Visual servoing sequencing able to avoid obstacles.

Nicolas Mansard, François Chaumette

Proceedings of the 2005 IEEE
International Conference on Robotics and Automation
Barcelona, Spain, April 2005

Visual Servoing Sequencing Able to Avoid Obstacles

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 the execution of a task. 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 from the desired position, and to incrementally constrain the global task by adding further tasks as the robot moves closer to the goal. As long as they are sufficient, the remaining DOF are used to avoid undesirable configurations, such as joint limits. Closer from the goal, when not enough DOF remain available for avoidance, an execution controller selects a task to be temporary removed from the applied tasks. The released DOF can then be used for the joint limits avoidance. A complete solution to implement this general idea is proposed. Experiments that prove the validity 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 [10]. 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 [2]. Approaches such as 2-1/2-D[12] or path planning [6], [15] provide solutions that enlarge the region where the system converges. But they each constrain all the available robot DOF from the beginning of the servo. This imposes an unique trajectory to the robot. Particularly, if a new problem, such as an obstacle, is encountered during the task execution, the entire trajectory has to be modified to take it into account. In this paper, we propose to sequence uncomplete tasks until the robot reaches its desired position rather than to use a complete one [16], [20], [17]. Very low constraints are used when the robot is far from the goal, in order to enlarge the trajectories available. Constraints are progressively added as the robot draws near to the required position [13]. When enough DOF are available, they can be used to avoid any undesirable situation [14]. Closer from the required position, when not enough DOF remain available for avoidance, additional DOF are obtained by temporary removing a specific constraint when solving the encountered problem.

A vast number of trajectories are usually available to reach the goal. However, by constraining all DOF from the beginning, the classical control schemes 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 [6], [15]. The idea is to choose an optimal trajectory among all the available ones. This provides a complete solution, which ensures optimality, stability and physical feasibility to the goal when it is reachable. Path planning solves the deficiencies of basic approaches, but by applying even stronger constraints to the trajectory of the robot. This means, however, that this solution is less reactive to changes of the goal, of the environment or of the constraints. Rather than deciding in advance which path should be used to reach the goal, switched systems use a set of subsystems along with a discrete switching control [8], [5]. 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. In [13], we described a solution to stack elementary tasks one on top of the others until all degrees of freedom of the robot are constrained, and the desired position is reached. When the robot is far from its desired position, only few DOF are constrained, and the remaining can be used to vary the trajectory, in order to avoid any problem encountered during the execution. Therefore this method does not give satisfactory results when the DOF needed to avoid the problem are already constrained. This is really problematic in the neighborhood of the desired position, when almost all elementary tasks are already stacked.

The key idea of this work is to separate a complete servoing task into several elementary tasks, that are added to a stack when the robot comes closer from the goal. 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. When these additional constraints can not be ensured because of the lack of appropriate DOF, an elementary task already completed can temporary be released.

In this paper, the structure of the stack of tasks used to sequence the elementary tasks is first defined. A control law is computed that maintains the 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 [18]. Contrary to the solution proposed in [13], we rather use the least square resolution method to extend the redundancy formalism to several tasks [19]. The stack also guarantees the continuity of the control law, whatever the manipulation on the stack. We then describe a first general solution to use the remaining DOF for additional constraints such as joint limits avoidance. This method is based on

the Gradient Projection Method [14], [11]. This method gives satisfactory results when the robot is highly under constrained (i.e. far from the goal). However the robot behavior is inadequate when the appropriate DOF are already constrained, especially when the robot is almost fully constrained. A method to provide additional DOF when necessary is then proposed. When the additional constraints can not be respected using the Gradient Projection Method, the elementary task that bothers the execution is detected. It is then temporary removed from the stack. When the problem is solved, the removed task can be put back into the stack. Section II presents the structure of the stack. In Section III, the Gradient Projection Method is recalled and applied on the control law obtained in the previous section. A general method to free additional DOF when needed is exposed in Section IV. The experimental results are finally set out in Section V.

