# What Metric Stereo Can Do for Visual Servoing

Bart Lamiroy* Cyrille Puget Radu Horaud

MOVI – GRAVIR†– INRIA Rhône-Alpes

655 Avenue de l'Europe, 38330 Montbonnot, FRANCE

e-mail: Frist.Last@inrialpes.fr

**Abstract**

*This paper describes a number of geometric tools that can be easily implemented in order to obtain a very robust visual servoing platform. Most often, visual servoing algorithms are either based on calibrated systems or dispose of enough information to self-calibrate (the most noteworthy exception being [15]). We show that this abundance of information can be exploited in the case of stereo servoing as to offer a solution for some problems related to signal loss such as temporary occlusions, specularities or* CAD *tracking difficulties.*

*The presented servoing algorithm can cope with large image perturbations and is able to control a robot through a non modeled set of reference points.*

## 1 Introduction

Visual servoing is the technique of using 2D visual data, coming from one or more cameras observing, to control a robot and guide it to a goal position. Most visual servoing approaches [3, 10, 7, 12], exception made of [15], are based on the computation of a kinematic screw that is then applied to a robot end-effector. Since this kinematic screw is computed from 2D image data, but needs to be expressed in a metric reference frame related to the robot, the imaging system needs to be calibrated, one way or another. The needed information most often is obtained by using pose estimation on modeled objects with known intrinsic camera parameters.

While this information is necessary in the case of monocular servoing, it becomes redundant in the case of stereo servoing. On the other hand, there is no direct way to relax the constraints. The main idea of this paper is therefor not to look for a minimal set of data for stereo servoing, but rather to take advantage of the redundant information in order to robustify the servoing approach and to cope with perturbing image data.

This paper is organized as follows:

- First, we give a short introduction to the basic formalism adopted for describing visual servoing. We more particularly focus on the different relationships between monocular and stereoscopic cases, especially where *a priori* knowledge is concerned.
- Second, we introduce the notion of *virtual stereo servoing* which allows us to have temporary visual disturbances and still be able to continue the servo task, without loss of precision or convergence quality.
- Third, we show how we can use metric stereo vision to extend the CAD control model and finally allow full loss of all *a priori* modeled information and still continue to servo.
- Finally, we validate our approach through as series of experiments, supporting the theoretical statements made throughout the paper.

## 2 Visual Servoing

In what follows we are considering fixed cameras observing a moving robot. The whole reasoning holds for eye-in-hand approaches with some straightforward modifications to the relationships between the different reference frames we are introducing.

The main idea of task function visual servoing, as introduced by ESPIAU *et al.* in [3] concerns the application of a kinematic screw $\mathcal{T}$ to a robot end-effector

*Bart Lamiroy is now with ISA at LORIA, Nancy–France. Bart.Lamiroy@loria.fr

†GRAVIR is a joint research programme between CNRS, INPG and UJF.

which is derived from the image data and the pseudo-inverse of a Jacobian matrix relating 3D motion to 2D image motion.

## 2.1 Relating 2D Image Speed to 3D Kinematic Screw

Consider a given 3D point $P = (x\ y\ z)^{\mathsf{T}}$, undergoing a kinematic screw $\mathcal{T} = (\Omega\ V)^{\mathsf{T}}$. Its projection $(u\ v)^{\mathsf{T}}$ onto an image, with

$$u = \alpha_u \frac{x}{z} + u_0 \quad (1)$$

$$v = \alpha_v \frac{y}{z} + v_0 \quad (2)$$

is animated by the motion defined in equation (3) (see [10] for justifications).

$$\begin{pmatrix} \dot{u} \\ \dot{v} \end{pmatrix} = \mathbf{J}_P\, \mathcal{T} \quad (3)$$

with $\mathbf{J}_P$ equal to:

$$\begin{pmatrix} \alpha_u & 0 \\ 0 & \alpha_v \end{pmatrix} \begin{pmatrix} \frac{1}{z} & 0 & \frac{x}{z^2} & \frac{-xy}{z^2} & 1+\frac{x^2}{z^2} & \frac{-y}{z} \\ 0 & \frac{1}{z} & \frac{-y}{z^2} & -1-\frac{y^2}{z^2} & \frac{xy}{z^2} & \frac{x}{z} \end{pmatrix} \quad (4)$$

