An efficient method to compute the inverse Jacobian matrix in visual servoing

Jean-Thierry Lapresté, Frédéric Jurie, Michel Dhome, François Chaumette

HAL Id: inria-00352026

https://inria.hal.science/inria-00352026

Submitted on 12 Jan 2009

An Efficient Method to Compute the Inverse Jacobian Matrix in Visual Servoing

J.T. Lapresté[1]   F. Jurie[1]   M. Dhome[1]   F. Chaumette[2]

[1] LASMEA, UMR 6602 du CNRS,Université Blaise Pascal, 63177 Aubière Cedex, France
[2] IRISA/INRIA Rennes, Campus de Beaulieu, 35042 Rennes Cedex, France

Abstract— The paper presents a method for estimating the inverse Jacobian matrix of a function, without computing the direct Jacobian matrix. The resulting inverse Jacobian matrix is shown to perform much better in modelling a relation $\theta = f^{-1}(x)$ than the classical Moore-Penrose inverse J_f^+. Theoretical insight as well as comparisons in the domain of visual servoing are provided to demonstrate this assertion.

I. INTRODUCTION

Inverse kinematics plays a central role in robotics [14] and infography [5]. In both cases, the basic idea is the same: there exists a direct geometric relationship $x_t = f(\theta_t)$ between the degrees of freedom of the system θ_t and its output x_t for any time t, and the goal is to solve this equation to obtain θ_t from x_t. This problem can be solved in two different ways. The first is to use a global method, finding the θ's optimal path along the full trajectory. This process is clearly time consuming and cannot be realised on line, especially if f is highly non linear. The alternative, which is more suited to on line computations is to use local estimates, by differentiating the basic relation:

$$\dot{x}_t = \frac{\partial f}{\partial \theta_t}\dot{\theta}_t \qquad (1)$$

and locally inversing this relation to obtain $\dot{\theta}_t$ as a function of $\dot{x}_t$. This computation is classically performed using the Moore-Penrose pseudo-inverse $J_f^+(\theta)$ where $J_f = \frac{\partial f}{\partial \theta_t}$ is the Jacobian matrix of f at point θ.

In this paper, a new definition of the generalised inverse is proposed, which generalizes and justifies the approach proposed in [11] for template tracking in an image sequence. After theoretical justifications of the method presented in Section II, its application to visual servoing is proposed in Section III. The article concludes with a discussion of the method.

II. GENERAL CASE

A. Methods

Assume that we are in the common situation where parameters $\theta = (\theta_i)_{1 \leq i \leq n}$ explain measurements $x = (x_j)_{1 \leq j \leq m}$ by a formula or computing process $x = f(\theta)$. The generalized inverse problem is to estimate from the observation x the parameters θ, in as large a neighbourhood as possible and with as little computation as possible.

f is generally neither linear nor invertible, but always, we can seek a local solution $\Delta\theta = A\Delta x$ for Δx in some neighbourhood of zero and some matrix A to be estimated. Traditionally, one takes a Taylor expansion of x,

$$f(\theta) = J_f \Delta\theta$$

(J_f being the Jacobian matrix) and inverts using the Moore-Penrose pseudo-inverse i.e. $A = J^+$.

It is possible to consider another solution than computing the Moore-Penrose pseudo inverse $J_f^+(\theta)$. Instead, we propose to use a learning approach to directly compute the Jacobian matrix of a supposed local inverse f^{-1} of f.

The learning stage is very straightforward. One generates a random sample of N_r increments $\Delta\theta$, producing a $n \times N_r$ matrix of perturbations $\theta + \Delta\theta$ and the $m \times N_r$ matrix of the corresponding measurements variations $\Delta x = f(\theta + \Delta\theta) - f(\theta)$ which can easily be computed using f.

Resolving the linear problem $A\Delta x = \Delta\theta$ in the least square sense will produce a new matrix A that will be denoted $J_f^{\oplus}$.

This computation must be performed using a numerically stable process (SVD, for instance) as Δx may not necessarily have full rank.

B. Why is this method better?

This method is already no worse than the classical ones as the computation of J_f^+ is almost always done by a finite difference scheme rather similar to the learning stage and almost as costly.

Moreover, if f is nearly linear around θ, it is clear that the classical approach leads to satisfying results. Unfortunately this it is not always the case and in general the validity of the linear approximation of $f(\theta)$ is restricted to a very small neighborhood of θ.

