accepted to the IEEE Int. Conf on Inteligent Robots and Systems, Sendai, Japan, September 28 - October 2 2004.

Tasks Sequencing for visual servoing

Nicolas Mansard, François Chaumette
IRISA - ENS Cachan and INRIA Rennes
Campus de Beaulieu, 35042 Rennes, France
E-Mail : Firstname.Lastname@irisa.fr

***Abstract*— Classical visual servoing approaches tend to constrain all degrees of freedom (DOF) of the robot during a task's execution. In this article a new approach is proposed. The key idea is to control the robot with a very under-constrained task when it is far of the desired position, and to incrementally constrain the global task by adding further tasks as the robot moves closer to the goal. A method is first proposed that stacks elementary tasks until the robot is fully constrained. To insure the continuity of the articular velocities when adding constraints, a new control law is then proposed. Experiments that prove the interest of the approach are also provided.**

I. Introduction

Visual servoing provides very efficient solutions to control robot motions from an initial position to a precise goal [7]. It supplies high accuracy for the final pose, and good robustness to noise in image processing, camera calibration and other setting parameters. However, if the initial error is large, such a control may become erratic [1]. Approaches such as 2-1/2-D [8] or path planning [3], [10] provide solutions that enlarge the region where the system converges. But they each constrain all available robot DOF from the beginning. This imposes an unique trajectory, while in this paper, we propose to rather use very low constraints when the robot is far from the goal, in order to enlarge the trajectories available. Constraints are progresively added as the robot approches the required position.

This paper deals with tasks sequencing [11], [12], [14], and describes a solution to stack elementary tasks one on top of the other until all degrees of freedom of the robot are constrained, and the desired position is reached. A vast number of trajectories are usually available to reach the goal. Indeed, by constraining all DOF from the beginning, the classical controls choose a particular trajectory, without knowing if it is valid or not. In some particular cases, this can lead to singularity or instability problems. To always obtain an ideal execution, a first solution is to plan the trajectory. For example by using the potential field method [3], [10]. The idea is to choose an optimal trajectory among all the available trajectories. This provides a complete solution, which ensures optimality, stability and physical feasibility until the goal when it is reachable. Path planning solves the deficiencies of basic approaches, but by applying even greater constraints to the trajectory of the robot. This means, however, that this solution is less reactive to changes of the goal, environment or constraints. Rather than decide in advance which path should be used to reach the goal, switched systems use a set of subsystems along with a discrete switching control [5], [2]. The robot will then avoid difficult regions by switching from a first control law (a particular trajectory) to another one when necessary. This enlarges the stable area to the union of the stable area of each task used.

The key idea of this study is to use the least amount of DOF when the robot is far from the goal, and to add constraints only as the goal comes closer. At each step, the robot moves to achieve an elementary task, maintaining all the elementary tasks already completed. At the end, the robot is entirely constrained by the sum of the constraints of each elementary task. Additional constraints such as joint limit avoidance can also be added using the remaining DOF. Our scheme consists of three phases:

First, the elementary tasks are chosen. These tasks do not have to be properly decoupled. The only restriction is that each DOF of the robot should be constrained by at least one task. This is a difficult choice, currently done off-line by the programmer. He also has to choose the nature and order of the tasks to be applied.

A control law then has to be computed from the selected elementary task. The idea is to maintain the elementary tasks already achieved when moving the robot according to the last elementary task. This is done by a stack of tasks, using the redundancy formalism introduced in [13]. It guarantees that every new task added will not disturb the ones already achieved.

The last step consists of insuring a smooth transition when adding a new elementary task. As shown in [14], one cannot insure a continuous switching by using a first order control law like in classical servoing. One can alternatively use a second order control law [14], or a non homogeneous first order control law as described below.

Section II will recall the redundancy formalism, and presents the adaptation done to apply it to a stack of several tasks (more than two). In Section III, a way to insure a continuous control law is given. The experiments and results are finally set out in Section IV.

II. Stack of tasks using redundancy formalism

This section presents the redundancy formalism [13], and the way it is used to stack elementary tasks. It has first been used for visual servoing in [4], and in numerous applications since (e.g. avoiding visual features occultation

[9], or human-machine cooperation using vision control [6]). The idea is to use the DOF left by a first task having priority, to realize a secondary task at best without disturbing the first one.