The observation of the image speeds of a sufficient number of rigidly linked 3D points $P_1 \ldots P_n$ undergoing the same kinematic screw $\mathcal{T}$, in conjunction with the pseudo-inverse $\mathbf{J}^\dagger$ of the resulting Jacobian matrix $\mathbf{J}^{\mathsf{T}} = \left(\mathbf{J}_1^{\mathsf{T}} \ldots \mathbf{J}_n^{\mathsf{T}}\right)$, can be used to compute the kinematic screw that was applied to the set of points. Or, in a similar matter, imposing an image speed to a sufficiently high number of image points constrains the possible 3D motion to a unique value of $\mathcal{T}$ [3, 10].

## 2.2 Required Information

In the context of visual servoing, $\mathcal{T}$ is typically the speed of the tool center point, TCP, of the robot that needs to be controlled. In that case, the observed 3D control points $P_1 \ldots P_n$ are rigidly linked to the robot, and the kinematic screw needs to be expressed in a reference frame related to this same robot.

In any case, on has to know the 3D position of the control points within the camera reference frame(s) in order to compute $\mathcal{T}$. To express $\mathcal{T}$ in the TCP frame, one needs to know the mapping between the TCP and the camera reference frames.

### 2.2.1 Monocular Case

In the case of a single image, the only solution for recovering the 3D positions of $P_1 \ldots P_n$ in the camera reference frame, denoted $\mathcal{P}_C$, is to use pose computation [9, 11], based on the known *a priori* 3D positions of the points in some local reference frame (*e.g.* CAD model), which we shall refer to as $\mathcal{P}_L$. This allows us to compute the mapping $\mathbf{T}_{C\to L}$ between the two reference frames, express $\mathcal{P}_C$ in function of $\mathcal{P}_L$, and compute $\mathcal{T}_C$.

Furthermore, to express $\mathcal{T}$ in the TCP-frame (denoted $\mathcal{T}_R$ for *robot*), one has to add the relationship between the known local reference frame $L$ and the TCP reference frame $R$, represented by the mapping $\mathbf{T}_{L\to R}$ [16]. We can assume *wolg* that $L$ and $R$ are confounded.

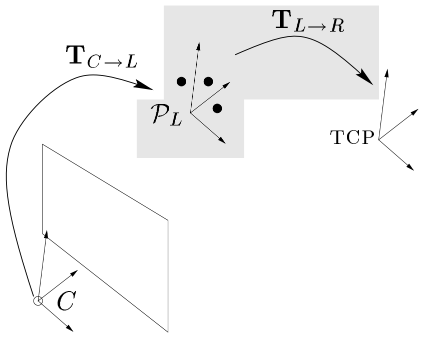


Figure 1: Relationship between different reference frames for monocular visual servoing

Figure 1 depicts the different relationships between the required *a priori* knowledge (in grey) and the computed data needed for a successful computation of $\mathcal{T}_R$.

### 2.2.2 Stereoscopic Case

Generally, when more than one camera is used for servoing [14, 8, 12], the same Jacobian-based paradigm is used for for computing the required $\mathcal{T}_R$, but combining information provided by two views. Here again, computation of the Jacobian depends on the 3D coordinates of the reference points $P_i$ in the camera reference frames ($\mathcal{P}_{C_1}$ and $\mathcal{P}_{C_2}$)

The main difference with the monocular case is that there is enough information available to directly obtain the 3D information, and that pose computation is not necessary anymore [4].

Figure 2 shows the dependency of all intervening reference frames. In this case, there is a lot of redundant information, so there is no real required *a priori* knowledge that can be highlighted.

The next two sections will show what can be done to exploit the redundancy of the servoing information to develop more robust servoing systems.

