2013 IEEE/RSJ International Conference on
Intelligent Robots and Systems (IROS)
November 3-7, 2013. Tokyo, Japan

Partial visibility constraint in 3D visual servoing

Olivier Kermorgant

Abstract— **In this paper we address the problem of visibility in position-based visual servoing. It is well known that the observed object may leave the field of view in such schemes, as there is usually no control in the image. Recent control schemes try to cope with this issue by defining an image-based constraint such that the object stays in the image. We propose to increase the convergence domain of such schemes by defining a new constraint that allows the observed object to leave partially the field of view. The general formulation is exposed and the computation of this constraint is detailed. Experiments show that controlling the visibility loss allows performing position-based visual servoing tasks that were impossible to perform while keeping the whole object in the image.**

Index Terms— **Visual servoing, visibility constraint, visual tracking**

I. Introduction

The definition of visual servoing is to control a robot by using features extracted from the image of a camera. If the corresponding interaction matrix is correctly estimated [1] then the chosen visual features will usually converge to their desired values with an exponential decrease of the error. The main concern is about the actual trajectory of the robot. Indeed, depending on the visual features that are used, the corresponding 3D trajectory may be very satisfactory and predictable (such as a straight 3D line from the initial to the desired position) or quite unpredictable if the visual features are not suited for the task that the robot has to perform.

The two classical classes of visual features have opposite approaches to this issue. Image-based visual servoing (IBVS) consists in using only 2D geometric features from the image. The goal is thus to find 2D features that have good properties in terms of induced 3D trajectory. For instance it is known that using the Cartesian coordinates of 2D points is not suited when large rotational motions around the optical axis of the camera are involved. On the opposite, position-based visual servoing (PBVS) schemes use the image to retrieve 3D information on the robot position. Is it then possible to get nice 3D trajectories, but the observed object may leave the field of view during the task as the control is not performed in the image anymore.

In this paper we focus on PBVS and propose a framework that improves current methods to keep the object in sight while trying to follow a straight 3D line. The main existing approaches are detailed and classified in Section II. We then propose a new formulation that allows the observed object to leave partially the field of view. The corresponding criterion is called the partial visibility constraint. It consists in controlling the object visible area in the image, which is less restrictive than keeping the whole object visible. This criterion is detailed in Section III with the corresponding control law. Finally, experiments show that controlling the visibility loss allows performing position-based tasks that were impossible to perform while keeping the entire object in the image.

Olivier Kermorgant is with the ICube laboratory, Université de Strasbourg, Strasbourg, France.
E-mail: kermorgant@unistra.fr

II. 3D visual servoing and visibility

In this section we recall the behavior of classical PBVS schemes and detail the main approaches to the visibility issue.

A. Position-based visual servoing

Visual servoing schemes consist in defining a robot task by an error function to be minimized:

$$\mathbf{e} = \mathbf{s} - \mathbf{s}^* \tag{1}$$

where $\mathbf{s}$ is a vector of m visual features with $\mathbf{s}^*$ being their desired values. In the following, we assume $m \geq 6$. Once the visual features have been chosen, the time variation of $\mathbf{s}$ is directly related to the camera velocity screw $\mathbf{v}$ by:

$$\dot{\mathbf{s}} = \dot{\mathbf{e}} = \mathbf{L}_s \mathbf{v} \tag{2}$$

where $\mathbf{L}_s$ is the interaction matrix related to $\mathbf{s}$ and can usually be computed analytically [1]. Assuming the robot can be controlled with the camera velocity, (2) leads to the following control law:

$$\mathbf{v} = -\lambda \widehat{\mathbf{L}_s^+} \mathbf{e} \tag{3}$$

where $\widehat{\mathbf{L}_s^+}$ is an estimation of the Moore-Penrose pseudo-inverse of $\mathbf{L}_s$, that ensures at best that the error $\mathbf{e}$ is exponentially minimized in terms of euclidean norm.

Position-based visual servoing consists in using directly the 3D pose (position and orientation) of the camera as visual features [14]. This scheme is known to be globally asymptotically stable if the pose is sufficiently well estimated.