If one estimates the dimension N of the vector space spanned by the image of a small neighbourhood of θ under f, by observing the approximate rank of a matrix of samples, it will find that this rank is greater than the dimension of the parameter space and that, in fact, the greater the dimension, the worse the approximation...

This dimensional gap can be seen as the proof of existence of supplementary 'hidden' parameters, knowledge of which could lead to a far better linear approximation if they could be taken into account.

For instance, suppose that f is quadratic, but that we add to the θ_i parameters, the new (non linear) parameters $\nu_{ij} = \theta_i\theta_j$. That gives us a linear expression in place of a quadratic one. Using the computation of the Jacobian matrix of this

new function of $n(n + 1)/2$ variables will make the validity domain of the approximation grow at the expanse of a lot of supplementary computations.

Our approach allows to take these 'hidden' parameters into account without having to use them explicitly, and without introducing supplementary in line computations (only the learning step will suffer).

As already said, the computation of the matrix Δx of perturbations allows the number of hidden parameters necessary to a correct linear estimate to be estimated (if we decide to do this).

The $J_f^{\oplus}$ computation implicitly uses this fact. We pretend that $J_f^{\oplus}$ is made of the n first lines of a pseudo-inverse (not necessarily the Moore-Penrose one) of the Jacobian matrix of a supposed function $\tilde{f}(\theta, \nu)$ where the ν_i are hidden.

This can be seen by saying that the random generation of the set of perturbations around θ leads to the estimation of a set of Δx that takes into account the variations of the hidden parameters, due to the non-linearity of f.

Obviously, the top of the Jacobian matrix is all we need, as the only variations that we wish to estimate are those of the original parameters (θ), and not of the unknown hidden parameters (ν).

It is also clear that, as soon as f is not perfectly affine, the information contained in $J_f^{\oplus}$ is far greater than that in J_f^+ as soon as f is not perfectly affine, since the J_f pseudo-inverse is not the same as the n first lines of J_f^+!

C. Two examples

1) A quadratic function: Consider a function $f(\theta_1, \theta_2)$ from $\mathbb{R}^2$ into $\mathbb{R}^p$, $f = (f_i)_{1 \le i \le p}$ defined by:

$$x_i = f_i(\theta_1, \theta_2) = a_{i1}\theta_1 + a_{i2}\theta_2 + a_{i3}\theta_1\theta_2 + a_{i4}\theta_1^2 + a_{i5}\theta_2^2$$

At $(0, 0)$, the value of J_f is:

$$J_f(\theta_1, \theta_2) = ((a_{ij})_{1 \le i \le p, 1 \le j \le 2})$$

Also, let

$$x_i = \tilde{f}(\theta_1, \theta_2, ..., \theta_5) = a_{i1}\theta_1 + a_{i2}\theta_2 + a_{i3}\theta_3 + a_{i4}\theta_4 + a_{i5}\theta_5$$

$\tilde{f}$ is a linear function of 5 reals that is identical to f on the manifold of $\mathbb{R}^5$ generated by $(\theta_1, \theta_2, \theta_1\theta_2, \theta_1^2, \theta_2^2)$.

$J_{\tilde{f}}$ is a constant matrix :

$$J_{\tilde{f}}(\theta_1, \theta_2, ..., \theta_5) = ((a_{ij})_{1 \le i \le p, 1 \le j \le 5})$$

If we suppose that our interest point is the origin, there are at least three ways to solve our inversion problem,

1) compute J_f and deduce J_f^+,
2) compute $J_{\tilde{f}}$ and the pertinent part of $J_{\tilde{f}}^+$
3) estimate $J_f^{\oplus}$

The following tables show the accuracy of the different approximations. For a fixed choice of $\tilde{f}$, the norms $||K\Delta x - \Delta\theta||$ are presented (where K is one of the different Jacobian matrices. In the first table the computations are made with the

matrices Δx and $\Delta\theta$ that were previously used to generate $J_f^{\oplus}$. The first column contains the number of perturbations used for $J_f^{\oplus}$.