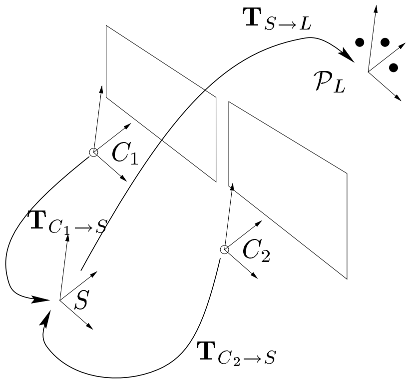


Figure 2: Relationship between different reference frames for stereoscopic visual servoing, assuming reference frames $L$ and $R$ are confounded.

## 3 Virtual Stereo Servoing

The most straightforward case is the one where all available information in the monocular case is preserved in the stereoscopic case (*i.e.* camera parameters, 3D model points $\mathcal{P}_L$, and hand-tool calibration $\mathbf{T}_{L\to R}$). In this case it may still be advantageous to use stereo servoing, since 3D trajectory and convergence quality is superior to those when servoing with only one camera [12].

### 3.1 Coping with Temporary Signal Loss

It is clear that this configuration allows for a full Euclidean self-calibration of the stereo rig. Pose computation with either of the cameras gives us the relationship between the camera reference frames $C_1$ and $C_2$ and the local control point reference frame $L$. This pose is a simple rigid transform between the two reference frames. We therefor obtain (assuming homogeneous notations):

$$\left\{ \begin{array}{lcl} \mathcal{P}_{C_1} & = & \mathbf{T}_{C_1\to L}\, \mathcal{P}_L \\ \mathcal{P}_{C_2} & = & \mathbf{T}_{C_2\to L}\, \mathcal{P}_L \end{array} \right.$$

$$\begin{array}{rcl} \Rightarrow \mathcal{P}_{C_2} & = & \mathbf{T}_{C_2\to L}\, \mathbf{T}^{-1}_{C_1\to L}\, \mathcal{P}_{C_1} \\ \mathcal{P}_{C_2} & = & \mathbf{T}_{C_2\to C_1}\, \mathcal{P}_{C_1} \end{array} \qquad (5)$$

This means that, at any moment, of the servoing control loop we dispose of enough information to predict the position of the projected points in one image, from the sole information of these points in the other one with their 3D correspondences $\mathcal{P}_L$.

Above self-calibration equations are known to be unstable, and given here only as to show that enough data is available. Robust methods for stereo self-calibration exist in this case [2].

Now, if for some reason (local occlusion, specularities, *etc.*) the signal in either of the two cameras is affected as to cause the loss of the tracked 2D control points, and provided we have a means of detecting this loss, we can temporarily use this prediction to continue to servo to our goal position, until the visual perturbance is lifted. Since the supposed position of the locally occluded points is known, it is fairly straightforward to implement a system that detects when the control points are visible again, and then reintegrates the visual information as soon as it becomes available.

There is a reason for inserting this prediction in the stereo servo loop and not brutally switching to monocular servoing. As a matter of fact, one could as well consider one of the cameras "out" and use the information of the remaining camera to continue servoing with only one. This would consist in switching between monocular and stereoscopic servoing, and would not be the *virtual stereo servoing* we propose in this paper. Since both methods strictly use the same visual information one might be tempted to suppose that they are equivalent.

They are not! The principal difference lies in the implicit constraints used in the computation of the pseudo-inverse of $\mathbf{J}$. As one knows, the solution $\hat{\mathcal{T}} = \mathbf{J}^{\dagger}\dot{s}$ to the over-constraint $\dot{s} = \mathbf{J}\mathcal{T}$ is in fact a least-square minimization of $\|\mathbf{J}X - \dot{s}\|$ [6]. From a control theory point of view, it is important to continue to minimize the same criterion during the whole process in order to guarantee its convergence. We show, in the following paragraphs that the minimization criterion fundamentally differs between the monocular and stereo case, even using the same visual information.