The main advantage is that the decrease of the error (1) induces a 3D straight line trajectory of the end-effector, which is of course a predictable and very satisfactory behavior. Computer vision methods are used to retrieve the 3D pose of the camera. If a CAD model of the object is known, tracking the edges of the object [3], [5] is a popular pose estimation approach. On the other hand, a model-free method has been presented in [10], allowing for the homography estimation from a set of corresponding points. The pose matrix that transforms points from object frame to camera frame is an element of the group of rigid body transformations $SE(3)$ and can be written:

$${}^c\mathbf{M}_o = \begin{bmatrix} {}^c\mathbf{R}_o & {}^c\mathbf{t}_o \\ 0 & 1 \end{bmatrix} \tag{4}$$

where ${}^c\mathbf{R}_o \in SO(3)$ is a rotation matrix and ${}^c\mathbf{t}_o \in \mathbb{R}^3$ is a translation vector.

978-1-4673-6357-0/13/$31.00 ©2013 IEEE

In order to draw a 3D straight line, the 3D features have to be chosen as :

$$\mathbf{s} = ({}^{c*}\mathbf{t}_c, {}^{c*}\theta\mathbf{u}_c) \tag{5}$$

where ${}^{c*}\mathbf{t}_c$ expresses the 3D translation and ${}^{c*}\theta\mathbf{u}_c$ expresses the angle and the axis of the 3D rotation that the robot has to achieve. The corresponding interaction matrix is bloc-diagonal, inducing decoupled translational and rotational motions. The main drawback of such a PBVS is that there is no control at all in the image (actually, the observed object frame does not appear at all in the features (5)). The observed object can thus leave the field of view (FoV), which of course would result in the task not being performed. We now summarize the main approaches to this issue.

B. Classical approaches

Several different approaches have been proposed to improve PBVS with regards to the visibility issue. They consist in choosing slightly different 3D features, relying on partitioned or switching systems, or directly taking the visibility constraint into account.

1) Better 3D features: Instead of using the 3D features (5), another popular choice is $\mathbf{s} = ({}^{c}\mathbf{t}_o, {}^{c*}\theta\mathbf{u}_c)$ where ${}^{c}\mathbf{t}_o$ expresses the translation between the camera and the observed object frame. The corresponding interaction matrix is bloc-diagonal, and the resulting control law is globally asymptotically stable and ensures that the reference point of the object frame draws a straight line trajectory in the image. Similarly, in 2 1/2 visual servoing [10], some of the 3D features are replaced by 2D information, leading to a set of 6 features that allow analytical proof of convergence and study the sensibility to calibration errors. Popular choices are to use 2D coordinates of an image point together with 3D translation (respectively rotation) along the z-axis and the whole vector for 3D rotation (respectively translation). In this case, the chosen image point (typically the centroid of the observed object) draws a straight line in the image. Yet, the induced trajectory of the end-effector is not a straight line anymore, the object may partially leave the FoV and no control is done on the reliability of the tracking with regards to the sole visible part of the object. Finally, in [12] another 6-feature-set is designed to cope the visibility issue: 2D coordinates and 3D rotation together with the radius of the circumcircle of the object projection. Yet, this shape may not be suited for all 3D objects, and a planning scheme must be used in the general case.

2) Partitioned and switching systems: In [2], PBVS translational and rotational motion are switched when the image points are near to the border. The goal is to perform only the 3D motion that will keep the points in the image. Isolating the z-axis is also considered in [4] with a partitioned strategy. In practice, this leads to backward translational motions which is not an optimal 3D trajectory. In [6], a switching between IBVS and PBVS is proposed. A maximum error is defined for each scheme, that makes the system switch to IBVS (resp. PBVS) if the 2D (resp. 3D) error norm is too high. However the maximum acceptable 2D error may be difficult to define: if too high, a point may be able to reach the image border. If too small the scheme may stay in IBVS mode. Similarly, a hybrid control law has been proposed in [7] with a 5D-objective function that allows determining the best weighting between IBVS and PBVS. Once again, the tuning may be difficult in practice and does not ensure the visibility because the 2D weights are bounded.

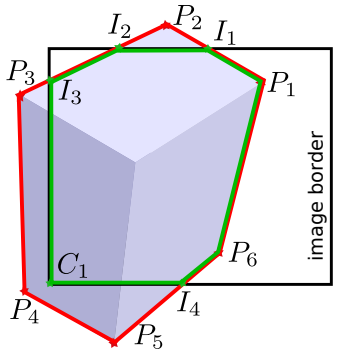


Fig. 1. Whole area (large red polygon) and visible area (small green polygon) of the 3D object projection in the image plane. While all points of the red polygon belong to the object (P_i), the green polygon may involve intersection points (I_i) and corner points (C_i).