II. VISUAL SERVOING USING A STACK OF TASKS

In this section, we expose how to sequence elementary tasks and to maintain the tasks already achieved. The control law is computed from the tasks in the stack, in accordance with two rules:

- any new task added in the stack should not disturb the tasks already in the stack. Particularly, the tasks already completed should remain to zero.
- the control law computed should be continuous, even when a task is added or removed from the stack. The robot is controlled by the articular velocity $\dot{\mathbf{q}}$. A break of continuity means an infinite acceleration during a short period of time, which implies that the control will not be correctly applied.

Section II-A presents the redundancy formalism [18]. It has first been used for visual servoing in [7], and in numerous applications since (e.g. avoiding visual occultations [14], or human-machine cooperation using vision control [9]). 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. Section II-B sets out the way the redundancy formalism is used to stack several elementary tasks. In Section II-C, we briefly recall the method proposed in [13] to ensure the control law continuity, using a non homogeneous first order differential equation.

A. Redundancy formalism for two tasks

Let $\mathbf{q}$ be the articular vector of the robot. Let $\mathbf{e_1}$ and $\mathbf{e_2}$ be two tasks, $\mathbf{J_i} = \frac{\partial \mathbf{e_i}}{\partial \mathbf{q}}$ $(i = 1, 2)$ their jacobian, defined by:

$$\dot{\mathbf{e}}_\mathbf{i} = \frac{\partial \mathbf{e_i}}{\partial \mathbf{q}} \dot{\mathbf{q}} = \mathbf{J_i} \dot{\mathbf{q}} \tag{1}$$

Since the robot is controlled using its articular velocity $\dot{\mathbf{q}}$, (1) has to be inversed. The general solution (with $i = 1$) is:

$$\dot{\mathbf{q}} = \mathbf{J_1^+} \dot{\mathbf{e}}_\mathbf{1} + \mathbf{P_1 z} \tag{2}$$

where $\mathbf{P_1}$ is the orthogonal projection operator on the null space of $\mathbf{J_1}$ and $\mathbf{J_1^+}$ is the pseudoinverse (or least-squares inverse) of $\mathbf{J_1}$. $\mathbf{z}$ can be used to apply a secondary command, that will not disturb the task $\mathbf{e_1}$ having priority. Here, $\mathbf{z}$ will be used to carry out at best the task $\mathbf{e_2}$. Introducing (2) in (1) (with $i = 2$):

$$\dot{\mathbf{e}}_\mathbf{2} = \mathbf{J_2 J_1^+} \dot{\mathbf{e}}_\mathbf{1} + \mathbf{J_2 P_1 z} \tag{3}$$

By inversing this last equation, and introducing the computed $\mathbf{z}$ in (2), we finally get:

$$\dot{\mathbf{q}} = \mathbf{J_1^+} \dot{\mathbf{e}}_\mathbf{1} + \mathbf{P_1} (\mathbf{J_2 P_1})^+ (\dot{\mathbf{e}}_\mathbf{2} - \mathbf{J_2 J_1^+} \dot{\mathbf{e}}_\mathbf{1}) \tag{4}$$

Since $\mathbf{P_1}$ is Hermitian and idempotent (it is a projection operator), (4) can be written:

$$\dot{\mathbf{q}} = \mathbf{J_1^+} \dot{\mathbf{e}}_\mathbf{1} + \widetilde{\mathbf{J_2}}^+ \widetilde{\dot{\mathbf{e}}_\mathbf{2}} \tag{5}$$

where $\widetilde{\mathbf{J_2}} = \mathbf{J_2 P_1}$ is the limited jacobian of the task $\mathbf{e_2}$, giving the available range for the secondary task to be performed without affecting the first task, and $\widetilde{\dot{\mathbf{e}}_\mathbf{2}} = \dot{\mathbf{e}}_\mathbf{2} - \mathbf{J_2 J_1^+} \dot{\mathbf{e}}_\mathbf{1}$ is the secondary task function, without the part $\mathbf{J_2 J_1^+} \dot{\mathbf{e}}_\mathbf{1}$ of the job already accomplished by the first task. A very good intuitive explanation of this equation is given in [1].