### 3.2 Analytical Difference Between Mono and Stereo

Equation (4) tells us that the left and right Jacobian matrices $\mathbf{J}_1$ and $\mathbf{J}_2$ non-linearly depend on the position of the 3D control points, while equation (5) gives us the rigid transformation between the two camera reference frames. Let us suppose that $\mathbf{T}_{C_2\to C_1}$ is made up of a rotation $\mathbf{R}_{C_2\to C_1}$ and a translation $\mathbf{t}_{C_2\to C_1}$.

We know that, for a 3D point $P^{\mathsf{T}} = (x\ y\ z)$ and its projection $p_1^{\mathsf{T}} = (u\ v)$ in the first camera, the following equation holds (by simple derivation, and neglecting intrinsic parameters for readability):

$$\left(\begin{array}{c} \dot{u} \\ \dot{v} \end{array}\right)\Big|_{C_1} = \left(\begin{array}{ccc} \frac{1}{z} & 0 & \frac{x}{z^2} \\ 0 & \frac{1}{z} & \frac{-y}{z^2} \end{array}\right)\Big|_{C_1} \left(\begin{array}{c} \dot{x} \\ \dot{y} \\ \dot{z} \end{array}\right)\Big|_{C_1} \quad (6)$$

The same point $P$, observed by the second camera, gives an image speed that can be expressed as follows:

$$\left(\begin{array}{c} \dot{u} \\ \dot{v} \end{array}\right)\Big|_{C_2} = \left(\begin{array}{ccc} \frac{1}{z} & 0 & \frac{x}{z^2} \\ 0 & \frac{1}{z} & \frac{-y}{z^2} \end{array}\right)\Big|_{C_2} \mathbf{R}_{C_2 \to C_1} \left(\begin{array}{c} \dot{x} \\ \dot{y} \\ \dot{z} \end{array}\right)\Big|_{C_1} \quad (7)$$

These equations can be rewritten as

$$\dot{p}_1 = \mathbf{A}_1 \dot{P} \quad (8)$$

$$\dot{p}_2 = \mathbf{A}_2 \dot{P} \quad (9)$$

$\mathbf{A}_1$ and $\mathbf{A}_2$ being linearly independent. Rewriting the minimization problem of equation (3) gives us

$$[\mathbf{A}_1 \quad \mathbf{B}_1]\,\mathcal{T} = \mathbf{A}_1 \dot{P} \quad (10)$$

for the monocular case and

$$\left[\begin{array}{cc} \mathbf{A}_1 & \mathbf{B}_1 \\ \mathbf{A}_2 & \mathbf{B}_2 \end{array}\right] \mathcal{T} = \left[\begin{array}{c} \mathbf{A}_1 \\ \mathbf{A}_2 \end{array}\right] \dot{P} \quad (11)$$

for the stereo case. Both $\mathbf{A}_1$ and $\mathbf{A}_2$, as well as $\mathbf{B}_1$ and $\mathbf{B}_2$ are linearly independent [10]. The fact that the interacting sub-matrices are independent assure that the obtained least-squares solution $\hat{\mathcal{T}}$ (or the corresponding pseudo-inversed Jacobian-matrix) differs from the monocular and stereoscopic viewpoint.

### 3.3 Geometric Interpretation

The following example shows the reason for the previously shown analytical difference. Consider 4 coplanar 3D control points $P_1 \ldots P_4$ forming a square. One classical problem in monocular servoing consists in tempting to operate a $180^{\mathrm{O}}$ rotation around its central point [1], as depicted in Figure 3. The computed image speeds result in a 3D translation to "infinity and back". Aside from being unrealistic, this solution is, from an analytical point of view, the single valid motion respecting the given monocular image speeds (in red on the figure).

