Evaluation of Visual Servoing Techniques for Robotic Applications

Santos, Ricardo

Departamento de Engenharia Electrotécnica e de Computadores
Universidade Técnica de Lisboa
Instituto Superior Técnico
Av. Rovisco Pais 1049-001 Lisboa Portugal
rmcdss@ist.utl.pt

Abstract – Control systems based on visual data input, termed visual servoing systems, are one of the most promising fields of research to develop a system as much autonomous as is possible.
In the last years, several methods have been proposed in this area, with their strengths and weaknesses. To clarify it, it is necessary to evaluate and compare them. In this thesis some visual servoing techniques are evaluated with the goal to clarify in which situations each one of them is a good option to solve a specific task. The methods evaluated are: *position based visual servoing*, *image based visual servoing*, *2.5D visual servoing*, *decoupled visual servoing* and *shortest path visual servoing.*

Introduction – I

In the last decades have been increased the research in robotics. The challenge is developing an autonomous system that can act in one unknown environment. To build a system like that, the first concern should be trying to get as much information as possible from the world. The sensor that best adapts to this requirement is the vision. The video camera has the advantages of providing a high accuracy to robotic system and allowing a bigger flexibility to the applicability of the data input. The video camera also gives the possibility to extract different kinds of information from the same sensor.
Visual servoing is the name given to the control techniques that use image data as input. It is a very large field of research and to build a system like that it is necessary to join knowledge from many areas of research: robot modeling, control theory, computer vision and others, [1].
The two most common branches of application for visual servoing are: positioning of a robot in a desired position (typically with the goal of grasping some object) and following targets in movement. Our work considers the first one.
Recently, several new visual servoing methods have been proposed trying to build control systems based on vision as good as possible. These methods are normally presented together with some experimentation to evaluate its performance. However, in many cases, these tests are not very exhaustive.
It is useful to have a global evaluation of the performance of several visual servoing techniques. This kind of work was done by Gans, [2], however, more tests can be done, other features can be measured and other visual servoing methods can be evaluated.
In this paper, different visual servoing approaches are explored and their performances evaluated. The goal is measuring the influence of the image noise in the performance of those visual servoing techniques.
The visual servoing techniques analyzed are: position based visual servoing, [1], image based visual servoing, [1], 2.5D visual servoing, [3], Corke & Hutchinson visual servoing, [4], and shortest path visual servoing, [5].
In this paper first is given, in Section II, a background about the work developed. In Section III are described the visual servoing techniques analyzed. In Section IV are exposed the results obtained. Finally in Sections V and Section VI are discussed the main conclusions of the work done and suggestions to develop in the future.

Background – II

The typical structure of a visual servoing control scheme is exposed in Figure 1.

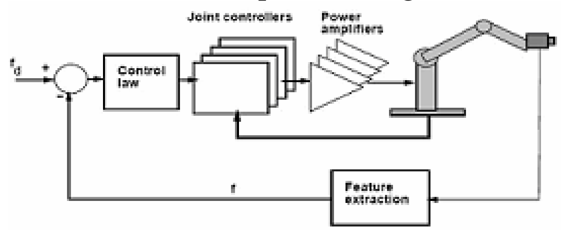

Figure 1 – Control scheme of a visual servoing system; eye – in hand approach.

The goal of a visual servoing system is to move a robot from one initial to one desired position, and keep it there using always a close-loop control scheme. In the beginning of the process, is provided to the system one image of the target with the camera in the desired position. Then, in the starting position, the camera obtains an image of the target. From that image, the necessary features, f, are extracted and compared with the desired ones extracted from the desired image, f_d; the error obtained from the difference between the desired and the current image features is used to compute the control law. The control signal is sent to the joints and a new image, with the camera in the new position, is taken. This is done until could be possible reducing to zero the error between the image features in the current and in the desired position. The features extracted can be for example: points, lines, ellipses, differences in the texture.

The visual servoing control law structure is almost the same for all the methods, changing basically the features used for tracking and the quantity of object *a priori* knowledge used. The definitions presented now are generic for all the methods.

The *feature vector*, s, is defined by the list of features, extracted directly or indirectly from the image, used in the control law and the *features error*, or *task vector*, e, is the difference between the current features vector and the desired features vector.

$$e = s_{current} - s^{*}_{desired} \qquad (1)$$

For an static target it is obtained the equality $\dot{e} = \dot{s}_{current}$.

To ensure the zeroing of the features errors in the task space, it is imposed a restriction in the control law defined by the following first order equation

$$\dot{e} = -\lambda \cdot e \qquad (2)$$

In visual servoing is necessary to make a transformation between the Cartesian space and the feature space. This is done by the *image Jacobian*, J, which relates both dimensions

$$J_q = \frac{\partial e}{\partial q} \qquad (3)$$

The vector $q_{[1x6]}$ represents the position of the joints and ∂q their velocities.