3) Constrained visual servoing schemes: The visibility can also be seen as a constraint to be ensured while performing the PBVS. As PBVS schemes are rank 6, classical avoidance approaches such as Gradient Projection Method (GPM) [15] are usually not suited in this case. In [9] we have proposed a hybrid approach that tries to perform a PBVS but that adds 2D information when some parts of the object are about to leave the FoV. The induced end-effector trajectory is as near as possible to a straight line, with only the minimal changes allowing for the visibility. The framework of cascaded quatradic programming (QP) [8] could also be used to model the visibility as a constraint. These approaches ensure the object is always entirely visible in the image. Paradoxically this may also be seen as a too important constraint that may prevent the system from reaching its desired position. Indeed, the tracking can usually be performed even if some parts of the object are out of the FoV, as long as enough of the object is visible.

III. Partial visibility

In our approach we rely on the constrained control formalism. However, compared to previous works [9] we do not define the constraint as keeping the whole object in the FoV. Here, the constraint is just to keep a sufficient part of the object in the FoV. The corresponding visual feature, that we call the visibility ratio, is defined in this section together with its interaction matrix. We assume a 3D object is being observed. We denote a the area of the projection of the object in the whole image plane, and a_v the area of the projection of the object in the actual image (see Fig. 1). The visibility ratio is defined as :

$$r = \frac{a_v}{a} \tag{6}$$

r is of course equal to 1 if the object is entirely visible, and equal to 0 if the object is entirely out of the FoV. The partial

visibility constraint consists in imposing a lower bound $r_{\min}$ for r, which will be considered as a constraint during the control law. In order to ensure at best an exponential decrease of the PBVS error, such a control law can be written [8]:

$$\begin{aligned} \mathbf{v} = \quad & \underset{\text{s.t. } r \geq r_{\min}}{\operatorname{argmin}} \|\mathbf{L}_s \mathbf{v} + \lambda \mathbf{e}\| \end{aligned} \tag{7}$$

where $\mathbf{v}$ is the camera velocity screw and λ is the control gain. $\mathbf{L}_s$ and $\mathbf{e}$ refer to the position-based visual task.
$r_{\min}$ is the minimum acceptable value for the visibility ratio. It may be a given percentage, or depend on the current visual tracking confidence. An efficient way to take such constraint into account is to transpose it to a constraint on $\dot{r}$:

$$\dot{r} \geq \alpha(r - r_{\min}) \tag{8}$$

where $\alpha > 0$ tunes the constraint. It is easily shown that if (8) is verified then $r \geq r_{\min}$ at all time. The control law thus involves the time variation of r, which is linked to $\mathbf{v}$ through its interaction matrix $\mathbf{L}_r$. From (6), $\mathbf{L}_r$ yields:

$$\mathbf{L}_r = \frac{1}{a}\mathbf{L}_v - \frac{a_v}{a^2}\mathbf{L}_a \tag{9}$$

where $\mathbf{L}_v$ and $\mathbf{L}_a$ are the interaction matrices of the visible area a_v and of the whole area a. To be performed, the control law (7) thus requires the knowledge of a and a_v together with their interaction matrices. We now expose the proposed formulation, before going into computational details.

A. General formulation

In the sequel we assume the knowledge of the 3D points corresponding to the object 3D envelop. This is typically the case when an object is tracked with a CAD model, which is a common situation in PBVS. If the camera pose ${}^c\mathbf{M}_o$ is known, then the points defining the 3D envelop can be expressed in the image as 2D virtual points. Indeed, denoting ${}^o\mathbf{X}$ the coordinates of a 3D point in the object frame, the corresponding coordinates ${}^c\mathbf{X} = (X, Y, Z)$ in the camera frame yield:

$${}^c\mathbf{X} = {}^c\mathbf{M}_o {}^o\mathbf{X} \tag{10}$$

and 2D coordinates can then be expressed by:

$$\begin{cases} x = X/Z \\ y = Y/Z \end{cases} \tag{11}$$

As shown in Fig. 1, the polygon corresponding to the whole area a is called the large polygon. Similarly, the polygon corresponding to the visible area a_v is called the small polygon.