A. Redundancy formalism

Let $\mathbf{q}$ be the articular vector of the robot. Let $\mathbf{e_1}$ and $\mathbf{e_2}$ be two tasks, $\mathbf{L_1} = \frac{\partial \mathbf{e_1}}{\partial \mathbf{q}}$ and $\mathbf{L_2}$ their jacobian. The robot is controled in speed, i.e. by computing the articular velocity vector $\dot{\mathbf{q}}$. We use the command

$$\dot{\mathbf{q}} = -\lambda \cdot \mathbf{e} \tag{1}$$

where λ is a custom parameter chosen to regulate the convergence speed of the global task $\mathbf{e}$. This equation will be explained in detail in Section III.

We want to combine the two tasks, using $\mathbf{e_2}$ as a secondary goal realized under the constraint $\mathbf{e_1} = 0$. This clause can be formulated mathematically:

$$\dot{\mathbf{e_1}}|_{\dot{\mathbf{q}}_1} = \dot{\mathbf{e_1}}|_{\dot{\mathbf{q}}_{12}} \tag{2}$$

where $\dot{\mathbf{e_1}}|_{\dot{\mathbf{q}}_1}$ is the variation of the task $\mathbf{e_1}$ using the control law $\dot{\mathbf{q}}_1$ which does not take the second task into account, and $\dot{\mathbf{e_1}}|_{\dot{\mathbf{q}}_{12}}$ is the variation of the task $\mathbf{e_1}$ using a control law $\dot{\mathbf{q}}_{12}$ which realizes both tasks.

A task $\mathbf{e}$ that realizes (2) is

$$\mathbf{e} = \mathbf{e_1} + \mathbf{P_1} \cdot \mathbf{e_2} \tag{3}$$

where $\mathbf{P_1}$ is the orthogonal projection operator on the null space of $\mathbf{L_1}$. Introducing (3) in (1), one gets:

$$\dot{\mathbf{e_1}}|_{\dot{\mathbf{q}}_{12}} = \mathbf{L_1} \cdot \dot{\mathbf{q}}_{12} = -\lambda \cdot \mathbf{L_1} \cdot \mathbf{e_1} - \lambda \cdot \mathbf{L_1} \cdot \mathbf{P_1} \cdot \mathbf{e_2} \tag{4}$$

Since $\mathbf{P_1} \cdot \mathbf{e_2}$ is in the null-space of $\mathbf{L_1}$, the second part of the sum is 0. Thus

$$\dot{\mathbf{e_1}}|_{\dot{\mathbf{q}}_{12}} = -\lambda \cdot \mathbf{L_1} \cdot \mathbf{e_1} = \dot{\mathbf{e_1}}|_{\dot{\mathbf{q}}_1} \tag{5}$$

The task $\mathbf{e}$ introduced in (3) realizes at best $\mathbf{e_2}$ without altering $\mathbf{e_1}$, as specified.

B. Using redundancy formalism with three tasks and more

Equation (3) enables to stack two elementary tasks. However we want to add as many tasks as needed, until there is no DOF left. Let $\mathbf{e_1} \dots \mathbf{e_n}$ be n different tasks, and $\mathbf{L_1} \dots \mathbf{L_n}$ their jacobians. One may think that the solution is simply to project the last task n on the null space of the previous one, and then project this composed task on the null space of the task. The result would be:

$$\mathbf{e} = \mathbf{e_1} + \mathbf{P_1} \cdot \big(\mathbf{e_2} + \mathbf{P_2} \cdot (\dots + \mathbf{P_{n-1}} \cdot \mathbf{e_n})\big) \tag{6}$$

That is:

$$\mathbf{e} = \mathbf{e_1} + \mathbf{P_1} \cdot \mathbf{e_2} + \mathbf{P_1} \cdot \mathbf{P_2} \cdot \mathbf{e_3} + \dots \prod_{i=1}^{n-1} \mathbf{P_i} \cdot \mathbf{e_n} \tag{7}$$

However, since the projection operators do not commute, $\mathbf{P_1} \cdot \mathbf{P_2} \cdot \mathbf{e_3}$ is not in the null space of $\mathbf{e_2}$. Thus, task $\mathbf{e_2}$ will be modified by $\mathbf{e_3}$ and by each supplementary task, which is not an adequate behaviour.