Inverting the relation

$$\dot{e} = \dot{s} = J \cdot v \qquad (4)$$

it is possible to obtain the control law

$$v = -\lambda \cdot J^{+} \cdot \left(s_{current} - s^{*}\right) \qquad (5)$$

The difference between (3) and (4) is that the first is expressed in the joint space, $\left(q, J_q\right)$, and the second in the Cartesian space, $\left(v, J\right)$. The velocity of the camera is defined as:

$$v = \lfloor v_x \quad v_y \quad v_z \quad \omega_x \quad \omega_y \quad \omega_z \rfloor \qquad (6)$$

Typically J is a not square matrix, so in (5), it is necessary to use the pseudo-inverse of J.

An important tool in visual servoing is the homography matrix. This matrix makes the transformation between two image plane (two images). It can be estimated if is known the relation of four points in both images (for a planar target). From this matrix it is possible estimating the displacement of the camera between two images [6], which very useful in several visual servoing techniques.

Evaluated Methods – III

Position Based Visual Servoing

In position based visual servoing (PBVS), the features are extracted indirectly from the

image; it is also called 3D visual servoing. From the data in the 2D image plane it is built a 3D model of the target. The image is used to localize the object in the world.
In PBVS, the feature vector is defined as

$$s = \begin{bmatrix} {}^{c}t_{c*} \\ {}^{c}u_{c*}\theta_{3x1} \end{bmatrix} \qquad (7)$$

${}^{c}t_{c*}$ and ${}^{c}u_{c*}\theta_{3x1}$ represent the translation and the rotation from the initial to the desired position; in this last feature, ${}^{c}u_{c*}$ are the rotation axis and θ_{3x1} are the rotation angles. These features can be obtained from the transformation coordinates from the current camera's position to the desired, ${}^{c}T_{c*}$. That is obtained from the composition of the transformation coordinates from the current position to the target with the desired position to it

$${}^{c}T_{c*} = {}^{c}T_{o}\,{}^{o}T_{c*} = \begin{bmatrix} {}^{c}R_{c*} & {}^{c}t_{c*} \\ 0 & 1 \end{bmatrix} \qquad (8)$$

As the velocity vector and the features vector are both defined in the Cartesian space, the position based Jacobian can be denied as a simple identity matrix:

$$J_{3D} = \begin{bmatrix} I_3 & 0_3 \\ 0_3 & I_3 \end{bmatrix} \qquad (9)$$

And the correspondent control law is

$$v = -\lambda \cdot J_{3D}^{-1} \cdot \dot{s} \qquad (10)$$

PBVS has some disadvantages: it is necessary to have a thorough knowledge about the target to build a specific model of it. This is only possible to apply to objects with a specific shape which also reduces the applicability of the algorithm; other disadvantage is that, due to the control calculus are done in the Cartesian space, the system requires an accuracy calibration; another problem is that this computation, in the Cartesian space, also requires a high computational effort, which reduces the applicability of the algorithm in a real time system.
On the other hand the main advantages of PBVS are the accuracy and robustness of its performance.

Image Based Visual Servoing

In image based visual servoing (IBVS), the target features are extracted directly from the image, without any image interpretation. This control architecture does not need to make a 3D reconstruction of the target.
The feature vector, s, is defined as

$$s = \begin{bmatrix} x & y \end{bmatrix}^{T} \qquad (11)$$

And its error as

$$\dot{s} = \begin{bmatrix} x - x^{*} \\ y - y^{*} \end{bmatrix} \qquad (12)$$

The relation between the features error and the velocity to send to the robot is

$$v = -\lambda \cdot J_{2D}^{+} \cdot \dot{s} \qquad (13)$$

And the image base Jacobian, for a unite camera's focal length is defined as

$$J = \begin{bmatrix} \frac{1}{Z} & 0 & -\frac{x}{Z} & -x \cdot y & \left(1 + x^{2}\right) & -y \\ 0 & \frac{1}{Z} & -\frac{y}{Z} & -\left(1 + y^{2}\right) & x \cdot y & x \end{bmatrix}. \qquad (14)$$

The variables x and y are the metric coordinates of one point in the image and Z the depth of that point.
The Jacobian presented in (14) has the dimension of [2 x 6]. This is the Jacobian size to track one single point. For n points, the Jacobian dimension would be [n 2 x 6].
One problem of IBVS is that only the local convergence can be ensured. Another problem is the difficulty of this method in managing rotations around the optical axis.
To compute the control law for image based visual servoing, it is necessary to track at least three non-collinear points.