We use Green's theorem that allows computing the area of any polygon defined with the points (x_i, y_i), $i \in [1, n]$, sequenced counterclockwise [13]. In this case, the area can be expressed as:

$$a = \frac{1}{2}\sum_{i=1}^{n} x_{i-1}y_i - x_i y_{i-1} \tag{12}$$

with $(x_0, y_0) = (x_n, y_n)$. For readability reasons, and as we are only interested in the area ratio (6), the factor $\frac{1}{2}$ is ignored in the sequel. From (12), the time variation of a yields:

$$\dot{a} = \sum_{i=0}^{n} \dot{x}_{i-1}y_i + x_{i-1}\dot{y}_i - \dot{x}_i y_{i-1} - x_i \dot{y}_{i-1} \tag{13}$$

$$= \sum_{i=0}^{n} \begin{bmatrix} y_i & -x_i \end{bmatrix} \begin{bmatrix} \dot{x}_{i-1} \\ \dot{y}_{i-1} \end{bmatrix} - \begin{bmatrix} y_{i-1} & -x_{i-1} \end{bmatrix} \begin{bmatrix} \dot{x}_i \\ \dot{y}_i \end{bmatrix} \tag{14}$$

The corresponding interaction matrix $\mathbf{L}_a$ can thus be expressed as:

$$\mathbf{L}_a = \sum_{i=1}^{n} \mathbf{R}_i \mathbf{L}_{i-1} - \mathbf{R}_{i-1}\mathbf{L}_i \tag{15}$$

where $\mathbf{R}_i = \begin{bmatrix} y_i & -x_i \end{bmatrix}$ and $\mathbf{L}_i$ is the well-known interaction matrix of the 2D image point (x_i, y_i) [1].

Hence, the partial visibility ratio (6) and the interaction matrix (9) can be computed as soon as the vertices defining the large and the small polygon have been determined, together with their interaction matrices.

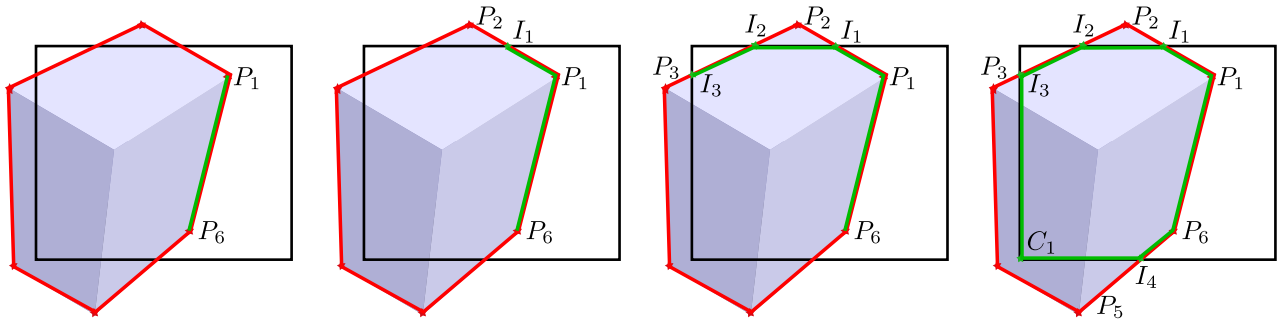
At each iteration, the first step is thus to determine the sequence of points that corresponds to the large polygon. A simple algorithm to do so is to initialize the sequence with the point that is the most far from the centroid, as this point necessarily belongs to the contour. The following points are then added counterclockwise until the initial point is found again. From (12) the computation of the whole area a is then straightforward. Similarly, from (15) the interaction matrix $\mathbf{L}_a$ can be computed. The visible area a_v and its interaction matrix $\mathbf{L}_v$ are more complex to compute. Indeed, as shown in Fig. 1, the small polygon is defined by a sequence of points that may involve intersection points between the image borders and the object 2D envelop, and corner points that are simply the corners of the image. We now detail the corresponding algorithm and computation.

B. Determining the visible area

In this section we first expose how to retrieve the sequence of points corresponding to the small polygon. The computation of the visible area a_v and its interaction matrix $\mathbf{L}_v$ are then presented.

a) Finding the small polygon: From the large polygon $(P_1, \ldots, P_n)$, Algorithm 1 exposes the retrieving of the small polygon sequence. We assume that the point $P_0 = P_n$ is visible, which can always be ensured by choosing another starting point for the sequence. If no points are visible, other 3D points can be defined along the envelop.

The algorithm behavior is illustrated on Fig. 2. Starting from $P_0 = P_6$, the next point P_1 is visible, hence it is added to the polygon (Fig. 2a). P_2 is not visible, the intersection I_1 is thus computed and added to the polygon (Fig. 2b). P_3 is not visible either, but $[P_2P_3]$ has two intersections I_2 and I_3 with the image borders. They are added to the polygon (Fig. 2c). P_4 and P_5 are not visible, thus nothing happens for them. Since P_6 is visible, the intersection I_4 is computed. As it does not belong to the same side as I_3, the intermediary corner C_1 is first added to the polygon, then I_4 and finally P_6 (Fig. 2d). In the next section we expose the computation of the interactions matrices of the intersection points.