Two better solutions are proposed. Let it be assumed that a task $\mathbf{e_{1\dots n}}$, realizes the n first tasks while respecting the priority constraints (task i should not disturb task j, when $j < i$). A supplementary task $\mathbf{e_{n+1}}$ is to be added with respect to the n first tasks. The first idea is to consider the first n elementary tasks as a big one, named $\mathbf{e_{1\dots n}}$, and to project $\mathbf{e_{n+1}}$ in the null space $N_{1\dots n}$ of this task. The orthogonal projection operator $\mathbf{P_{1\dots n}}$ onto $N_{1\dots n}$ is computed from the jacobian $\mathbf{L_{1\dots n}}$ of the task, which is

$$\mathbf{L_{1\dots n}} = \frac{\partial \mathbf{e_{1\dots n}}}{\partial \mathbf{q}} = \frac{\partial (\mathbf{e_1} + \mathbf{P_1} \cdot \mathbf{e_2} + \dots + \mathbf{P_{1\dots n-1}} \cdot \mathbf{e_n})}{\partial \mathbf{q}} \tag{8}$$

That is:

$$\begin{aligned} \mathbf{L_{1\dots n}} = \mathbf{L_1} + \mathbf{P_1} \cdot \mathbf{L_2} + \dots + \mathbf{P_{1\dots n-1}} \cdot \mathbf{L_n} \\ + \tfrac{\partial \mathbf{P_1}}{\partial \mathbf{q}} \cdot \mathbf{e_2} + \dots + \tfrac{\partial \mathbf{P_{1\dots n-1}}}{\partial \mathbf{q}} \cdot \mathbf{e_n} \end{aligned} \tag{9}$$

The $\frac{\partial \mathbf{P_{1\dots k}}}{\partial \mathbf{q}}$ are very difficult to compute, therefore they are assumed to be $\mathbf{0}$, which seems to be a satisfactory approximation around the position $\mathbf{e_{1\dots n}} = 0$. $\mathbf{P_{1\dots n}}$ is finally the projection operator onto

$$N_{1\dots n} = Null\big(\mathbf{L_1} + \mathbf{P_1} \cdot \mathbf{L_2} + \dots + \mathbf{P_{1\dots n-1}} \cdot \mathbf{L_n}\big) \tag{10}$$

This approximation introduces lags. When the control due to $\mathbf{e_{n+1}}$ is strong (i.e. $\mathbf{e_{n+1}}$ is high), the n first errors $\mathbf{e_1} \dots \mathbf{e_n}$ do not remain at 0 as required (see Fig. 1.a).

A better solution is to project the task into the space $N_{1\dots n}$ of the motions left available by the first n tasks, where $N_{1\dots n}$ is defined as the intersection of the null spaces of each task.

$$N_{1\dots n} = \bigcap_{k=1}^{n} Null(\mathbf{L_k}) \tag{11}$$

The intersection is computed by stacking the n jacobians.

$$N_{1\dots n} = Null \begin{pmatrix} \mathbf{L_1} \\ \vdots \\ \mathbf{L_n} \end{pmatrix} \tag{12}$$

and can be easily obtained with a S.V.D. The projection operator computed from the jacobians is no longer an approximation. No more lag occurs, except those due to modeling errors in the jacobian matrices $\mathbf{L_k}$ (see fig 1.b).

C. Computing the control law using a stack of tasks

We used a stack structure for our experiments. Tasks are stacked one by one. The bottom of the stack (task $\mathbf{e_1}$) has priority. The top of the stack (task $\mathbf{e_n}$) constraints the DOF left by the bottom. Adding or removing a constraint is as easy as putting a new task in the stack.

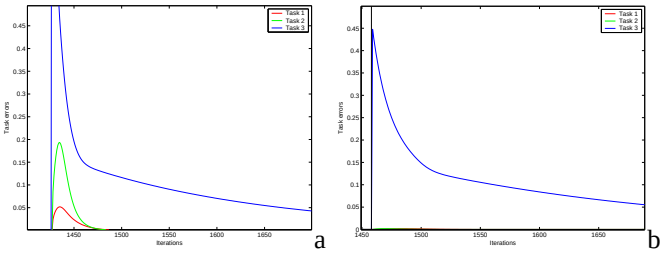


Fig. 1. Adding a new task : (a) the first task errors do not remain to 0 using the approximation (10) to compute the projection operator. (b) The errors remain to 0 using (12). (See Section IV for details about the chosen tasks.)

III. Smooth transition

The robot is controlled by the articular velocity $\dot{\mathbf{q}}$. The control law has to be continuous. Since, a break of continuity means an infinite acceleration during a short period of time, which implies that the control will not be correctly applied. Discontinuities may occur when we add a new elementary task into the stack.