#	$||J_f^+ \Delta x - \Delta\theta||$	$||J_f^{\oplus} \Delta x - \Delta\theta||$	$||J_{\tilde{f}}^+ \Delta x - \Delta\theta||$
1	4.1633e-17	0.0000e+00	2.0977e-14
2	1.5701e-16	1.8916e-16	1.1932e-15
3	9.4907e-03	6.4980e-16	8.2088e-16
4	6.6804e-02	8.4902e-16	2.4521e-15
5	1.7664e-01	1.2310e-15	1.2910e-15
10	9.7743e-02	5.0797e-16	2.7565e-15
50	6.0087e-01	3.6708e-15	2.3398e-14
100	6.0682e-01	4.2373e-15	3.5011e-15
500	1.8890e+00	1.3453e-14	1.8441e-14

The next table presents similar results, but once the three Jacobian computed, new random matrices $\Delta\vartheta$ and Δx are drawn.

#	$||J_f^+ \Delta x - \Delta\vartheta||$	$||J_f^{\oplus} \Delta x - \Delta\vartheta||$	$||J_{\tilde{f}}^+ \Delta x - \Delta\vartheta||$
1	3.1735e+00	3.1735e+00	1.1941e-13
2	1.0325e+00	1.0325e+00	1.2159e-14
3	7.9791e-01	7.1834e-01	9.7157e-15
4	7.1033e-01	1.5500e-01	1.3281e-14
5	8.1970e-01	2.1996e-14	4.8466e-15
10	4.4527e-01	2.7380e-15	8.2650e-15
50	8.1853e-01	5.0798e-15	2.9975e-14
100	6.4036e-01	4.2410e-15	2.9141e-15
500	7.7098e-01	5.6362e-15	7.6079e-15

It is clear that outside the points used for learning, it is necessary to have a minimum of 5 perturbations in order to get a good estimate of $J_f^{\oplus}$. Of course, for the linear $\tilde{f}$ the result is always perfect (up to roundoff precision).

It is also noticeable that in this case, the validity domain of the approximation is only a small neighbourhood of the origin for the inverse Jacobian J_f^+, but, for $J_f^{\oplus}$, the whole space, as it is a constant Jacobian which is estimated.

2) The minimum of the Rosenbrock function:

If we denote by $P = (u, v)$ a point of the plane, the Rosenbrock function is given by

$$r(P) = r(u, v) = (u - 1)^2 + 10(u^2 - v)^2;$$

it is always non negative and reach its minimum of 0 at the point $(1, 1)$.

In figure 1 one can see the level curves of a translation of this function by $(0.5, 0)$. It is often called 'the banana function' because of the form of its curvature around the origin. It is frequently given as an example of a non trivial optimization problem, as most classical numerical optimization process (as Levenberg-Marquardt or Gauss-Newton) presents a slow rate of convergence to its minimum.

Our method is not able to replace minimization in every circumstances, but in some cases its use can be interesting.

Let us suppose we know the form of the Rosenbrock function and the fact its minimum is at $(1, 1)$, but that we have to find the minimum of an unknown translation (u_0, v_0) of it. So the function to minimise is :

$$r_0(P) = r_0(u, v) = (u - u_0 - 1)^2 + 10 * ((u - u_0)^2 - v + v0)^2.$$

As we know the original function and its minimum location, it is possible to learn the shape of the function around the minimum, and hence to convert the scalar minimization to a vector value-seeking problem, by choosing:

- some random points $(P_e^i)_{1 \leq i \leq p}$ around this position,
- some random perturbations $(\Delta r_j)_{1 \leq j \leq q}$.

We now compute $J_{r_0}^{\oplus}$ using $\Delta x_{ij} = r(P_e^i + \Delta r_j) - r(P_e^i)$ and use this to find the Δr giving the desired vector of the target values.

The figure presents the trajectory to the solution. One iteration leads directly to the minimum.
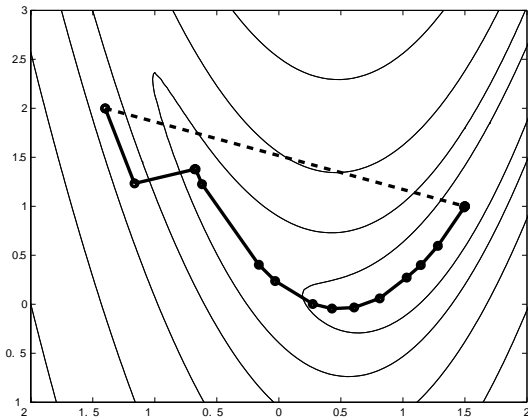

Fig. 1. One iteration and Levenberg Marquardt convergences

It is necessary to have at least $p = 5$ and $q = 5$ to ensure convergence in one iteration . In fact if we expand the banana formula:

$$r(u+1, v+1) = 41u^2 - 40uv + 10v^2 + 10u^4 + 40u^3 - 20u^2v;$$

a priori, 6 parameters are necessary to obtain a linearization, but it is easy to imagine that a rotation will diagonalise the quadratic part, removing a parameter. So a rank of 5 is necessary and observed.