B. Extending redundancy formalism for several tasks

Let $(\mathbf{e_1}, \mathbf{J_1})$ ... $(\mathbf{e_n}, \mathbf{J_n})$ be n tasks. We want to extend (5) to these n tasks. Task $\mathbf{e_i}$ should not disturb task $\mathbf{e_j}$ if $i > j$. A recursive extension of (5) is proposed in [19]:

$$\dot{\mathbf{q}}_\mathbf{i} = \dot{\mathbf{q}}_\mathbf{i-1} + (\mathbf{J_i P_{i-1}^A})^+ (\dot{\mathbf{e}}_\mathbf{i} - \mathbf{J_i} \dot{\mathbf{q}}_\mathbf{i-1}) \tag{6}$$

where $\mathbf{P_i^A}$ is the projector onto the null-space of the augmented Jacobian $\mathbf{J_i^A} = (\mathbf{J_1}, \dots \mathbf{J_i})$. The recursion is initialized by $\dot{\mathbf{q}}_\mathbf{0} = 0$. The robot velocity is $\dot{\mathbf{q}} = \dot{\mathbf{q}}_\mathbf{n}$.

Using directly this recursive equation, a projector has to be computed on each step of the computation. A recursive formula for the computation of the projector is proposed in [1]. We recall this equation here

$$\mathbf{P_i^A} = \mathbf{P_{i-1}^A} - \widetilde{\mathbf{J}}_\mathbf{i}^+ \widetilde{\mathbf{J}}_\mathbf{i} \tag{7}$$

where $\widetilde{\mathbf{J}}_\mathbf{i} = \mathbf{J_i P_{i-1}^A}$ is the limited jacobian of the task i. The recursion is initialized by $\mathbf{P_0^A} = \mathbf{I}$ (identity matrix).

C. Smooth transition

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

$$\dot{\mathbf{e}} = f_1(\mathbf{e}) = -\lambda \mathbf{e} \tag{8}$$

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

$$\dot{\mathbf{q}} = -\lambda \widehat{\mathbf{J}^+} \mathbf{e} \tag{9}$$

where $\widehat{\mathbf{J}^+}$ is an approximation of the pseudo-inverse of $\mathbf{J}$ and λ is used as a parameter to control the robot speed. The f_1 function in (8) is chosen by the programmer to link $\dot{\mathbf{e}}$ and $\mathbf{e}$. One chooses generally $f_1(\mathbf{e}) = -\lambda \mathbf{e}$ to set an exponential decoupled decreasing of the error.

The problem of continuity when changing the task $\mathbf{e}$ 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 $t = 0$. At this time, the control law switches to a second task $\mathbf{e_B}$. Since $\mathbf{e}$ and $\dot{\mathbf{q}}$ are linearly linked, no continuity guarantee can be ensured on $\dot{\mathbf{q}}$, at time t=0.

In [13], we have proposed to use a non homogeneous first order differential equation to ensure the continuity

and to properly decouple the tuning parameters. The differential equation is

$$\dot{\mathbf{e}} = f_2(\mathbf{e}) = -\lambda \mathbf{e} + \rho(t) \tag{10}$$

where the non homogeneous part $\rho(t)$ is

$$\rho(t) = e^{-\mu \cdot t} \cdot \left(\dot{\mathbf{e_A}}(0) + \lambda \mathbf{e_B}(0) \right) \tag{11}$$