Usually, the control is computed from the following equation that constrains the behavior of the task function:

$$\dot{\mathbf{e}} = f_1(\mathbf{e}) = -\lambda \cdot \mathbf{e} \tag{13}$$

Since $\dot{\mathbf{e}} = \mathbf{L} \cdot \dot{\mathbf{q}}$, we obtain:

$$\dot{\mathbf{q}} = -\lambda \cdot \widehat{\mathbf{L}^+} \cdot \mathbf{e} \tag{14}$$

where $\mathbf{L}^+$ is an approximation of the pseudo-inverse of $\mathbf{L}$ and λ is used as a parameter to control the robot speed. The f_1 function in (13) is chosen by the programmer to link $\dot{\mathbf{e}}$ and $\mathbf{e}$. One chooses generally $f_1(\mathbf{e}) = -\lambda \cdot \mathbf{e}$ to set an exponential decoupled decreasing of the error. The task $\mathbf{e}$ is so that a good approximation of $\mathbf{L}^+$ is the identity matix $\mathbb{I}$. Equation (14) is thus equivalent to (1). The problem of continuity is due to the lack of constraints on the initial value of $\dot{\mathbf{e}}$.

Let $\mathbf{e_A}$ be a global task, used to drive the robot until time 0. At time $t = 0$, the control law switches to a second task $\mathbf{e_B}$. Since $\mathbf{e}$ and $\dot{\mathbf{q}}$ are linked linearly, no continuity guarantee can be ensured on $\dot{\mathbf{q}}$. At time t=0, $\dot{\mathbf{q}}$ is not continuous. A first solution was proposed in [12], using a mixed control during a short transient time after t=0 to ensure the continuity ($\dot{\mathbf{q}} = (1 - \theta(t)) \cdot \dot{\mathbf{q}}_A + \theta(t) \cdot \dot{\mathbf{q}}_B$ where θ is a decreasing continuous function of time which takes on values between $[0, 1]$). The obtained continuity was perfect. However there was no guarantee for the corresponding task to be well conditioned or to correspond to a correct motion of the robot.

A. Using a second order differential equation

Soueres et al. proposed a solution to this problem in [14]. They used a second order linear dynamics instead of (14) to take into account two initial conditions $(\mathbf{e}(0), \dot{\mathbf{e}}(0))$:

$$\ddot{\mathbf{e}} + \alpha \cdot \dot{\mathbf{e}} + \beta \cdot \mathbf{e} = 0 \tag{15}$$

where the two parameters α and β are used to control both the robot speed and the length of the transient time reponse. The main drawback is the difficulty in choosing these two parameters to obtain the desired behavior.

B. A simple particular case : non homogeneous first order differential equation

It is choosen to link the task function and its derivative with a non homogeneous first order differential equation. In the general case, the equation is:

$$\dot{\mathbf{e}} = f_2(\mathbf{e}) = -\lambda \cdot \mathbf{e} + \rho(t) \tag{16}$$

where $\rho(t)$ has to be chosen so that it ensures the continuity constraint, and equals to 0 after the transient period:

$$\rho(0) = \dot{\mathbf{e}}(0) + \lambda \cdot \mathbf{e}(0) \text{ and } \lim_{t \to +\infty} \rho(t) = 0 \tag{17}$$

The function used for the experiments is

$$\rho(t) = \left(\dot{\mathbf{e}_\mathbf{A}}(0) + \lambda \cdot \mathbf{e_B}(0)\right) \cdot e^{-\mu \cdot t} \tag{18}$$

where μ is used to set the length of the transient time, and λ to set the decreasing speed of the error. This equation is equivalent to a second order one:

$$\ddot{\mathbf{e}} + (\lambda + \mu) \cdot \dot{\mathbf{e}} + (\lambda \cdot \mu) \cdot \mathbf{e} = 0 \tag{19}$$

Nevertheless, unlike (α, β), this couple of parameters (λ, μ) is properly decoupled. In particular, the end of the transient time is only set by μ. Indeed, the transient period ends when f_1 (see (13)) and f_2 (see (16)) are numerically equivalent, that is to say when $\rho(t)$ is insignificant compared to $\mathbf{e}(t)$, i.e.

$$\delta(t) = \frac{f_1(t) - f_2(t)}{||f_1(t)||} = \frac{\rho(0)}{\lambda} \cdot e^{-\mu \cdot t} = 0 \tag{20}$$

The term δ is exponentially decreasing, with a speed set by μ. The task function $\mathbf{e}(t)$ is equivalent to a decreasing exponential function set by λ. It is simply necessary to choose μ bigger than λ to ensure a short transient time reponse, in comparison with the decreasing time of the task error. The bigger the value μ, the shorter the transient time, but the stronger the acceleration. Experimentally $\mu = 10 \cdot \lambda$ is chosen.

