathbf{R}_o\mathbf{u}\theta \end{bmatrix} \end{aligned} \tag{37}$$

where $\mathbf{u}\theta$ are the axis and angle the angle which correspond to the rotation matrix

$$\mathbf{R}(\mathbf{u}\theta) = {}^p\mathbf{R}_c\,{}^c\mathbf{R}_o = ({}^d\mathbf{R}_o)^{-1}\,{}^c\mathbf{R}_o \tag{38}$$

The computation of the control law consists of three steps:

1. Obtain ${}^c\mathbf{t}_o$ and ${}^c\mathbf{R}_o$ from the current image and the model of the observed object with a reconstruction algorithm (e.g. a fast algorithm is proposed in [2]).

2. Calculate $\mathbf{u}\theta$ as shown in Equation 38.

3. Compute the velocity screw in the camera frame ${}^c\mathbf{v}$ as given by Equation 37. This screw can be readily transformed to the end-effector or base frame and, via the Jacobian of the robot, to joint velocities.

As explained in the literature [7, 1, 5], the system is globally asymptotically stable if $\widehat{\mathbf{L}}_s^{-1}\mathbf{L}_s > 0$ where $\widehat{\mathbf{L}}_s^{-1}$ is an estimation based on the current measurements. This condition is met since, *in the absence of calibration errors*, $\widehat{\mathbf{L}}_s^{-1}\mathbf{L}_s = \mathbf{I}_6$. The determination of stability bounds for camera and/or model errors is the subject of our current research.

2.5 The trajectory of the camera

Since the object is supposed static, then:

$$\mathbf{t}_c(t) = \mathbf{R}_o{}^o\mathbf{t}_c(t) + \mathbf{t}_o \tag{39}$$

where $\mathbf{t}_o$ and $\mathbf{R}_o$ are respectively the position and orientation of the object with respect to an absolute frame. It follows immediately that,

$$\mathbf{t}_c(t) = -\mathbf{R}_o{}^o\mathbf{R}_c(t){}^c\mathbf{t}_o(t) + \mathbf{t}_o \tag{40}$$

which allows to calculate the trajectory of the camera in terms of the trajectory of the object in the camera frame and the pose of the object on the absolute frame.

Since the trajectory of the object in the camera frame is described by Equations 15 and 17, the trajectory of the camera can be completely expressed in terms of the task parameters $({}^c\mathbf{T}_o(0), {}^d\mathbf{T}_o, \mathbf{T}_o)$:

$$\begin{aligned} \mathbf{t}_c(t) &= -\mathbf{R}_c(t)\left(\left({}^c\mathbf{t}_o(0) - {}^d\mathbf{t}_o\right)e^{-\lambda t} + {}^d\mathbf{t}_o\right) + \mathbf{t}_o \\ \mathbf{R}_c(t) &= \mathbf{R}_o\mathbf{R}(\mathbf{u}\theta e^{-\lambda t})({}^d\mathbf{R}_o)^{-1} \end{aligned} \tag{41}$$

where $\mathbf{u}\theta$ are the axis and the angle which correspond to the rotation matrix $({}^c\mathbf{R}_o(0))^{-1}{}^d\mathbf{R}_o$.

3 Experimental results

The control law has been implemented on a robotic platform with 6 degrees of freedom and an eye-in-hand configuration. The target object is composed of four points which define a tetrahedron. The pose of the object is extracted from the images and an internal model with Dementhon's algorithm [2]. Initial and desired poses of the camera are shown in Table 1.

Pose	Translation (mm)	Rotation $\mathbf{u}\theta$ (deg)
Initial	0 0 -500	0 0 0
Desired	-304 251 -366	-6 26 -50

Table 1: Initial and desired poses of the camera.

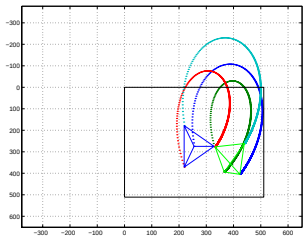

Figure 2: Trajectory of the object in the image plane with position-based visual servoing (simulation).