Under the same conditions an algorithm such as Levenberg-Marquardt typically uses between 10 and 30 iterations and even may fail to converge if the starting point is bad enough.

On Figure 1 the banana has been translated by 0.5 in the x-direction and the minimum is at $(1.5, 1)$. The convergence of our algorithm takes only one iteration, independent of the starting point, as the linear model is perfect. This would not be the case if the banana had been rotated in the (u, v) plane. Also the method based on the inverse Jacobian (gradient descent) generally does not converge in an acceptable number of iterations.

III. APPLICATION TO VISUAL SERVOING

A. Introduction

In this section, we show how the preceeding method brings interesting results in visual servoing.

Image-based visual servoing is known to be satisfactory when the error between initial measures x and desired ones x^* are small [4], [6], [8]. As soon as the error grows, it is no longer possible to prove the system stability and potential

problems may appear in some cases [1]: convergence to a local minimum, inappropriate robot trajectory due to strong coupling, etc. To address these problems, a first approach consists of selecting visual features sharing good properties [12], [2], [9], [16], but a lot of work remains to be done in this area. An other solution is to use a planning step jointly with the servoing one. Using such an approach, the tracking error along the planned path always remains small [13], [19], that is in the stability domain of the control scheme.

The approach we propose here is simpler: it consists of performing a classical visual servoing scheme, but suppressing the strong non-linearities of the system (which are the main cause of trouble) by taking them into account in the learning stage of our Jacobian pseudo-inverse.

It must be noted that learning techniques have previously been used in visual servoing. Off-line numerical learning was even proposed in the first relevant work in the field [17], as the analytical form of the Jacobian had not then been established for most types of visual features. For the same reasons, similar methods have been presented to deal with objects whose forms and textures are unknown or complex [3]. Neural networks have also been used [18]. Finally, on-line learning methods, generally based on the Broyden estimation method, have also been proposed to deal with uncalibrated systems [7], [10], [15]. But in all of these previous studies, the authors have preferred to estimate the Jacobian and invert it, instead of the direct estimation of a pseudo-inverse.

We still denote by $x = (x_1, \ldots, x_m) \in \mathbb{R}^m$ a set of visual features measured in the image resulting from the perspective projection of a 3D scene. At this point we do not restrict the type of possible features used. They can be coordinates of the projection of 3D points, gray levels of an image region produced by the projection of a 3D object, etc.

The aim of visual servoing is to control the end effector pose θ so that the current visual features x reach a desired value x^*. As usual, the end effector pose is represented by six parameters, three to define its position, and three to define its orientation.

If the objects that the camera observes are assumed to be motionless, the relationship between the value of the visual features and the end effector pose can be written in the form $x = f(\theta)$ where f is highly non linear.

Going to kinematics, it is well known that the relationship between the velocity of the visual features and the velocity in the operational space can be written:

$$\dot{x} = J(x, z)\, \dot{\theta},$$

where $J(x, z)$ is called the *interaction matrix* related to x when $\dot{\theta}$ corresponds to the camera velocity screw [4]. This is also commonly called the *image Jacobian* when image points coordinates are being used [8]. Note that J depends on the current value x of the visual features, but also on some unknown 3D parameters, denoted here z, that typically represent the depth of the considered object.

The control scheme can then be designed from the definition of a task function e that has to be regulated to 0. More

precisely, we have [4] $e = C\,(x - x^*)$. Matrix C is classically chosen equal to $C = J^+(x, \hat{z})$ or $C = J^+(x^*, \hat{z}^*)$. In the first case, C is computed at each iteration of the control law, using the current value of x and an estimation $\hat{z}$ of 3D parameters z. In the second case, C is constant and computed off line, using $x = x^*$ and $z = \hat{z}^*$.