IV. Experiments and results

This section presents the experimental results obtained with a six DOF eye-in-hand robot which is to be positionned with respect to a square object. The initial image is given in Fig. 2.a. The desired image corresponds to the square centered in the image, at a depth of 0.8m. The camera displacement we have considered is very large ($t_x = -10mm$, $t_y = -777mm$, $t_z = 175mm$, $(u\theta)_x = -37dg$, $(u\theta)_y = -7dg$, $(u\theta)_z = -157dg$).

The results obtained for a basic servoing task using the coordinates of four points as visual features are quickly presented. The results of the experiment using our method

are then provided in part IV-B. Three elementary tasks have been chosen, using the centre of gravity, the angle of one diagonal and the second order moments respectively. A last task, using the four points, completes the task.

The task functions $\mathbf{e}$ used in the remainder of the text are computed from the visual features [4]:

$$\mathbf{e} = \mathbf{C}(\mathbf{s} - \mathbf{s}^*) \tag{21}$$

where $\mathbf{s}$ is the current value of the visual features, $\mathbf{s}^*$ their desired value and $\mathbf{C}$ is combination matrix. The interaction matrix $\mathbf{L_s}$ related to $\mathbf{s}$ is defined so that $\dot{\mathbf{s}} = \mathbf{L_s}\mathbf{v}$, where $\mathbf{v}$ is the kinematic camera screw, which is considered as input of the low level robot controller (i.e. $\dot{\mathbf{q}} = \mathbf{v}$). $\mathbf{C}$ has to be chosen so that $\mathbf{CL_s}$ be full rank [13]. From (21), it is clear that the interaction matrix $\mathbf{L_s}$ and the task jacobian $\mathbf{L}$ are linked by the relation:

$$\mathbf{L} = \mathbf{CL_s} \tag{22}$$

Since $\mathbf{CL_s}$ is of full rank, (22) means that the kernel of the two matrices $\mathbf{L}$ and $\mathbf{L_s}$ are equal, and that they can be used indiscriminately to compute the projection operators in (12). In practice, the better choice for $\mathbf{C}$ is $\widehat{\mathbf{L}_\mathbf{s}^+}$ [4].

A. Using a 4-points task

The task based on 4-Points is presented here by way of a comparison with next results. The features used are the coordinates of the four points in the image. The pose is not computed at each iteration of the control law. Thus, the true values of Z, the features depths, are not known. An approximation of the interaction matrix is used instead:

$$\widehat{\mathbf{L}} = \mathbf{L}(Z^*, \mathbf{s}) \tag{23}$$

Due to the high rotation around the X- and Z-axis, the servo does not converge. The features are in fact quickly lost, which means obviously that the desired position cannot be reached (see Fig. 2.b).

B. Four elementary tasks to constraint the six DOF

The four elementary tasks that have been chosen for controlling the robot motion are presented here. As explained in the previous parts, there is no need to choose them independent, thanks to the redundancy formalism. Nevertheless, in order to have a better and easier control over the robot trajectory, decoupled tasks are chosen. Choosing a task consists of selecting specific visual features, and computing the associated interaction matrix. Features and interaction matrices are given for each elementary selected task.

At each iteration, let $P_i = (x_i, y_i)$ be the position of the four points in the image ($i \in [1 \dots 4]$). Let $p_i = (X_i, Y_i, Z_i)$ their position in the 3-D space.

The first task $\mathbf{e_{grav}}$ is based on the position of the centre of gravity. The associated features are easy to compute:

$$\begin{pmatrix} x_G \\ y_G \end{pmatrix} = \begin{pmatrix} \frac{1}{4} \cdot \sum_{i=1}^{4} x_i \\ \frac{1}{4} \cdot \sum_{i=1}^{4} y_i \end{pmatrix} \tag{24}$$