2 ½ D Visual Servoing

The 2.5D visual servoing (2.5D) was presented by Malis, [3], in 1999 and was developed with the goal of joining the best skills of the two main visual servoing approaches (IBVS and PBVS).
This hybrid method does not need any 3D model of the object and ensures the convergence of the control law in the whole task space.
In the 2.5D, the task vector is defined

as $e_{[6x1]} = \left(x, y, \log(\frac{Z}{Z^*}), {}^c u_{c*} \theta_{3x1} \right)$.

In which, x and y are the metric coordinates of a point, Z is the current distance of the camera to that point and u_{3x1} and θ_{3x1} are the axis of the rotation and the angles that the camera needs to rotate.
The 2.5D visual servoing Jacobian is defined as

$$J_{2,5D} = \begin{bmatrix} \frac{1}{d^*} \cdot L_v & L_{(v,w)} \\ 0 & I_3 \end{bmatrix} \qquad (15)$$

with

$$L_v = \frac{1}{\rho} \cdot \begin{bmatrix} -1 & 0 & x \\ 0 & -1 & y \\ 0 & 0 & -1 \end{bmatrix} \qquad (16)$$

$$L_{v,w} = \begin{bmatrix} xy & -(1+y^2) & y \\ (1+y^2) & -xy & -x \\ -y & x & 0 \end{bmatrix} \qquad (17)$$

In which d^* is the distance of the camera to the target in the desired position and ρ is defined by the ration Z / d^*.
Then, the control law is:

$$v = -\lambda \cdot J_{2.5D}^{-1} \cdot e \qquad (18)$$

Corke & Hutchinson Visual Servoing

The Corke and Hutchinson visual servoing (CH), [4], was developed to address some of the IBVS problems, mainly the camera retreat problem, which occurs for pure rotations around the camera's optical axis. The CH method has the drawback that its performance depends of the relative position of the features in the image.
The error vector is defined as in IBVS

$$e = \begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = \begin{bmatrix} x - x^* \\ y - y^* \end{bmatrix} \qquad (19)$$

In this method, only one point is tracked, and (x, y) are their image plane coordinates.
In CH method the translation, v_z, and the rotation, ω_z, velocities along/around z-axis are computed decoupled from the other velocities and are extracted from the next features:
Theta Feature, θ - drive the camera inclination around the optical axis.. The theta feature is the angle between the horizontal pixel axis and a straight line built from two points in the image..
Sigma Feature, σ – control the movement thought the z-axis. It consists in the square root of the area of a polygon made from the image points.
The translational and rotational velocities along/around the optical axis are given by:

$$\begin{bmatrix} v_z \\ \omega_z \end{bmatrix} = \begin{matrix} \lambda_{Tz}(\sigma^* - \sigma) \\ \lambda_{\omega z}(\theta_{ij}^* - \theta_{ij}) \end{matrix} \qquad (20)$$

λ_{Tz} and $\lambda_{\omega z}$ are gain factors to adapt the speed of conversion along the associated directions. Then, the others velocities ($v_{xy} = \lfloor v_x \quad v_y \quad \omega_x \quad \omega_y \rfloor$,) can be obtained using the expression:

$$v_{xy} = -\left[J_{xy}^+\right]_{4x2} \left\{ \begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} + \begin{bmatrix} \frac{x}{Z} & y \\ \frac{y}{Z} & -x \end{bmatrix} \cdot \begin{bmatrix} \dot{\sigma} \\ \dot{\theta} \end{bmatrix} \right\} (21)$$

with

$$J_{xy} = \begin{bmatrix} -\frac{1}{Z} & 0 & x \cdot y & -(1+x^2) \\ 0 & -\frac{1}{Z} & 1+y^2 & -x \cdot y \end{bmatrix} (22)$$

Shortest Path Visual Servoing

The shortest path visual servoing (SP) was presented by Kyrki and Kragic, [5], with the goal of addressing two typical problems in the visual servoing: the possibility of the target leaving the camera's field of view and the risk of the joints working too close to their limits. This method was developed in order to the camera trajectory presenting one straight line which goes from the initial to the desired position. In the shortest path visual servoing the position based control is used directly to control the translation of the camera. To control the rotation around x and y axis is used one virtual point which lies in the zero of the object frame; this point

helps the control scheme to keep the target in the image. The rotation over the optical axis is controlled using the ${}^{c}u_{z c^*}\theta$, as in 2.5D visual servoing, which should be driven to zero.
The task vector is

$$e_{[6x1]} = \left({}^{c}X_{c^*}, {}^{c}Y_{c^*}, {}^{c}Z_{c^*}, x, y, {}^{c}u_{z c^*}\theta\right)^T .$$

The transformation from the initial position to the desired can be obtain, as in PBVS,

$$ {}^{c}T_{c^*} = {}^{c}T_o \, {}^{o}T_{c^*} = \begin{bmatrix} {}^{c}R_{c^*} & {}^{c}t_{c^*} \\ 0 & 1 \end{bmatrix} \qquad (23)$$