(a) Initial segment P_6P_1 (b) Segment P_1P_2 adds I_1 (c) Segment P_2P_3 adds I_2 and I_3 (d) Segment P_5P_6 adds C_1, I_4 and P_6

Fig. 2. Sequence followed to find the small polygon. Several configurations may exists depending on the segments crossing the image borders.

Data: Large polygon $(P_0 = P_n, P_1, \ldots, P_n)$
Result: Small polygon $\mathcal{P}_s$
$\mathcal{P}_s \leftarrow P_0$;
inside $\leftarrow$ true;
for $i \in [1, n]$ **do**
  if *inside* **then**
    if P_i *is visible* **then**
      add P_i to $\mathcal{P}_s$;
    else
      find intersection I_{out} of $[P_{i-1}P_i]$ with the image sides;
      store the side S_{out} that has been crossed;
      add I_{out} to $\mathcal{P}_s$;
      inside $\leftarrow$ false;
    end
  else
    if P_i *is visible* **then**
      find intersection I_{in} of $[P_{i-1}P_i]$ with the image sides;
      store the side S_{in} that has been crossed;
      if $S_{out} \neq S_{in}$ **then**
        add corners C_i between S_{out} and S_{in};
      add I_{in} to $\mathcal{P}_s$;
      add P_i to $\mathcal{P}_s$;
      inside $\leftarrow$ true;
    else
      check for intersections of $[P_{i-1}P_i]$ with the image sides;
      if *Found two intersections* I_{in} *and* I_{out} **then**
        add potential corners as above;
        add I_{in} to $\mathcal{P}_s$;
        add I_{out} to $\mathcal{P}_s$;
        store the last crossed side as S_{out}
      end
    end
  end
end

Algorithm 1: Get the sequence of points defining the small polygon.

b) Intersection interaction matrix: The corner points being motionless, their interaction matrix is null. To retrieve the interaction matrix of the intersection points, we actually need to compute the corresponding 3D intersection. Indeed, the sole 2D intersection would prevent from knowing the Z-depth that appears in the interaction matrix. Denoting (x^-, x^+, y^-, y^+) the image limits, the borders correspond to the four 3D planes of equation:

$$X = x^- Z, \quad X = x^+ Z, \quad Y = y^- Z, \quad Y = y^+ Z \tag{16}$$

We denote $aX + bY + cZ = \mathbf{U}^\top \mathbf{X} = 0$ the equation of such a plane. Denoting $\mathbf{X}_1$ and $\mathbf{X}_2$ two 3D points, the 3D intersection $\mathbf{I}$ of $(\mathbf{X}_0\mathbf{X}_1)$ and a 3D plane yields:

$$\mathbf{I} = \mathbf{X}_0 + k(\mathbf{X}_1 - \mathbf{X}_0) \tag{17}$$

with:

$$k = \frac{-\mathbf{U}^\top \mathbf{X}_0}{\mathbf{U}^\top (\mathbf{X}_1 - \mathbf{X}_0)} = \frac{N}{D} \tag{18}$$

The intersection is considered valid only if $k \in [0, 1]$, that is $\mathbf{I} \in [\mathbf{X}_0\mathbf{X}_1]$. From (17) and (18), the interaction matrix $\mathbf{L}_{3D}$ of the 3D point $\mathbf{I}$ yields:

$$\begin{aligned}\mathbf{L}_{3D} =& \mathbf{L}_0 - (\mathbf{X}_1 - \mathbf{X}_0)\left(\frac{D\mathbf{U}^\top \mathbf{L}_0 + N\mathbf{U}^\top(\mathbf{L}_1 - \mathbf{L}_0)}{D^2}\right) \\ &+ k(\mathbf{L}_1 - \mathbf{L}_0)\end{aligned} \tag{19}$$

with $\mathbf{L}_0$ and $\mathbf{L}_1$ being the interactions matrices of the 3D points $\mathbf{X}_0$ and $\mathbf{X}_1$. $\mathbf{L}_{3D}$ is thus known and verifies:

$$\dot{\mathbf{I}} = (\dot{X}, \dot{Y}, \dot{Z}) = \mathbf{L}_{3D}\mathbf{v} \tag{20}$$