Since the projective projection does not preserve the barycenter, this point G in the image does not correspond to any physical point. Nevertheless this approximation is chosen to be made and G is considered as the centre of gravity of our 3D-object. The approximate interaction matrix is thus obtained from [4]:

$$\widehat{\mathbf{L}}_\mathbf{G} = \begin{pmatrix} -\frac{1}{Z_G} & 0 & \frac{x_G}{Z_G} & x_G y_G & -1 - x_G^2 & y_G \\ 0 & \frac{1}{Z_G} & \frac{y_G}{Z_G} & 1 + y_G^2 & -x_G y_G & -x_G \end{pmatrix} \tag{25}$$

After completion of this task, the object will be centered in the image, which is really desirable for two reasons. First of all, the object is in the middle of the camera field of view, as far as possible from the border of the image. And, since $\mathbf{e_{grav}}$ has priority over the other tasks, it is thus highly unlikely that any point be lost during the execution. In the second place, the problem has been well linearized by centering the object. Indeed, by writing (1), the servo is considered as a linear problem. A good estimator of the validity of this approximation is the distance between the interaction matrices at current and desired positions. As shown in Fig. 3, the distances for the interaction matrices of the four tasks chosen are almost 0 at the end of the centering task. The weakness of the next tasks will be thus rather better.

The second task $\mathbf{e}_\alpha$ rotates the camera around the optical axis, so that the object will be correctly oriented in the image. The feature that is used is the angle α of a segment in the image (we used the diagonal of the 4-points target). The approximated interaction matrix is [4]

$$\widehat{\mathbf{L}}_\alpha = \begin{pmatrix} 0 \\ 0 \\ 0 \\ X_c \sin^2\alpha + Y_c \cos\alpha \sin\alpha \\ X_c \cos\alpha \sin\alpha - Y_c \cos^2\alpha \\ -1 \end{pmatrix}^T \tag{26}$$

where l is the length of the segment, and $P_c = (x_c, y_c)$ its center.

After completing the first two tasks, the rectangle is centered in the image, and properly oriented around the optical axis. In this position, the rest of the execution is easier, and could be realized using the basic 4-points task with a very high probability of success. As shown in Fig. 3, the distance between current and desired interaction matrices of task 4 have again decreased. However, it is preferable to scale the rectangle properly before.

The third task $\mathbf{e_Z}$ uses the secondary centered moments to control the range between the robot and the target. The most intuitive solution is to consider the quadrilateral

area, i.e. the first moment of the continuous object. The area can also be computed geometrically from the discrete rectangle. Since the considered object is discrete, we have used discrete centered moments of second order. The moments are computed using [15]. The centered moment $\mu_{i,j}$ of a set of N points is defined by

$$\mu_{i,j} = \sum_{k=1}^{N} (x_k - x_G) \cdot (y_k - y_G) \tag{27}$$

where (x_G, y_G) is the centre of gravity of the set. The feature a_n considered is computed from the second order moments μ_{20} and μ_{02}:

$$a_n = Z^* \cdot \sqrt{\frac{a^*}{a}} \tag{28}$$

where $a = \mu_{20} + \mu_{02}$. The associated interaction matrix is also given in [15]. When the object is parallel to the image plane, it has the following form:

$$\widehat{\mathbf{L}}_{\mathbf{Z}} = \begin{pmatrix} 0 & 0 & -1 & \epsilon_1 & \epsilon_2 \end{pmatrix} \tag{29}$$

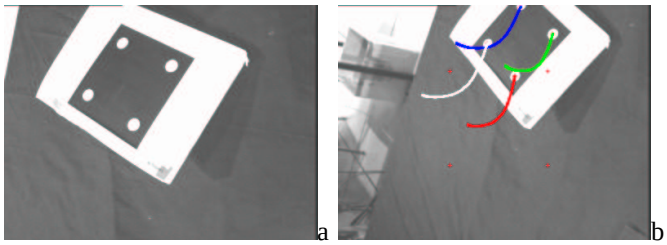
$$\text{where} \quad \begin{cases} \epsilon_1 = y_G + \frac{y_G \cdot \mu_{02} + x_G \cdot \mu_{11}}{a} \\ \epsilon_2 = x_G + \frac{x_G \cdot \mu_{20} + y_G \cdot \mu_{11}}{a} \end{cases}$$

The experiments have shown that one of the four points may be lost during this particular task, even if the object remains centered. A first solution to this problem is to stop the motion on the optical axis at the middle of the distance, i.e. to servo on the desired value $a^*/2$ rather than a^*. In this position, there is a lower probability of loosing the visual features during the final task. However, this would only patch-up the problem. According to us, the good solution is to use the DOF remaining in order to keep the points in the field of view by adding a supplementary task based on a cost function [9]. More attention will be paid to this avenue in the near future.