Due to the simple mathematical definition that J_{SP} has, it is possible to define directly its inverse, J_{SP}^{-1}, which facilitate its implementation.

$$J_{SP}^{-1} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ -\frac{X \cdot Y}{Z \cdot P} & \frac{X^2 \cdot Z^2}{Z \cdot P} & -\frac{Y}{P} & -\frac{X \cdot Y}{P} & \frac{X^2 + Z^2}{P} & \frac{X}{Z} \\ -\frac{Y^2 + Z^2}{Z \cdot P} & \frac{X \cdot Y}{Z \cdot P} & \frac{X}{P} & -\frac{Y^2 + Z^2}{P} & \frac{X \cdot Y}{P} & \frac{Y}{Z} \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} (24)$$

With $(X,Y,Z)^T = \left({}^{c}X_o, {}^{c}Y_o, {}^{c}Z_o\right)^T$ and $P = \dfrac{X^2 + Y^2 + Z^2}{Z^2}$.

Attending in J_{SP} matrix, its determinant is $|Z| = \dfrac{X^2 + Y^2 + Z^2}{Z^2}$, which it is never zero, except when the camera is on the zero of the object frame, which it is not possible. This ensures global convergence of the method.

Simulation Results - IV

Simulation Methodology

The simulations were performed in VISP (VIsual Servoing Platform). The VISP, [7], is a C++ library that has been developed in the last years, with the goal of integrating as much visual servoing tools as is possible and allowing a fast implementation of it in any specific robot, with little work.
It was simulated a robotic arm with six degrees of freedom using the eye-in-hand approach.The desired camera position was set as being located one meter far from the target, oriented to it, and without any inclination: P* = [0, 0, 1, 0, 0, 0]. The first three values of P* correspond to the pose of the camera and the last three to its orientation.
To make the performance comparison between the visual servoing techniques under analysis, it was defined one target consisting in five points; four building a square centered in the target frame and one fifth in its center.
The simulations measuring the influence of the image noise, η, in the performance of the visual servoing techniques for different initial camera configurations. For that, it was added random noise, with variance ranging between 0 and 1 pixel. The desired image, it was assumed as got it in perfect conditions and any noise was introduced on it.
In the simulations, the system reaches the goal when the average pixels error is less than 1 pixel; The system is halted as not succeeded whether the camera had displaced more than 11 meters from the target along the z direction, or more than 2 along the x or y direction, whether had spent more than 200 iterations, or whether the average pixels error change less the 0.05 pixels after 5 consecutive iterations. To ensure that the camera does not collide with the target, it is also required that the camera does not get closer than 0,2 meters from it. To obtain smoother results, less influenced by outliers, each test was executed 50 times and it was made the average of these values.

Tasks Tested

The tasks tested were:
Rotation Around the Optical Axis
Translation Along the Optical Axis
Rotation Around an Axis Coplanar with the Target Plane
Translation Along an Axis Parallel to the Target Plane
Generic Motions

Metrics Evaluated

The metrics used to evaluate the performance of the visual servoing methods studied were:

Final Pixel Error, which indicates whether the algorithm had success in reaching its goal. Any value under 1 pixel is equally considered successful.
Number of Iterations until the algorithm being halted, as have had success or not.
Maximum Camera Distance from the desired position during the execution of the algorithm.
Maximum Feature Point Excursion, which consists in the maximum distance of all the point tracked to the center of the image during the algorithm execution. The value of this feature in the desired position is 56 pixels.
Simulation Time of the algorithm.

Results

Rotation Around the Optical Axis

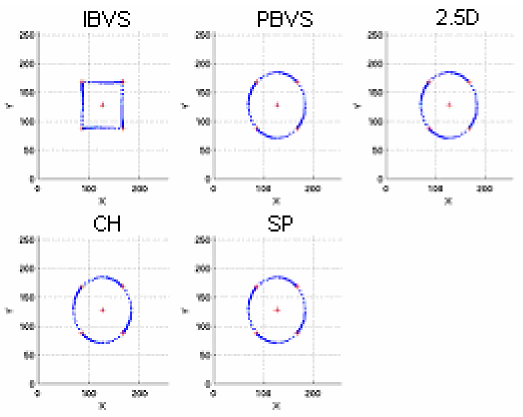

Figure 2 – Trajectory of the points in the image plane. P_0=[0,0,1,0,0,90], η =0.