A very simple control law can be designed by trying to ensure a decoupled exponential decay of the task function ($\dot{e} = -\lambda e$ with $\lambda > 0$), giving:

$$\Delta\theta = -\lambda\,C(x - x^*) \qquad (2)$$

Our approach is situated at the computation level of the pseudo-inverse C of the interaction matrix and in the case where the dimension m of the visual features is greater than the space of representation of the end effector pose ($m > 6$). Although this is not discussed in the present paper, the method can also of course be applied when less than 6 degrees of freedom have to be controlled.

B. Visual servoing using image points

We now present simulation results to demonstrate the impact of the inverse Jacobian computation method in visual servoing. The idea is to replace the computation of the C pseudo-inverse by the matrix $J^\oplus$ described previously in Section II.

For illustration, we choose, the specific case of an object composed of the four vertices of a square. But we recall that the method can be applied to any type of visual primitive (straight lines, circles, drawing, etc.).

In the case of a single point, the interaction matrix is given by:

$$J = \begin{pmatrix} -1/Z & 0 & x/Z & xy & -(1 + x^2) & y \\ 0 & -1/Z & y/Z & 1 + y^2 & -xy & -x \end{pmatrix} \quad (3)$$

where (x, y) are the coordinates of a point in the image whose depth is Z. Without loss of generality, the focal length is set to 1 in this formula.

Using four points, a vector x is thus defined as a set of eight coordinates: $x = (x_1, y_1, \ldots, x_4, y_4)$ and the dimension of C is 6×8. In each experiment, the square is 1 meter large and we consider an on board camera whose desired position is such that the square appears as a centered square in the image, located at 3 m from the camera optical center. The focal length has been set to 500 pixels and the principal point of the camera is at $(0, 0)$. Finally, for all experiments, λ (defined in equation 2) has been set to 0.2.

Three simulations are given, corresponding to three different initial camera positions. As detailed in [1], the first two are usually considered to be hard in image-based visual servoing based on image points coordinates, due to the strong coupling in the interaction matrix between translation along and rotation around the optical axis.

1) In the first simulation, the camera is rotated by 50 degrees around the optical axis (see Fig 3).
2) In the second simulation, the camera is rotated by 160 degrees around the optical axis (see Fig 4).

3) In the third one, a 50 degree rotation around the x camera axis is combined to a 50 degree rotation around the z camera axis and a translational motion so that the object appears in the image for this initial position. (see Fig 5).

For each simulation, we compare three different methods using respectively:

1) the Jacobian pseudo-inverse, computed at each iteration. We denote this matrix by J_i^+,
2) the Jacobian pseudo-inverse, computed at equilibrium. We denote this matrix by J^+,
3) the new $J^\oplus$ matrix.

In cases 1 and 2, the interaction matrix is given by equation (3). In the third case, $J^\oplus$ is numerically computed as indicated in Section II-A. Space perturbations $\Delta\theta$ are generated around the equilibrium position and the value of the visual features are measured. In the experiments, 1000 perturbations were used, but this figure is not critical for the results quality. Matrix $J^\oplus$ is then computed from the formula $J^\oplus = \Delta\theta\Delta x^+$. The maximal sizes of the perturbations during the learning stage are of 50 degrees in rotation and 1 meter in translation. The matrix is computed once and for all. It is not computed on line in contrast to the J_i^+.

To make the comparisons as meaningful as possible, we also tested the case of a pseudo-inverse J_n, numerically computed from the data used for $J^\oplus$ and using the relation $J_n = (\Delta x \Delta\theta^+)^+$. The results were systematically worse than those from J_i^+, so we do not present them here.

Figures 3 to 5 present the trajectories of the four image points during the completion of the task. Due to lack of space, it was not possible to present more detailed information (such as the camera trajectory, the output of the controller, etc.)

In the first experiment (see Figure 3), the trajectory produced by using $J^\oplus$ is clearly the best one since it only involves a camera rotational motion round the optical axis, while the other matrices produce additional spurious translations.

The initial conditions of the two other simulations are even more severe, and neither J^+ nor J_i^+ are able to provide satisfactory results, as can be seen in Figure 4 and 5. On the other hand, the results obtained using $J^\oplus$ are satisfactory, both for the camera trajectory and the image points trajectories. An analysis of the rank of Δx shows that this matrix is of rank 8 (and not 6). So we are in the situation described in section II-A and the improved results are as expected.