If we introduce a second camera, we add no new visual information whatsoever, since the new viewpoint relates to the initial one though a simple planar homography, fully parameterized by the relative positions of both cameras [4]. However, the vanishing point at infinity in the monocular case now becomes

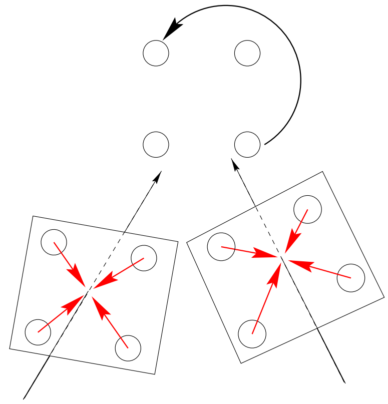

Figure 3: The computed image speed for a planar $180^{\mathrm{O}}$ rotation observed with two cameras. Both images are related by a planar homography.

a real 3D point in Euclidean space and the resulting kinematic screw is no longer a pure translation, but contains a rotational part, due to the minimization of over-constraint, and analytically unsolvable problem. The main reason for this is that, in the monocular case, there existed a 3D motion (or kinematic screw) projecting into the computed image motion that respected the implicit rigidity constraint, while the introduction of the second camera forces the 3D motion to be completely planar, thus making it impossible with a rigid body.

## 4 Using Unmodeled Control Points

Another advantage of using metric stereo is that one can relax the visibility constraints on the modeled control points $\mathcal{P}_L$, or even not use any *a priori* known 3D points at all. This section shows how.

### 4.1 The Visibility Constraint

We have shown in the previous section that temporary visibility problems of the control points in one of the images could easily be handled by virtual stereo servoing. However, some trajectories or configurations would require the modeled 3D control points to simultaneously vanish in both images of the stereo rig (most often this is due to self-occlusion).

The main idea is to track and reconstruct a number of non modeled points that are rigidly fixed to the robot end-effector (the number may vary as the end effector moves and as tracked points appear or disappear), as shown in Figure 4.

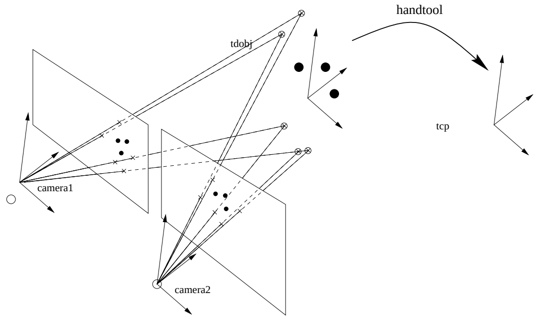


Figure 4: Reconstruction of unmodeled points

There are two major problems related to this question. First, one has to be sure that the detected and tracked points effectively belong to the same rigid body as the end-effector. Second, the reconstruction is obtained within the scene reference frame ($S$ in Figure 2) and needs to be related to the end-effector reference frame in order to compute the final kinematic screw.

### 4.2 Updating the 3D Data

Let us suppose that at the beginning of the task, a sufficient number of 3D control points are visible. A simple Euclidean alignment between the reconstruction of the control points in the scene reference frame $S$ and their representation in a local reference frame $L$ [5, 13, 16] gives us the $T_{S \to L}$ mapping shown in Figure 2.

As described in [2] or in various unpublished work by other authors, it is equally possible to determine if a set of detected other points belong to the same rigid body as the control points that are tracked. The reconstruction of these points, may be used to update the known 3D model on the fly, thus introducing new "virtual" control points. Doing so, we obtain a 3D Euclidean model in which

1. at any time in the tracking loop, we are sure to have a set of visible points (unless, of course, the whole end-effector leaves the image), of which we know the exact relationship with respect to the TCP frame,

2. at any time in the tracking loop, we can compute the "virtual" position of the modeled control points that are needed to be aligned with the required visual goal position.

The following algorithm guarantees that a complete loss of the initial control points $\mathcal{P}_L$, provided that a sufficiently large part of the end-effector remains visible in both images, has no effect on the continuity of the task execution.