Final Pixels Error

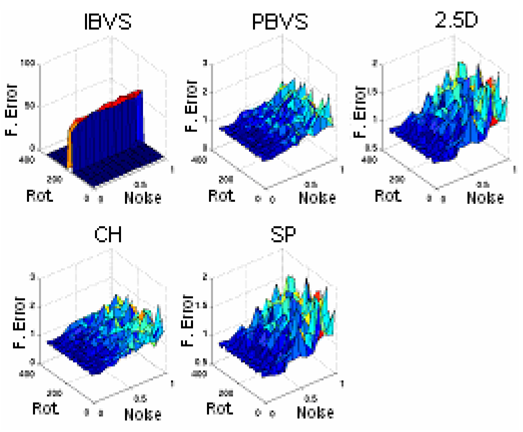

Figure 3 – Rotation around the optical axis – final pixels error.

Number of Iterations

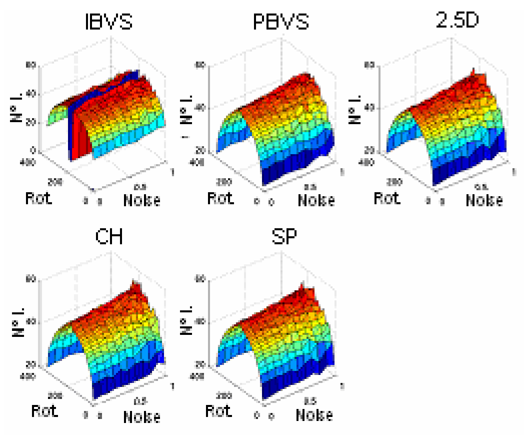

Figure 4 – Rotation around the optical axis – number of iterations.

Maximum Camera Distance from the Desired Position

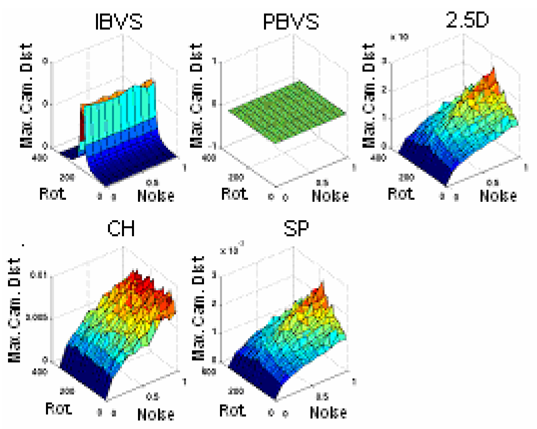

Figure 5 – Rotation around the optical axis – maximum camera distance from the desired position.

Maximum Feature Point Excursion

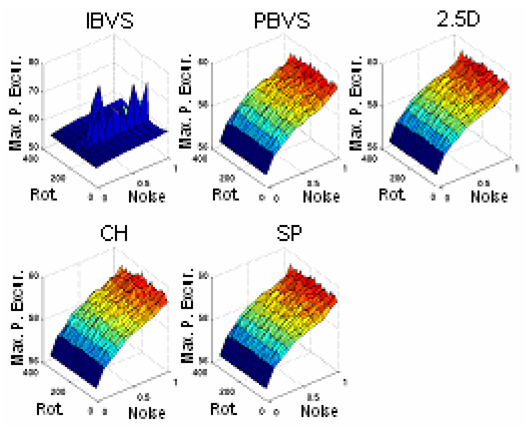

Figure 6 – Rotation around the optical axis – maximum feature point excursion.

Discussion

In this test was evaluated the performance of the visual servoing techniques in a pure rotation of the camera around its optical axis. It is possible to observe that IBVS could not reach always its goal. In IBVS the system tries that, the points tracked move in the screen along one straight line, from the initial position to the desired one, see Figure 2. For that, the camera needs to retreat. For a rotation of 180º, the camera theoretically needs to retreat until infinite becoming the system unstable. For the other four methods the motion in the Cartesian space is almost null and for PBVS it is exactly zero. Beyond IBVS, the methods performed correctly. In these cases, there is some dependence of the final pixel error with the increase of the noise.
In the maximum feature point excursion, the 2.5D, CH and SP have not presented dependency with the level of the rotation. This happens because it was tracked a point which lies in the center of the image. PBVS had the same behavior but in this case, the results are independent of where the points tracked lie. The number of iterations was almost the same for all methods. It had an important dependence with the initial camera inclination, and just a little with the increase of the noise.

Translation Along the Optical Axis

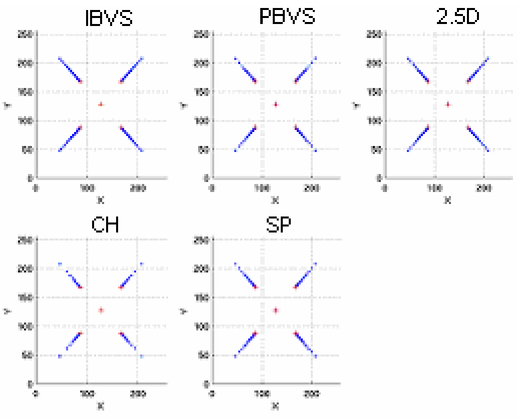

Figure 7 – Trajectory of the points in the image plane. P_0=[0,0,0.5,0,0,0] , η=0.