The last task is the one used in Section IV-A. The eight features are the vertex coordinates. The interaction matrix is easily computed from (25). When this task is added to the stack, the object is centered in the camera field of view, properly oriented and at the desired distance. The two remaining motions are combinations of translation along and rotation around X- and Y-axis. The last selected task is thus not optimal. It shows, nevertheless, that the redundancy formalism is powerful, even without properly decoupled elementary tasks.

C. Results

This section comments the results obtained with our method on the example presented in Section IV-A. As in this section, the depth Z is not known. The approximation $\hat{Z} = Z^*$ is used instead. The visual interaction matrix $\mathbf{L_i}$ and the projection operator are thus approximated too. It has been noticed experimentally that this approximation introduces tracking errors. For a better control, a specific



a b

Fig. 2. Experiment with the 4-Points task. (a) Initial image. (b) Features trajectories in the image. The features leave the camera field.

gain λ_i is thus computed for each elementary task $\mathbf{e_i}$. This gain is low when the corresponding error is high, and is high when the task is nearly completed. The gain applied to the last task added is much lower than the ones applied on the tasks already completed. The lags are thus strongly reduced. The control law is finally:

$$\dot{\mathbf{q}}(t) = -\sum_{i=1}^{n} \lambda_i(\mathbf{e_i}) \cdot \mathbf{P_{1 \ldots i-1}} \cdot \widehat{\mathbf{L}}^{+} \cdot (\mathbf{s_i} - \mathbf{s_i^*}) + \rho(t) \tag{30}$$

where n is the current size of the stack. The experimental results are presented in Figures 4 and 5. The servo is completed without difficulty. The errors decrease exponentially as specified. When a task error is close enough to 0, the next one is added. The tasks already completed stay at 0 during the remainder of the convergence. The first three tasks are well decoupled. Each of them correspond to a favorite DOF. Initialy using the X and Y translations as well as the pan-tilt, the object is centered in the middle of the image. A long rotation around Z-axis is then realized to orient properly the square. The depth is then adjusted using the Z-translation. Each motion corresponds to a recognizable part of the trajectories in the image (Fig. 5). The behavior during the last task is not as good as in the first three. One can notice that the curves are a little bit more noisy. This is due to the approximations in the projectors. A small disturbance appears on the completed tasks, which is compensated by an additional motion in the opposite way at the next step of the servo. This noise can be off set by using a lower gain on the last task. However the convergence is then slower. One can finally notice the continuity of the velocities. Using the classical control law (13), the velocities would have been proportional to the error, meaning that they would be highly discontinuous. Using (16), the velocities increase continuously after each switch.

V. Conclusion

In this paper, a new approach to control a robot using visual servoing has been presented. Rather than choosing a specific trajectory among all the possible trajectories from the beginning, the robot is left as free as possible when it is far from the desired position. Additional constraints are added only when it approaches the goal. A way to implement this general idea has been proposed, using

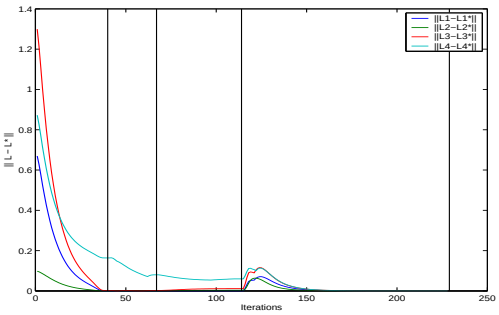


Fig. 3. Differences between visual jacobians at current and desired position. After having completed task 1, the four matrices are approximately equal to their value at desired position. Task 2 makes $\mathbf{L_4}$ decreases once again.

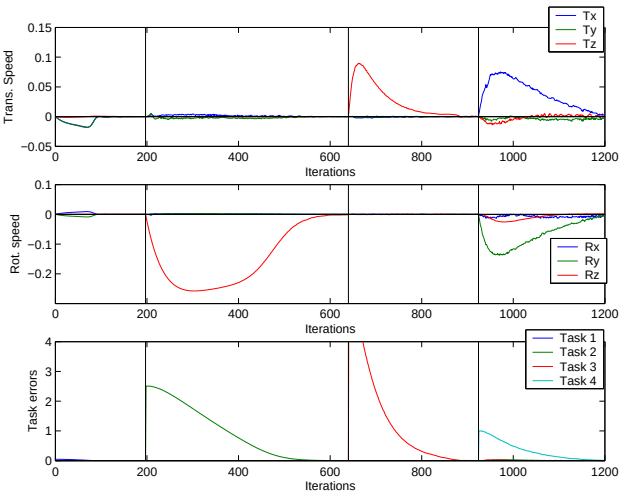


Fig. 4. Second experiment using task sequencing. (a) Translational and (b) rotational velocities (cm/s and dg/s). (c) Tasks errors decreasing.