Finally, note that even better results can be obtained using more than four points, because, with our approach this allows an even better account of the non-linearities in f. Such results can not be presented here, due to lack of space.

IV. DISCUSSION

a) Comparison of complexities: Of the three methods, only the second one, involving J_i^+, uses an on line computation. It is thus the more expensive, but recall that for the presented experiments, J_i is 8×6 and J_i^+ requires the inversion of the 6×6 matrix $J_i^T J_i$.

As for the off line computations, it is clear that our method is more expensive. $J^{\oplus}$ is computed from a system $J^{\oplus}\Delta x = \Delta\theta$, where (if N_p is the number of perturbations):

- There are $6 \times 8 = 48$ unknowns in $J^{\oplus}$;
- Δx is $8 \times N_p$;
- $\Delta\theta$ is $6 \times N_p$;

This system is equivalent to the least square resolution of a system of N_p equations with 8 unknowns for 6 right hand sides. This is practically and efficiently done with an economy size SVD (in fact the pseudo inverse of an 8×8 matrix).

The computation of J^+ is simply the pseudo-inverse of a 6×8 matrix.

b) Jacobian singularity: One of the drawbacks of classical methods is the existence of singularities that lead to failure of the control law. The new proposal does not share the same singularities. For instance, for a 180 degrees rotation around the optical axis, it is well known that the classical control laws using point coordinates can not converge [1].
In contrast, as can be seen on Figure 2, the $J^{\oplus}$ method converges, which is a very nice result, even if the output of the control is not the ideal one. Of course, this does not prove the absence of other singularities, but the study of these still seems to be out of reach.
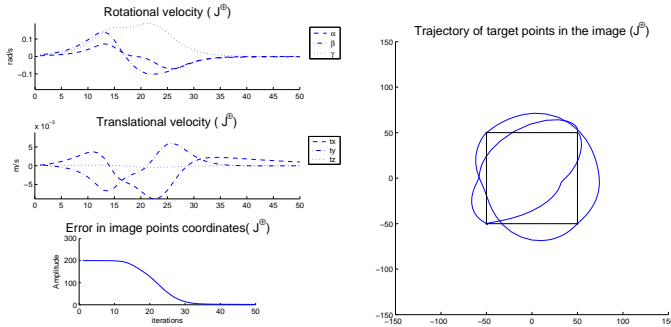

Fig. 2.   Convergence using $J^{\oplus}$ for a 180° rotation around the optical axis

c) Knowledge of the object model: With the $J^{\oplus}$-method, it is not necessary to know the 3D model of the object. But in this case a learning stage (which can be experimentally expensive to perform) must be done with real images of the object. However, this solution has the advantage of allowing an implicit compensation of the system calibration errors. Of course, if the 3D model is available, the learning can be performed easily and quickly through simulations.

V. CONCLUSIONS AND PERSPECTIVES

In this paper, we have proposed a learning method to compute the inverse Jacobian for visual servoing. We have explained why and under which conditions the direct computation of the inverse Jacobian can be a better choice than the computation of the pseudo-inverse of the Jacobian. We presented realistic simulations in which this direct computation was compared with classical methods have been presented. The results plainly justify this approach.

Future work will be devoted to making real experiments, implementing the proposed control law on a robotic system. On the other side, the direct computation of the inverse Jacobian, does not allow any easy stability analysis. This is an important theoretical point that has to be addressed.

Last, but not least, it should be noted that, independently of the servoing context, the proposed method consists in learning the inverse jacobian matrix around the global optimum and that it can be used in full generality to efficiently optimise functions for which the shape around the optimum is previously known.