D. Computing the control law

Let $(\mathbf{e_1}, \ldots, \mathbf{e_n})$ a stack of n tasks. The decreasing speeds of each task are chosen separately by using

$$\dot{\mathbf{e}} = \begin{bmatrix} \dot{\mathbf{e_1}} \\ \vdots \\ \dot{\mathbf{e_n}} \end{bmatrix} = - \begin{bmatrix} \lambda_1 & & 0 \\ & \ddots & \\ 0 & & \lambda_n \end{bmatrix} \begin{bmatrix} \mathbf{e_1} \\ \vdots \\ \mathbf{e_n} \end{bmatrix} = -\Lambda \mathbf{e} \tag{12}$$

Equation (4) can be written $\dot{\mathbf{q}} = \mathbf{L_p} \dot{\mathbf{e}}$, where the explicit expression of $\mathbf{L_p}$ is left to the reader. Using (10) and (12), we deduce the complete expression of the control law computed from a stack of task

$$\begin{cases} \dot{\mathbf{q}}_\mathbf{i} = \dot{\mathbf{q}}_{\mathbf{i-1}} + (\mathbf{J_i P^A_{i-1}})^+ (-\lambda_i \mathbf{e_i} - \mathbf{J_i} \dot{\mathbf{q}}_{\mathbf{i-1}}) \\ \dot{\mathbf{q}} = \dot{\mathbf{q}}_\mathbf{n} + e^{-\mu \cdot (t-\tau)} \cdot \left(\dot{\mathbf{e}}(\tau) + \Lambda \mathbf{e}(\tau) \right) \end{cases} \tag{13}$$

where τ is the time of the last modification of the stack.

III. Simple avoidance using the Gradient Projection Method

In the previous part, a control law based on a stack of elementary tasks was built, which drives the robot to realize a complete task, such as positioning. This control law is only based on the decreasing of a task function $\mathbf{e}$ (composed of visual features errors in our case). To integrate the servoing into a complex robotic system, the control law should also make sure that it avoids undesired configurations (such as joint limits, occlusion, obstacles or kinematic singularities). A solution very close from the work presented in Section II is used in [14], based on the task function approach [18]. It is called Gradient Projection Method. A cost function is built that reaches its maximum value at the undesired configurations. This cost function to be minimized is then embedded in a secondary task whose only components that do not affect the primary task are taken into account. This is achieved by projecting the secondary task onto the free space of the primary task.

In the following, we refer to the situations to avoid by the general term obstacles. The experiments presented in Section V have been realized using a joint limits avoidance, but the Gradient Projection Method can be applied to various kinds of obstacles [14], [4]. In this section, we first recall the general method to compute a control law as the gradient of a cost function [7]. The complete control law, using a stack of tasks and additional constraints is then computed. This section will be concluded by the problems encountered using this method, which will introduce the last part of this work.

A. General Gradient Projection Method

In this approach, the robot moves according to a repulsive potential V pushing it away from the obstacles. Let us consider the problem:

$$\min V(\mathbf{q}), \quad \mathbf{q} \in \mathbb{R}^k \tag{14}$$

where k is the number of robot articulations. The classical solution is to move the robot according to the gradient of the potential function, computed in the articular space.

$$\dot{\mathbf{q}} = -\kappa \mathbf{g}(\mathbf{q}) = -\kappa \vec{\nabla}^T_\mathbf{q} V \tag{15}$$

where κ is a positive scalar, used as a gain. The secondary task (15) is then projected onto the null space of the primary task, using (2).