Final Pixels Error

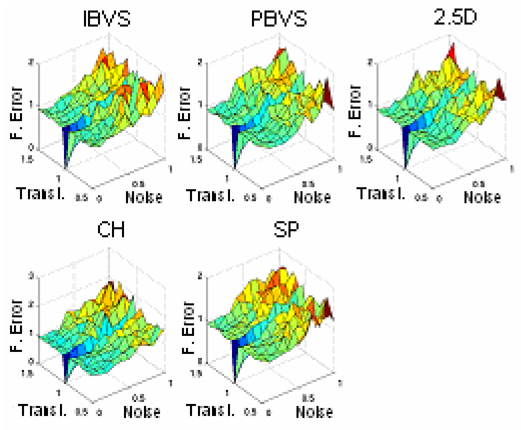

Figure 8 – Translation along the optical axis – final pixels error.

Number of iterations

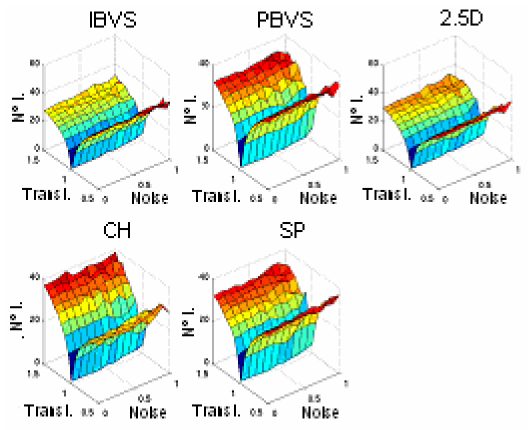

Figure 9 – Translation along the optical axis – number of iterations

Discussion

In this test was evaluated the performance of the visual servoing techniques for different initial camera positioning along its optical axis. All methods could put the final error under one pixel when without noise, although these values grow linearly with the increase of the image noise. With exception of the CH, all methods had a symmetric performance along the translations axis, in the final pixels error. In CH, the final error values are higher when, in the presence of the noise, the camera starts the movement farther away from the target; when the camera starts its displacement at 0.5 meters from the target (the closest position), the final error, with the maximum noise, is

around 1.5 pixels, performing in this case, even better than the other four methods.
About the number of iterations all graphics present just a little increase of the number of iterations with the increase of the image noise.

Rotation Around an Axis Coplanar with the Target Plane

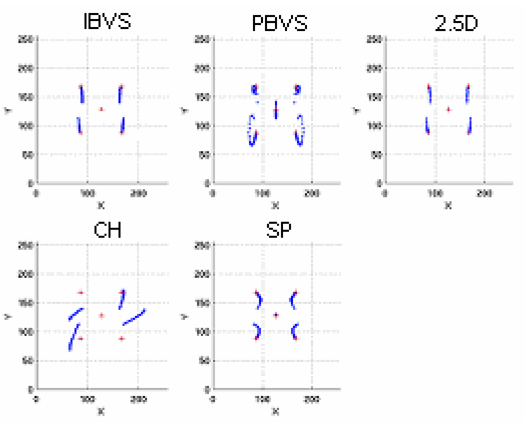

Figure 10 – Trajectory of the points in the image plane. P_0=[0,0,1,70,0,0]. , η=0.

Final Pixels Error

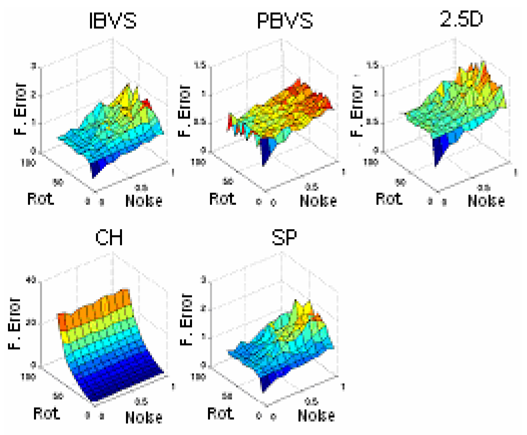

Figure 11 – Rotation around an axis coplanar with the target plane – final pixels error.