REFERENCES

[1] F. Chaumette, "Potential problems of stability and convergence in image-based and position-based visual servoing," in *The Confluence of Vision and Control*. LNCIS 237, Springer Verlag, pp. 66–78, 1998.
[2] P. Corke and S. Hutchinson, "A new partitioned approach to image-based visual servo control," *IEEE Trans. on Robotics and Automation*, 17(4):507-515, August 2001.
[3] K. Deguchi, "A direct interpretation of dynamic images with camera and object motions for vision guided robot control," *Int. Journal of Computer Vision*, 37(1):7-20, June 2000.
[4] B. Espiau, F. Chaumette and P. Rives, "A new approach to visual servoing in robotics", *IEEE Trans. on Robotics and Automtion*, 8(3):313-326, June 1992.
[5] M. Girard and A. Maciejewski, "Computational modeling for the computer animation of legged figures," *ACM Computer Graphics*, 19(3):263-270, 1985.
[6] K. Hashimoto, ed., *Visual servoing*, Series in Robotics and Automated Systems, vol. 7,   World Scientific, 1993.
[7] K. Hosoda, M. Asada: "Versatile visual servoing without knowledge of true jacobian" *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, IROS'94*, Munchen, Germany, September 1994.
[8] 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.
[9] M. Iwatsuki and N. Okiyama, "A new formulation of visual servoing based on cylindrical coordinate system with shiftable origin," *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, IROS'02*, Lausanne, Switzerland, October 2002.
[10] M. Jägersand, O. Fuentes, R. Nelson, "Experimental evaluation of uncalibrated visual servoing for precision manipulation", *IEEE Int. Conf. on Robotics and Automation, ICRA'97*, Vol. 3, p. 2874-2880, Albuquerque, New Mexico, April 1997.
[11] F. Jurie and M. Dhome, "Hyperplane approximation for template matching," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 24, 24(7):996-1000, 2002.
[12] E. Malis, F. Chaumette, and S. Boudet, "2 1/2 D visual servoing," *IEEE Trans. on Robotics and Automation*, 15(2):238-250, April 1999.
[13] Y. Mezouar and F. Chaumette, "Path planning for robust image-based control," *IEEE Trans. on Robotics and Automation*, 18(4):534-549, August 2002.
[14] R. Paul, *Robot Manipulators: Mathematics, Programming, and Control*. Cambridge: MIT Press, 1981.
[15] J. Piepmeier, "Uncalibrated eye-in-hand visual servoing", *Int. Journal of Robotics Research*, 22(10/11):805-820, October 2003..
[16] O. Tahri and F. Chaumette, "Application of moment invariants to visual servoing," *IEEE Int. Conf. on Robotics and Automation, ICRA'03*, Vol. 3, pp. 276-4281, Taipei, Taiwan, September 2003.
[17] L. Weiss, "Dynamic visual servo control of robots. an adaptive image-based approach", PhD Thesis, CMU-RI-TR-84-16; Carnegie Mellon University, April 1984.
[18] G. Wells, C. Venaille, and C. Torras, "Vision-based robot positioning using neural networks," *Image and Vision Computing*, 14:715-732, December 1996.
[19] P. Zanne, G. Morel, and F. Plestan, "Sensor-based control in the presence of uncertainties: bounding the task function tracking errors," *IEEE Int. Conf. on Robotics and Automation, ICRA'02*, Washington D.C., May 2002.
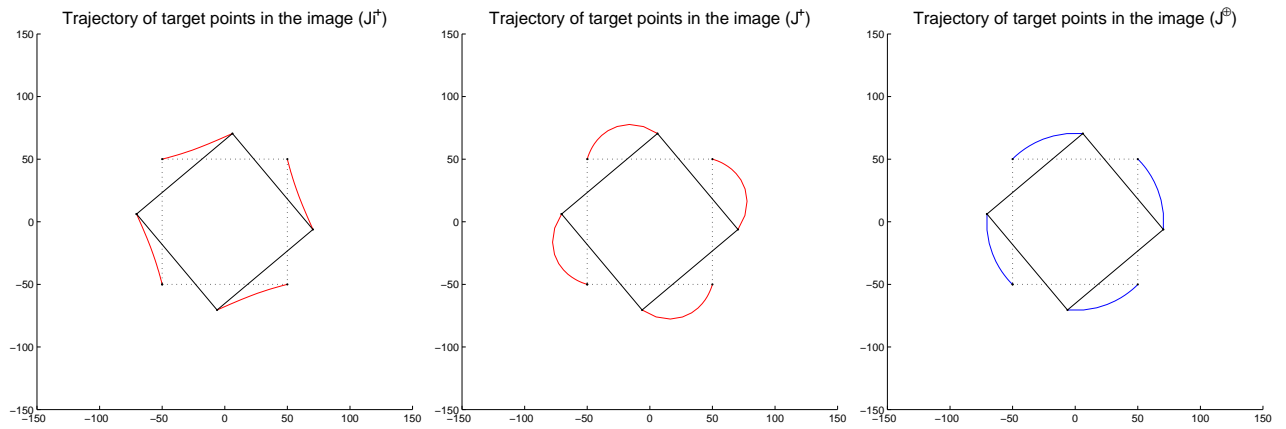
Fig. 3. First simulation (rotation of 50° around the optical axis): image trajectories for the three control schemes
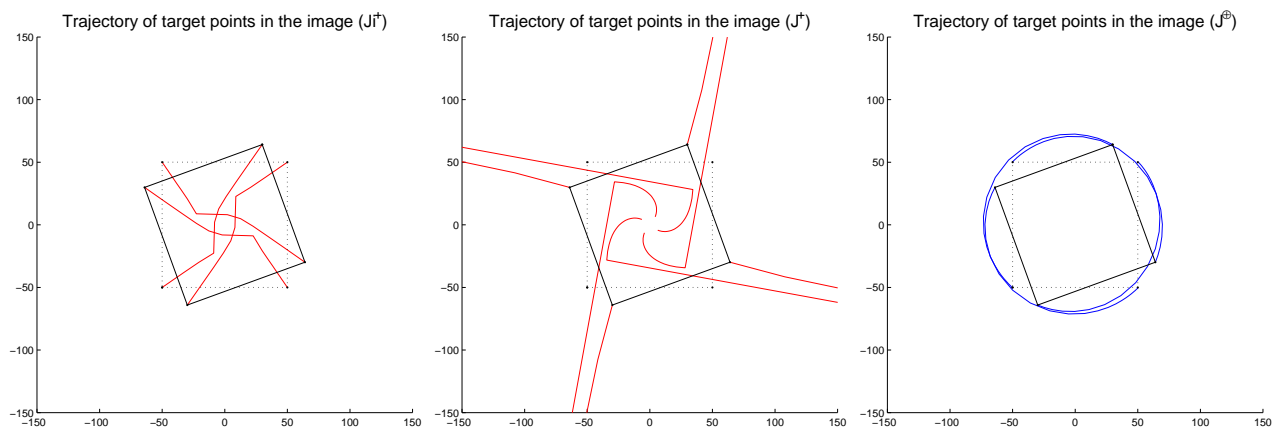