The image coordinates (x, y) of the 2D intersection can be expressed from (11) and (17). Their derivative are related to the 3D coordinates derivative:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = \begin{bmatrix} 1/Z & 0 & -x/Z \\ 0 & 1/Z & -y/Z \end{bmatrix} \begin{bmatrix} \dot{X} \\ \dot{Y} \\ \dot{Z} \end{bmatrix} \tag{21}$$

Finally, from (20) and (21), the interaction matrix $\mathbf{L}_I$ of the 2D intersection (x, y) can be expressed with:

$$\mathbf{L}_I = \begin{bmatrix} 1/Z & 0 & -x/Z \\ 0 & 1/Z & -y/Z \end{bmatrix} \mathbf{L}_{3D} \tag{22}$$

The coordinates and the interaction matrices of the points defining both the large and the small polygons have been exposed, making it possible to compute the visibility ratio (6), and its interaction matrix from (9) and (15). We now

detail the control law that ensures the partial visibility.

C. Control law

The control law (7) is expressed through the quadratic programming formalism with active sets [8]. First the unconstrained law is computed:

$$\mathbf{v}_1 = \operatorname{argmin} \|\mathbf{L}_s\mathbf{v} + \lambda\mathbf{e}\| = -\lambda\mathbf{L}_s^+\mathbf{e} \tag{23}$$

The partial visibility constraint is then checked:

$$\dot{r} = \mathbf{L}_r\mathbf{v}_1 \stackrel{?}{\geq} \alpha(r - r_{\min}) \tag{24}$$

If (24) is verified then $\mathbf{v}_1$ is applied on the system. If not, the constraint is changed to equality and the following optimization problem is considered:

$$\begin{aligned} \mathbf{v} = \quad & \operatorname{argmin} \|\mathbf{L}_s\mathbf{v} + \lambda\mathbf{e}\| \\ & \text{s.t. } \mathbf{L}_r\mathbf{v} = \alpha(r - r_{\min}) \end{aligned} \tag{25}$$

which can be solved through Lagrangian method. The joint limit constraint is considered with the same formalism. Experiments now illustrate the proposed approach.

IV. Experiments

In this section the proposed approach is exposed and compared to a visual servoing under full visibility constraint. Experiments are carried on a 6 DOFs gantry robot (see the video accompanying this paper). The camera and the robot are calibrated. The camera observes a mechanical object, the 3D model of which is known. The edges are tracked to allow for the pose estimation at camera rate [3]. The control law and tracking are performed using ViSP software [11].

A. Full visibility constraint

The robot joint limits have been defined such that the visual servoing cannot be performed with the full visibility constraint. In this configuration, joint q_1 approximately corresponds to a backward motion of the camera. This is represented in Fig. 3, where the final image (Fig. 3b) shows the object is very near to the border. The joint q_1 that would allow performing the servoing has already reached its limit as shown in Fig. 3a (normalized joint positions are represented). Consequently, the best solution is the null motion: the system stops as it is not possible anymore to minimize the position error without violating the constraints. The final position is thus the local minimum corresponding to the steepest gradient descent under the visibility and joint avoidance constraints. We now try to perform the same task with a partial visibility constraint.

B. Partial visibility constraint

The same experiment is carried, but this time the visibility constraint is to have at least $r_{\min} = 90\%$ of the object visible. The robot quickly reaches a configuration which is similar to the one previously shown in Fig. 3b but this time the object is allowed to partially leave the FoV, as shown in Fig. 4a.

Fig. 4 represents camera images at iterations 200 (94% visibility), 700 (90%) and 1400 (98%). The tracking is the same as before but only the small polygon is represented here, as in Fig. 2. We can point out that the tracking still performs well in this configuration, although only 90% of the object is visible (see the video accompanying this paper). The visibility ratio occasionally drops below 90% because of the tracking noise but still stabilizes around the acceptable value.

At iteration 1400 (Fig. 4c) the object is about to be fully visible again and the scheme is near from convergence.

Fig. 5 shows the behavior of the proposed scheme. As in the full visibility case, joint 1 quickly reaches its limit as shown in Fig. 5a. This time the robot does not have to stop, as a solution can be found by making the object partially leave the FoV. Fig. 5b represents the partial visibility ratio that decreases very quickly below 96%. The lower bound of the visibility ratio is reached around iteration 700. Joint 1 is still laying on its limit, but this does not prevent the scheme from being performed.

Once the low-visibility phase has been passed, the ratio starts to increase again and is equal to 1 (full visibility) a few iterations before convergence. We can see that the object is entirely visible again around iteration 1500, while Fig. 5c shows the position error is becoming null at the same time. Most of the scheme thus performs while controlling the visibility loss.