Maximum Feature Point Excursion

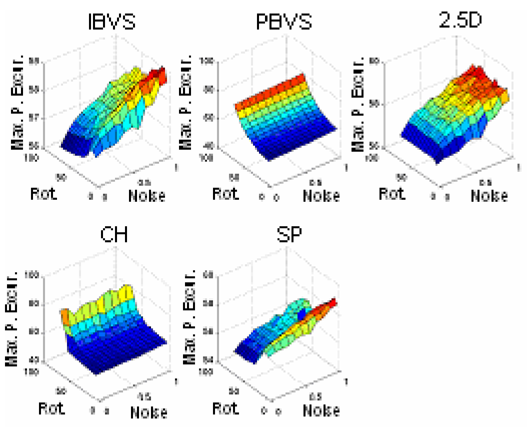

Figure 12 – Rotation around an axis coplanar with the target plane – maximum feature point excursion.

Discussion

As can be seen in Figure 10, for high initial camera angles, the features can become much skewed, so this is a hard test for the visual servoing techniques.
IBVS, 2.5D and SP despite of growing the final pixels error with the increase of the image noise, could perform correctly in this test. The 2.5D was the best of these, once it was the fastest in number of iterations and had the smallest final error.
In PBVS the final pixels error increased also with the increase of noise. However, in this test, it was necessary using a higher value of the gain, 5xλ, to avoid that the system stayed blocked in local minimums. The value of λ for this test was too small in comparison of the stop criterion.
The maximum feature point excursion in 2.5D, IBVS and SP grows with the increase of the noise from 56, until around 59 pixels. In PBVS the values grow as a parable, with the levels of the initial camera inclination and nothing with the noise. The CH had a performance similar with PBVS but, much more non-linear. CH only had good results for small levels of initial camera inclination, until around 20º, performing in this case well, even in the presence of image noise. After that level of initial inclination, it gets easily stopped in local minimums and has a high final error and maximum feature point excursion.

Translation Along an Axis Parallel to the Target Plane

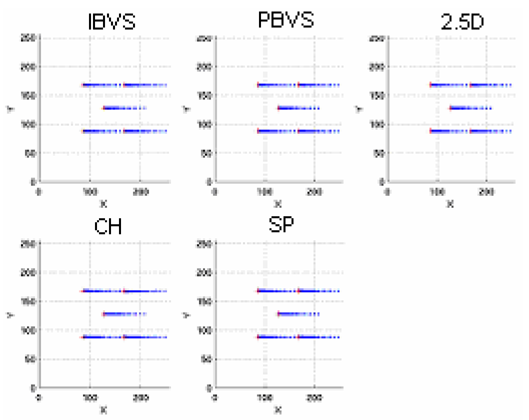

Figure 13 – Trajectory of the points in the image plane. P_0=[0.2,0,1,0,0,0] , η=0..

Final Pixels Error

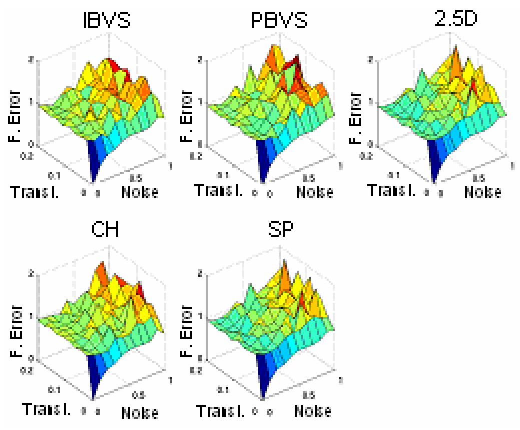

Figure 14 – Translation along an axis parallel to the target plane - final pixels error.

Discussion

For this test, all methods performed correctly. Figure 14 show that the final pixels error, in presence of noise, is a little higher in PBVS.

Generic Motions

For a final evaluation, it was performed tests in which the camera position moves along several dimensions at the same time. These were the most challenging tests made. In these tests, the initial position of camera was set in the following positions:

1 - P_0 = [0.2, 0.2, 1.5, 0, 0, 0], η = 1

In this test the camera needs to make a translation along the xx, yy and zz axis from the initial to the desired position. No rotation is required.

2 - P_0 = [0.2, 0, 1.5, 0, 0, 180], η = 1

This test, beyond the translation (the same as in Test 1) requires a rotation of 180º around the camera's optical axis.

3 - P_0 = [0, 0, 1.5, 30, 0, 90], η = 1

This test involves a translation and rotation along the optical axis of the camera, and a rotation around an axis coplanar with the target plane.

4 - P_0 = [0, 0, 1.5, 60, 0, 90], η = 1

In this test, it is evaluated the same as in the previous one, but with a higher level of inclination around the axis coplanar with the target plane.