B. Gradient Projection Method with a stack of tasks

We want to use the gradient of the cost function as the last task of the stack. The control law (15) has thus to be projected onto the null space of each task into the stack. Using the notations of (13), the complete control law is finally

$$\dot{\mathbf{q}} = \dot{\mathbf{q}}_\mathbf{n} + e^{-\mu \cdot (t-\tau)} \cdot \left(\dot{\mathbf{e}}(\tau) + \Lambda \mathbf{e}(\tau) \right) - \kappa \mathbf{P^A_n g} \tag{16}$$

Therefore, the avoidance depends on two factors. First of all, the control law depends on the projector $\mathbf{P^A_n}$. When the stack is almost empty, the rank of $\mathbf{P^A_n}$ is high, and the avoidance control law will not be much modified. However when the rank decreases near zero (the stack is almost full), the avoidance control law is highly disturbed, especially if the favorite vector direction of the gradient is not one of the range of $\mathbf{P^A_n}$. Of course, when the stack is full, the projector becomes 0. The gradient is thus not taken into account any more, and nothing is done to avoid obstacles. The second factor is the gain κ, which defines *a priori* the influence of the avoidance in the global control law. The choice of this parameter is very important. Indeed, if κ is too small, the gradient force may be too small to avoid the obstacle. Besides, if κ is too high, some overshoot can occur in the computed velocity. Methods that set this parameter automatically exist (for example [4] for the joint limits avoidance). However it is difficult to generalize to an arbitrary number of additional constraints simultaneously. Moreover, these methods do not provide any solution to the problem due to the rank of $\mathbf{P^A_n}$.

Instead, the gradient projection method is applied to stay away from the obstacles when it is possible. When this method is not sufficient, the solution is to choose which task of the stack produces the part of the control law that leads the robot in the obstacle, and to remove it from the stack. This solution is detailed in the next section.

IV. Using a stack controller

In this section, a stack controller that removes a task from the stack when necessary is proposed. We also explain how this solution solves the problem presented in the above paragraph.

Two actions are possible on the stack : add a task and remove a task. A task is added when the velocity sent to the robot is under a fixed level, that is to say, when the tasks in the stack are all realized at best, according to their priority range. The order of the tasks to be introduced in the stack are *a priori* decided. We will focus on a choice *inline* in future works. On the opposite, a task has to be removed from the stack when the current control law drives the robot in one of the configurations

to be avoided (for example the robot nearly reaches a joint limit). Two criteria have to be built, the first one to decide when a task should be removed, the second to choose which task to remove.

A. When removing a task ?

This criterion consists simply in determining the effect of the current control law by performing a prediction step before sending the velocity to the robot. Let $\mathbf{q}(t)$ be the current articular position of the robot. The predicted position $\widehat{\mathbf{q}}(t+1)$ is given by

$$\widehat{\mathbf{q}}(t+1) = \mathbf{q}(t) + \Delta t \dot{\mathbf{q}} \tag{17}$$

where $\dot{\mathbf{q}}$ is the control law, computed using (16). A task has to be removed from the stack if $V_(\widehat{\mathbf{q}}(t+1))$ is above a fixed threshold.

B. Which task to remove ?

The idea is to detect which task induces the most important conflicts with the current obstacle avoidance gradient. We propose two criteria to be computed for each task. The task to remove is the one corresponding to the maximum (or the minimum, in case of the second criterion) of the optimal values computed. Using both criteria simultaneously gives a more reliable choice. In the following, we propose the criteria, for a task $\mathbf{e_i}$, whose jacobian is $\mathbf{J_i}$, and for an avoidance gradient $\mathbf{g}(\mathbf{q})$.

1) First criterion: The first criterion compares directly the direction of the velocity induced by the task, and the one induced by the avoidance gradient. The task to remove is the one whose velocity direction corresponds to the opposite of the gradient direction. This is done by computing the inner product of the two velocities projected in the same space. The most logical common space seems to be the space of articular velocities. Criterion $\mathcal{C}_\mathbf{1}$ is thus

$$\mathcal{C}_\mathbf{1} = - < \mathbf{J_i^+ e_i} | \mathbf{g} > \tag{18}$$

Another common space can be used, such as the space of the task, using $\mathcal{C}_\mathbf{1b} =< \mathbf{e_i} | \mathbf{J_i g} >$. In this case, the common space depends on each task. The experiments have shown that the behavior using any of these criteria is very similar.