V. Conclusions

A new approach has been proposed to cope with the visibility constraint in position-based visual servoing. Compared to previous approaches, we do allow the observed object to leave the FoV but the visibility loss is actively controlled. The approach shows that a CAD model of the observed object makes it possible to compute the interaction matrix of the partial visibility ratio and to ensure this value keeps above a given percentage. This leads to an improved 3D trajectory compared to schemes that keep the entire object in the FoV. As shown in the experiments, controlling the visibility loss can also make a task possible, while there was no solution with the full visibility constraint. As the minimum visibility ratio $r_{\min}$ is likely to depend on the model characteristics and may be difficult to define, a possible improvement is to take into account the visual tracking confidence in order to adapt the acceptable visibility ratio. This would typically be the case if there are not enough visible segments to track the observed object.

VI. Acknowledgment

The author would like to acknowledge the Lagadic team at Inria Rennes for having made it possible to perform the experiments on the Afma6 robot.

References

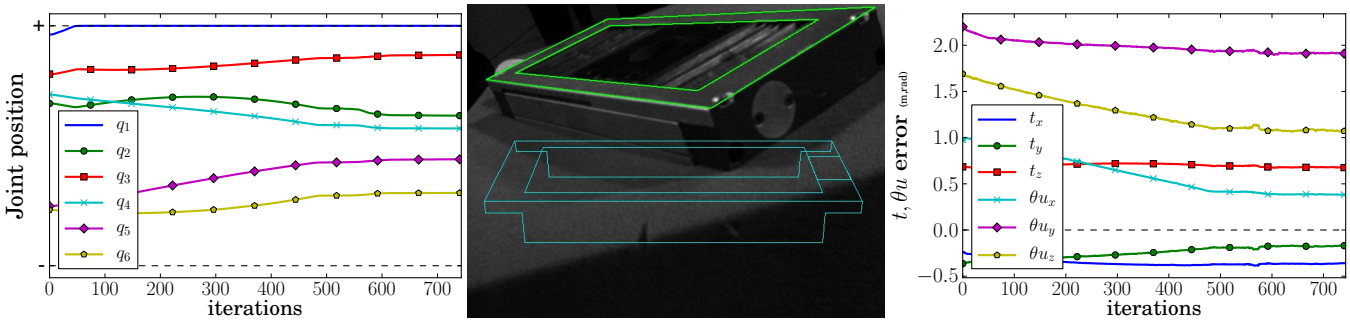
[1] F. Chaumette and S. Hutchinson, "Visual servo control. I. Basic approaches," *IEEE Robot. Autom. Mag.*, 2006.

[2] G. Chesi, K. Hashimoto, D. Prattichizzo, and A. Vicino, "Keeping features in the field of view in eye-in-hand visual servoing: A switching approach," *IEEE Trans. on Robotics*, 2004.

[3] A. Comport, E. Marchand, and F. Chaumette, "Efficient model-based tracking for robot vision," *Advanced Robotics*, October 2005.

[4] P. Corke and S. Hutchinson, "A new partitioned approach to image-based visual servo control," *IEEE Trans. Robot. Autom.*, 2001.

[5] T. Drummond and R. Cipolla, "Real-time visual tracking of complex structures," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 2002.