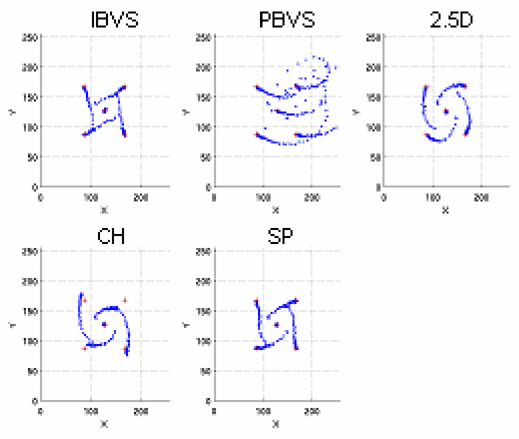

Figure 15 – Trajectory of the points in the image plane. P_0=[0,0,1.5,60,0,90] , η = 1.

Discussion

The CH method had a very good performance when the camera is almost perpendicular with the target plane. When the camera starts the motion, with its orientation far to be perpendicular with target plane, the method performs badly. This happens because the CH method uses the area of the target as a feature and this becomes much skewed with the inclination of the camera in relation of the target plane.

In PBVS was possible to see that, it is the one that has the highest maximum feature excursion. In the last test, one of the points left the camera's field of view. This occurs because in this method, there is not any control over the image plane features. In general PBVS took more iterations than the other methods to reach the goal.

In IBVS it was possible to observe that the problem of Chaumette Conundrum happens for rotation around the optical axis of the camera, independently of the translation that it needs to do.
The 2.5D and the SP were the only methods that had always acceptable results in all tests effectuated. The SP was always the fastest method in execution time and after it, the second was the CH.

Conclusions - V

After the conclusion of this work, it is possible better understanding the behavior of the visual servoing methods studied.
The ViSP proved to be a very useful, flexible and user-friendly tool.
The image based visual servoing had problems with rotations around the optical axis; the system becomes completely unstable for levels of rotation above 160º, and under 200º, and this behavior is independent of the translation that the camera needs to do. In the presence of noise, the IBVS was which performed best, minor final error, which was expected due to this method does not use the homography matrix to compute the control law.
The Corke & Hutchinson method performed correctly for the specific task for which it was designed, rotation around the optical axis. It performed also perfectly for translation through the optical axis. However, for generic motions, which involve rotation around axis coplanar with the target plane, it performs poorly. When the camera is almost perpendicular with the target plane it works very well, even in the presence of the noise. However when the camera starts its displacement in an initial position closer to be coplanar with the target plane, it can not reaches its goal. In this sense, this seems to be a weak method to be used in a generic application.
The position based visual servoing had also a good behavior, although for generic motions, with large rotations, it could be seen that the points tracked can leave the camera's field of view. This happens because in this technique, there is not any control over the image plane features
The 2.5D visual servoing performed correctly in all tests and it was the most versatile method. The shortest path visual servoing worked also correctly in all tests, and had behavior similar to 2.5D.
In matter of execution time, the shortest path visual servoing was always the fastest method followed by the Corke & Hutchinson visual servoing.

Future Work - VI

To go further in the evaluation of visual servoing techniques, other kind of conditions that can affect the performance of the system can be evaluated. These include for example, errors in the camera calibration, errors in the robot kinematics and different kinds of depth estimation.
In order to get a better knowledge about the methods evaluated, a next step could be for example implementing, and testing them in a real system.

References

[1] – Hutchinson, S., Hager, G. & Corke, P., 1996, "A tutorial on visual servo control", IEEE Transactions on Robotics and Automation 12(5), 651-670.
[2] – Gans, N. R., Hutchinson, S. A., & Corke, P. I., 2003, "Performance tests for visual servo control systems, with application to partitioned approaches to visual servo control" Int. J. Robot. Res., vol. 22, no. 10, pp. 955–981.
[3] – Malis, E., Chaumette, F., & Boudet, S., Apr. 1999, "2-1/2-D visual servoing", IEEE Transactions on Robotics and Automation, vol. 15, no. 2, pp. 238–250.
[4] – Corke, P. I. & Hutchinson S. A., Aug. 2001, "A new partitioned approach to image-based visual servo control" IEEE Transactions on Robotics and Automation, vol. 17, no. 4, pp. 507–515.
[5] – Kyrki, V., Kragic, D., and Christensen, H., Oct. 2004, "New shortest-path approaches to visual servoing" in IEEE Int. Conf. on Intelligent Robots and Systems, Sendai, Japan, pp. 349–355.
[6] – Malis, E., Nov. 1998, "Contributions à la modélisation et à la commande en asservissement visuel", PhD thesis, Université of Rennes I, IRISA.
[7] – Marchand, E., Spindler, F. & Chaumette, F., Dec. 2005, "ViSP for visual servoing: a generic software platform with a wide class of robot control skills". IEEE Robotics and Automation Magazine, Special Issue on "Software Packages for Vision-Based Control of Motion", P. Oh, D. Burschka (Eds.), 12(4):40-52.