**Algorithm 4.1**
Avoiding the Visibility Constraint by Updating the 3D Model

`/* Initialization */`
`Let` $\mathcal{M}_{3D} = \mathcal{P}_L$
`Let` $\{Tr\}$ `be the set of tracked 2D projections of` $\mathcal{M}_{3D}$

`/* Loop */`
`Reconstruct the tracked points` $\{Tr\}$ `in` $S$
`Estimate` $T_{S \to L}$

`Find a set` $\{\mathcal{N}\}$ `of new 2D points, rigidly liked to` $\mathcal{M}_{3D}$ `[2]`
$\{\mathcal{N}\}_S^{3D}$ `= reconstruction of` $\{\mathcal{N}\}$ `in` $S$
$\{\mathcal{N}\}_L^{3D} = T_{S \to L}^{-1} \{\mathcal{N}\}_S^{3D}$
`Update` $\mathcal{M}_{3D}$ `with` $\{\mathcal{N}\}_L^{3D}$
`Update` $\{Tr\}$ `with` $\{\mathcal{N}\}$

## 5 Experimental Results

In this section we present some experimental results supporting the different algorithms we presenting.

### 5.1 Virtual Stereo Servoing

In the following experiment, we execute the same visual task under different conditions. First, we consider the execution without signal loss or tracking problems. The results (2D and 3D trajectories, kinematic screw, *etc.*) will serve as ground truth for the rest of the experiment, where we shall introduce a number of perturbing factors.

#### 5.1.1 Experimental Setup and Self-Calibration

In our setup we are using a 6 DOF Puma-like Stäubli robot, that is controlled by a set of fixed cameras. The end-effector is controlled through a planar set of four control points, as shown in Figure 5. The experiment reported in this paper will always use the same initial position and the same goal position as shown. During the experiments, only the lighting conditions vary, causing specularities to appear, or camera resources are disrupted.

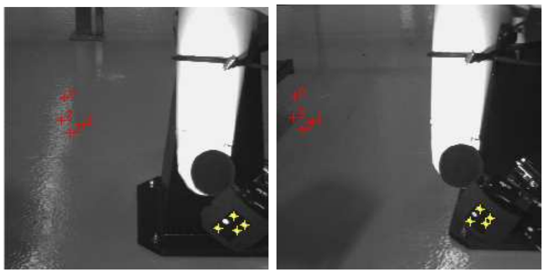

Figure 5: Stereo pair of initial robot position (in yellow) and goal position (in red)

We have also shown, in section 3 that both cameras need to be fully calibrated. We show with our experiments that a rough self-calibration suffices to execute the required task, and remain robust to the perturbations we introduce. We obtain this self-calibration by using a pose computation on the four control points at initialization, using factory given intrinsic parameters. Both left and right pose contribute then to estimate the left-right camera reference frame mapping.

#### 5.1.2 Reference Task Execution

Figures 6 and 7 show the reference trajectories in the case of non-perturbed tasks. Difference in trajectory smoothness is due to the instabilities of pose computation that are leveled by the stereo servoing [12].

#### 5.1.3 Comparison Between Monocular and *Virtual Stereo* Servoing

In this section we compare the behaviour between monocular and *virtual stereo* servoing in the case of signal loss during the task execution. We make a distinction between two major events: either the perturbation affects the task in the middle of the servo loop, either it occurs at convergence. The main difference is that, during the loop, the process still can recover, and correct possible deviations, after the visual perturbation has bin lifted. This is not the case in the case when the signal loss occurs at convergence.

**Perturbations during servoing**

Figure 8 shows the behaviour of the *virtual stereo* command. We did not present the same situation in case of monocular servoing (*i.e.* the control mode switches from full stereo to mono when signal loss is detected), since the trajectory curves are sensibly equivalent to the ones in Figure 6.

What we note is that the *virtual stereo* approach is able to absorb the instabilities proper to the monocular method, and give far smoother trajectories than

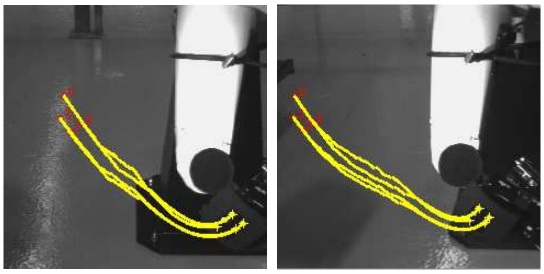

Figure 6: Stereo pair of initial robot position (in yellow) and goal position (in red) with reference trajectory using monocular servoing (note the irregularities in the curve)