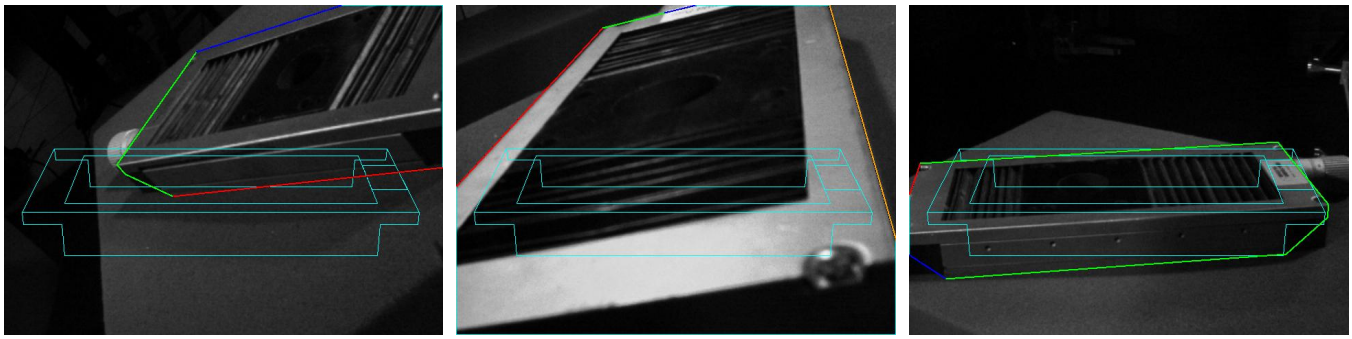


(a) Normalized joint positions

(b) Final position: the object is about to leave the image

(c) Position error

Fig. 3. Full visibility constraint. Both the joint limit (a) and the visibility constraint (b) are reached. The only solution is to stop the system and the position error stops decreasing (c).

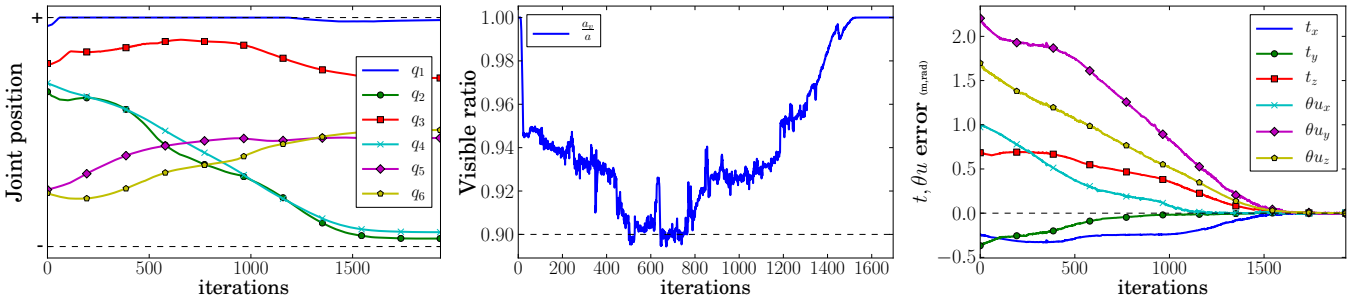


(a) Image at iteration 200 (94%)

(b) Image at iteration 700 (90%)

(c) Image at iteration 1400 (98%)

Fig. 4. Partial visibility constraint. Camera images at iteration 200 (a), 700 (b) and 1400 (c). The object is partially out of the FoV and the visibility ratio reaches its minimum value in (b).



(a) Normalized joint positions: joint 1 (blue) reaches its upper limit

(b) Visibility ratio value

(c) Position error

Fig. 5. Partial visibility constraint. Joint positions (a) reach the same limit, while the visibility ratio (b) stays above 90%. This allows the position error to converge to 0 (c).

[6] N. Gans and S. Hutchinson, "Stable visual servoing through hybrid switched-system control," *IEEE Trans. on Robotics*, 2007.

[7] A. Hafez and C. Jawahar, "Visual servoing by optimization of a 2D/3D hybrid objective function," in *IEEE Int. Conf. on Robotics and Automation*, Roma, Italy, 2007.

[8] O. Kanoun, F. Lamiraux, and P.-B. Wieber, "Kinematic Control of Redundant Manipulators: Generalizing the Task-Priority Framework to Inequality Task." *IEEE Trans. on Robotics*, 2011.

[9] O. Kermorgant and F. Chaumette, "Combining IBVS and PBVS to ensure the visibility constraint," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, San Francisco, USA, September 2011.

[10] E. Malis, F. Chaumette, and S. Boudet, "2-1/2D visual servoing," *IEEE Trans. Robot. Autom.*, Apr. 1999.

[11] E. Marchand, F. Spindler, and F. Chaumette, "ViSP for visual servoing: a generic software platform with a wide class of robot control skills," *IEEE Robot. Autom. Mag.*, December 2005.

[12] G. Morel, T. Liebezeit, J. Szewczyk, S. Boudet, and J. Pot, "Explicit incorporation of 2d constraints in vision based control of robot manipulators," *Experimental Robotics VI*, 2000.

[13] C. Steger, "On the calculation of arbitrary moments of polygons," *Munchen Univ., Munchen, Germany, Tech. Rep. FGBV-96-05*, 1996.

[14] W. Wilson, W. Hulls, and G. Bell, "Relative end-effector control using cartesian position based visual servoing," *IEEE Trans. on Robotics and Automation*, 2002.

[15] T. Yoshikawa, "Basic optimization methods of redundant manipulators," *Laboratory Robotics and Automation*, 1996.