The proposed control law has been experimented together with position-based and image-based ones. However, the classic approaches do not converge and simulated results are provided. The position-based approach failed due to the object going out of the camera field of view (see Figure 2). On the other hand, in both image-based visual servoing and our approach, the object remained always in the camera field of view (see Figures 3 and 4). But image-based visual servoing failed due to the imposed camera trajectory, which pushed the robot farther from its physical joint limits.

Figure 5 depicts the trajectory of the camera in an absolute frame. In position-based visual servoing, the trajectory is ideally a straight line, but convergence is not achieved, as explained before. In the image-based approach, the trajectory goes farther from the joint limits, thus it fails too. Finally, with the proposed model the trajectory is near to the straight path, and convergence to the desired pose is achieved.

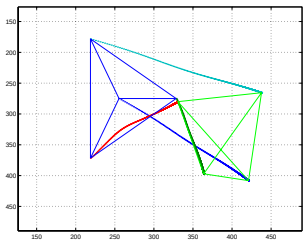

Figure 3: Trajectory of the object in the image plane with image-based visual servoing (simulation).

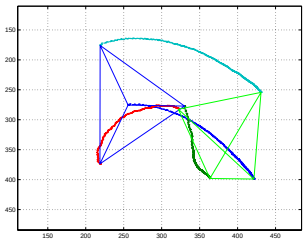

Figure 4: Trajectory of the object in the image plane with the proposed approach (real robot).

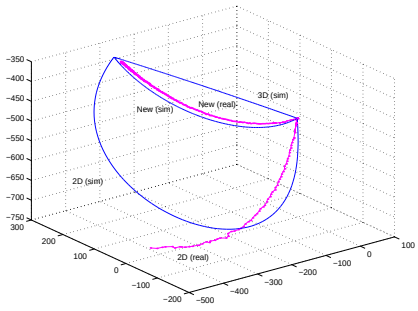

Figure 5: Trajectories of the camera with image-based (2D), position-based (3D) and the proposed visual servoing approaches.

The differences between real and simulated trajectories are due to the calibration errors of the real platform which cannot be completely captured in the simulations.

4 Summary and Conclusions

A new visual servoing model has been presented, which inherits the advantages from position-based and image-based approaches, namely the camera trajectory is predictable and the image features remain in the camera field of view. In addition, convergence is achieved for any configuration of the camera and object provided that the pose can be correctly reconstructed from the object image and model.

The existence of a model of the object is the most constraining requirement, and the influence of the accuracy of such model in the task has to be studied, as well as the influence of calibration errors.

Experimental results on a robotic platform show the feasibility of the proposed scheme in a real-world environment. An open issue is the study of its robustness and efficiency—stability and convergence—with respect to control, model and calibration errors.

Acknowledgements

Support from Fundació Caixa Castelló - Bancaixa for a temporal stay of E. Cervera in Blaise Pascal University is gratefully acknowledged.

References

[1] F. Chaumette. Potential problems of stability and convergence in image-based and position-based visual servoing. In G. Hager, D. Kriegman, and A. Morse, editors, *The Confluence of Vision and Control*. Springer Verlag, 1998.

[2] D. F. Dementhon and L. S. Davis. Model-based object pose in 25 lines of code. *International Journal of Computer Vision*, 15(1/2):123–141, June 1995.

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

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

[5] E. Malis. *Contributions à la Modélisation et à la Commande en Asservissement Visuel*. PhD thesis, Université de Rennes, France, November 1998.

[6] P. Martinet, N. Daucher, J. Gallice, and M. Dhome. Robot control using monocular pose estimation. In *Workshop on New Trends in Image-Based Visual Servoing (IROS'97)*, pages 1–12, Grenoble, France, September 1997.

[7] C. Samson, M. Le Borgne, and B. Espiau. *Robot Control: the Task Function Approach*, volume 22 of *Oxford Engineering Science Series*. Clarendon Press, Oxford, United Kingdom, 1991.

[8] A. C. Sanderson and L. E. Weiss. Image-based visual servo control using relational graph error signals. In *Proc. of the Int. Conf. on Cybernetics and Society*, pages 1074–1077, Cambridge, MA, October 1980.

[9] W. J. Wilson, C. C. W. Hulls, and G. S. Bell. Relative end-effector control using Cartesian position-based visual servoing. *IEEE Transactions on Robotics and Automation*, 12(5):684–696, October 1996.