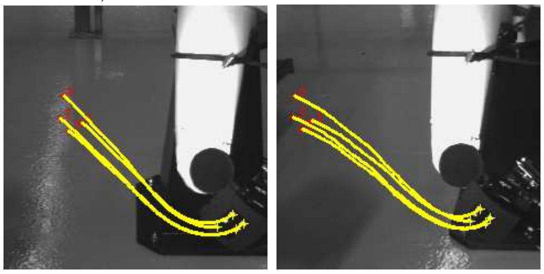

Figure 7: Stereo pair of initial robot position (in yellow) and goal position (in red) with reference trajectory using stereo servoing (note the smoothness of the curve)

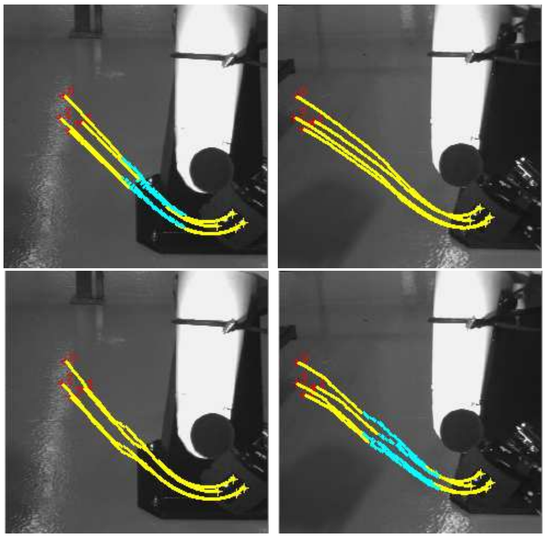

Figure 8: Perturbation of the stereo servo loop by signal loss (left camera drop-out at the top, right camera drop-out at the bottom). Blue trajectories correspond to predicted point positions in the affected image. Note the smooth behaviour of the corresponding yellow curve in the other image)

the ones observed in Figure 6.

In the example (Figure 8) where the right camera drops out, we observe a slight deviation in the left trajectory which is, however, nicely corrected, once the perturbation is lifted.

We can also note that the rough self-calibration (which affects the smoothness and continuity of the blue curve) does not affect the overall performance of the method.

**Perturbations at convergence**

However, if the signal loss occurs at convergence, the quality of the self-calibration is fundamental. Figure 9 shows an example, using the same parameters as before. Although we have a nice convergence in both

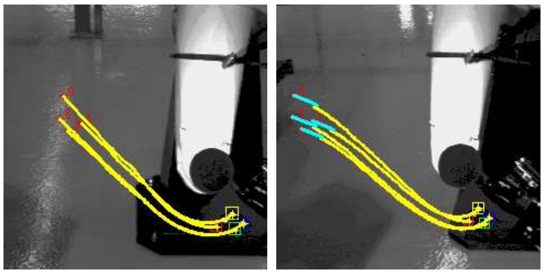

Figure 9: Perturbation at convergence.

images, we note that the self-calibration parameters induce an important loss of continuity in the right image when the signal loss occurs. It becomes even more clear when we observe the 3D Cartesian convergence. Table 1 reports the observed 3D distance between the known reference position, and the one attained with all different configurations:

Table 1: Observed 3D Convergence Errors for Different Servoing Methods.

| | |
|---|---|
| Full Stereo | 1.9 mm |
| Left Mono | 21 mm |
| Virtual Stereo (right drop-out) | 21.5 mm |
| Right Mono | 4.8 mm |
| Virtual Stereo (left drop-out) | 7.4 mm |

We clearly see that in this case, the imprecision of the reprojections induce a greater error than can be expected with single view servoing.

The most obvious solution, of course, is to use a more robust self-calibration method. The one we chose was deliberately unprecise, to show its influence on the different parts of the method. It also fundamentally relies on pose computation, rather than reconstruction, which is known to be unstable.