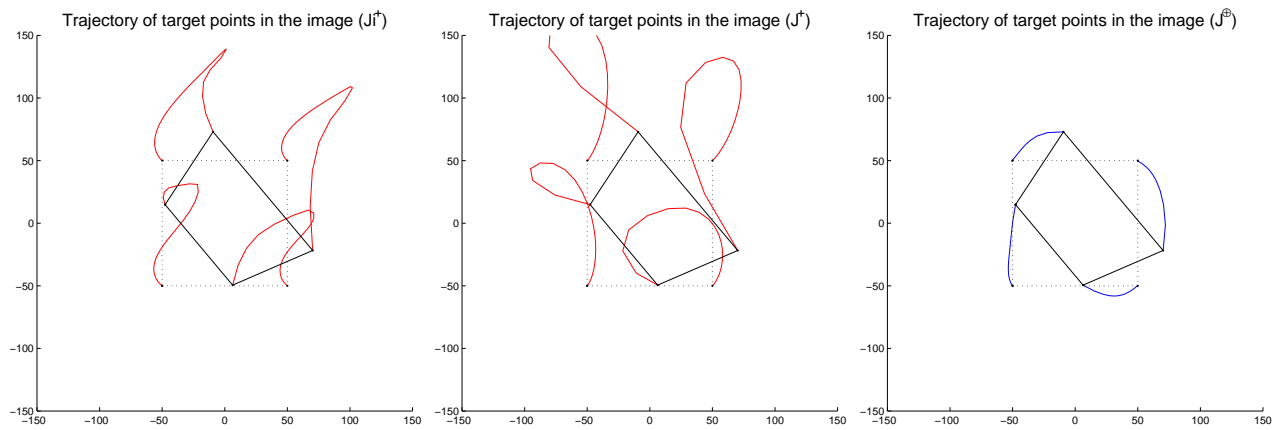Fig. 4. Second simulation (rotation of 160° around the optical axis)

Fig. 5. Third simulation: (large and complex displacement)