redundancy formalism and a stack of tasks. It guarantees that, at the end of the execution, the robot has reached the expected position. Additional work has been set up to guarantee the control law continuity with respect to time. Experiments presented in the last section have shown the interest of the approach. Using very simple features and tasks, very good trajectories have been obtained even in the case of a difficult initial position.

Further work is necessary to explore in depth this new approach. In particular, far from the goal, DOF are available but not used yet. It is nevertheless one of the aims of the proposed approach in order to take into account other

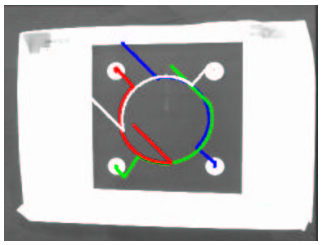


Fig. 5. Final image and vertexes trajectories in the image space.

constraints such as joints limit avoidance. Furthermore, a powerful use of the task sequencing method could be to move around elementary tasks within the stack during a short period of time in order to gain DOF that can be used to avoid obstacles, joint limits, etc. Another future perspective of this research is the automation of the elementary tasks choice. It would be useful for the robot to determine automatically alone when a specific task should be added.

ACKNOWLEDGMENT

The authors thank the partners of the French ROBEA project Egocentre, and especially Philippe Soueres. We thank also Anne-Sophie Tranchant and Omar Tahri for substantial help during the realization of this work.

VI. REFERENCES

[1] F. Chaumette. Potential problems of stability and convergence in image-based and position-based visual servoing. In D. Kriegman, G . Hager, and A.S. Morse, editors, *The Confluence of Vision and Control*, pages 66–78. LNCIS Series, No 237, Springer-Verlag, 1998.

[2] G. Chesi, K. Hashimoto, D. Prattichizzo, and A. Vicino. A switching control law for keeping features in the field of view in eye-in-hand visual servoing. *IEEE Int. Conf. on Robotics and Automation (ICRA'03)*, 3:3929–3934, September 2003.

[3] N.J. Cowan, J.D. Weingarten, and D.E. Koditschek. Visual servoing via navigation functions. *IEEE Trans. on Robotics and Automation*, 18(4):521– 533, August 2002.

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

[5] N. R. Gans and S. A. Hutchinson. An experimental study of hybrid switched approaches to visual servoing. *IEEE Int. Conf. on Robotics and Automation (ICRA'03)*, 3:3061–3068, September 2003.

[6] G. D. Hager. Human-machine cooperative manipulation with vision-based motion constraints. *Workshop on visual servoing, IROS'02*, October 2002.

[7] 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.

[8] 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.

[9] E. Marchand and G.-D. Hager. Dynamic sensor planning in visual servoing. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS'98)*, volume 3, pages 1988–1993, Lueven, Belgium, May 1998.

[10] Y. Mezouar and F. Chaumette. Path planning for robust image-based control. *IEEE Trans. on Robotics and Automation*, 18(4):534–549, August 2002.

[11] L. Peterson, D. Austin, and D. Kragic. High-level control of a mobile manipulator for door opening. *IEEE Int. Conf. on Robotics and Automation (ICRA'03)*, 3:2333–2338, September 2003.

[12] R. Pissard-Gibolet and P. Rives. Applying visual servoing techniques to control a mobile hand-eye system. *IEEE Int. Conf. on Robotics and Automation (ICRA'96)*, pages 166 – 171, May 1996.

[13] C. Samson, M. Le Borgne, and B. Espiau. *Robot Control: the Task Function Approach*. Clarendon Press, Oxford, United Kingdom, 1991.

[14] P. Soueres, V. Cadenat, and M. Djeddou. Dynamical sequence of multi-sensor based tasks for mobile robots navigation. *7th Symp. on Robot Control (SYROCO'03)*, 2:423– 428, September 2002.

[15] O. Tahri and F. Chaumette. Image moment: generic descriptors for decoupled image-based visual servo. *IEEE Int. Conf. on Robotics and Automation (ICRA'04)*, April 2004.