## 6 Conclusion

We presented in this paper a number of arguments in favour of stereoscopic visual servoing. Moreover, we showed that use of full Euclidian calibration can contribute to the robustness of servoing methods. We developed a number of algorithms that integrate the loss of one or more control points.

We more particularly advance that the use of Euclidian reconstruction, instead of pose computation enhances numerical stability, and allows for a more flexible use of the control point set, allowing it to partially disappear from the field of view, provided that a set random rigid points has been observed before.

**Acknowledgments**

This work was supported by VIGOR, Esprit-IV reactive LTR project, number 26247, and by the INRIA ARC−AVEC

## References

[1] F. Chaumette. Potential problems of stability and convergence in image-based and position-based visual servoing. In *Workshop on Vision and Control, Block Island, Rhode Island,*, June 1997.

[2] D. Demirdjian and R. Horaud. A projective framework for scene segmentation in the presence of moving objects. In *Proceedings of the Conference on Computer Vision and Pattern Recognition, Fort Collins, Colorado, USA*, 1999. to appear.

[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] O. Faugeras. *Three-Dimensional Computer Vision - A Geometric Viewpoint.* Artificial intelligence. The MIT Press, Cambridge, MA, USA, Cambridge, MA, 1993.

[5] O. Faugeras and M. Hebert. The representation, recognition, and locating of 3D objects. *The International Journal of Robotics Research*, 5:27–52, 1986.

[6] G.H. Golub and C.F. van Loan. *Matrix Computation.* The Johns Hopkins University Press, Baltimore, 1989.

[7] G. D. Hager, G. Grunwald, and G. Hirzinger. Feature-based visual servoing and its application to telerobotics. In *Proceedings of the IEEE/RSJ/GI International Conference on Intelligent Robots and Systems*, volume 1, pages 164–171, September 1994.

[8] G.D. Hager, W. Chang, and A.S. Morse. Robot hand-eye coordination based on stereo vision. IEEE *Control Systems*, page 30, 1995.

[9] R.M. Haralick, H. Joo, C. Lee, X. Zhuang, V.G. Vaidya, and M.B. Kim. Pose estimation from corresponding point data. IEEE *Transactions on Systems, Man and Cybernetics*, 6(19):1426–1446, November/December 1989.

[10] R. Horaud, F. Dornaika, and B. Espiau. Visually guided object grasping. *IEEE Transactions on Robotics and Automation*, 14(4):525–532, August 1998.

[11] R. Horaud, F. Dornaika, B. Lamiroy, and S. Christy. Object pose: The link between weak perspective, paraperspective, and full perspective. *International Journal of Computer Vision*, 22(2):173–189, March 1997.

[12] B. Lamiroy, B. Espiau, N. Andreff, and R. Horaud. Controlling robots with two cameras: How to do it properly. In *Proc. IEEE Int. Conf. on Robotics and Automation, San Fransisco, California, USA*, pages 2100–2105, April 2000.

[13] Z.-C. Lin, T. S. Huang, S. D. Blostein, H. Lee, and E.A. Margerum. Motion estimation from 3-d point sets with and without correspondences. In *Proceedings Computer Vision and Pattern Recognition*, pages 194–201, Miami-Beach, Florida, USA, June 1986.

[14] N. Maru, H. Kase, S. Yamada, and A. Nishikawa. Manipulator control by visual servoing with the stereo vision. In *International Conference on Intelligent Robots and Systems, Yokohama, Japan*, pages 1866–1870. IEEE, July 1993.

[15] A. Ruf, F. Martin, B. Lamiroy, and R. Horaud. Visual control using projective kinematics. In *Proceedings of the Ninth International Symposium on Robotics Research (ISRR 1999)*, pages 66–73, Snowbird, Utah, USA, October 1999.

[16] R.Y. Tsai and R.K. Lenz. Real time versatile robotics hand/eye calibration using 3d machine vision. In *Proceedings of* IEEE *International Conference on Robotics and Automation, Philadelphia, Pennsylvania, USA*, pages 554–561. IEEE Robotics and Automation Society, 1988.