This first criterion depends linearly of the task function $\mathbf{e_i}$. If the task is nearly completed ($\mathbf{e_i}$ is very low), the criterion will be very low. We have experimentally noticed that, using (18), the task controller always removes the last task added. We thus use a normalized criterion

$$\mathcal{C}_\mathbf{1}{}' = \frac{1}{||\mathbf{e_i}||} \mathcal{C}_\mathbf{1} \tag{19}$$

Using this last definition, the choice is only based on the velocity direction, and no more on the velocity norm. Therefore, when the velocity induced by a task is very low, the normalization is equivalent to a division by a nearly zero value. That can produce unstable results. The next criterion solve this problem.

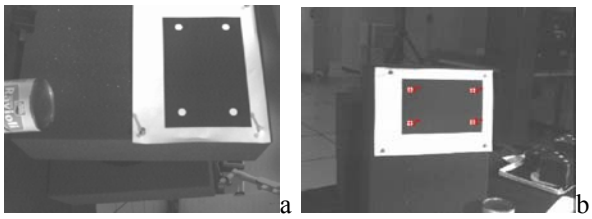

Fig. 1. (a) Initial image (b) Desired image

2) Second criterion: To compute the final control law, the gradient is projected onto the null space of each task. The second criterion computes the contribution of each task to this projection. The idea is to remove the task whose contribution disrupts the most the avoidance.

$$\mathcal{C}_\mathbf{2} = ||\mathbf{P_i g}|| \tag{20}$$

where $\mathbf{P_i} = \mathbf{I} - \mathbf{J_i^+ J_i}$ is the projection operator onto the null space of the task. Since $\mathbf{P_i}$ is a projection operator, for all vector $\mathbf{x}$, $||\mathbf{P_i x}|| \leq ||\mathbf{x}||$. The less the gradient is in the null space of the task, the more it is disturbed, the smaller the value of the criterion. The task to be removed is the one corresponding to the minimum of $\mathcal{C}_\mathbf{2}$.

3) Another way to compute the second criterion: Another idea is to check if the gradient vector is in the null space of the control law due to the task. This subspace is given by (2) : it is the range of $\mathbf{J_i^+}$. Consider a basis $(\mathbf{v_1} \dots \mathbf{v_k})$ of the range of $\mathbf{J_i^+}$ (where k is the rank of $\mathbf{J_i^+}$). The criterion is the norm of the gradient, projected in the range of $\mathbf{J_i^+}$

$$\mathcal{C}_\mathbf{2b} = || \sum_{i=1}^{k} (\mathbf{g} \cdot \mathbf{v_i}) \mathbf{v_i} || \tag{21}$$

In fact, $\mathcal{C}_\mathbf{2}$ checks if the gradient is not in the null space of the jacobian, while $\mathcal{C}_\mathbf{2b}$ checks if the task is in the range of the pseudo inverse of the jacobian, which is equivalent. $\mathcal{C}_\mathbf{2}$ is minimal when $\mathcal{C}_\mathbf{2b}$ is maximal. The experiments confirm that the behaviors using the two criteria are exactly similar. We thus will consider only criterion $\mathcal{C}_\mathbf{2}$ in the following.

V. Experiments and results

We present in this section the experiments realized to validate the proposed methods. A six DOF eye-in-hand robot must reach a position with respect to a visual target (a rectangle composed of four points easily detectable). The robot should moreover avoid its joint limits to complete the positioning task. The experiments have been realized for very large displacements, using as much articular domain as possible. For such a displacement, classical visual servoing fails while the stack using avoidance and the controller (17) enables to reach the desired position. The camera displacement presented in the following is such that the robot has to move from one corner of the articular domain to another. The initial and desired images are presented in Fig. 1. The target in the desired image is centered but the object is not paralell to the image plane. The displacement is ($t_x = -491$mm, $t_y = -120$mm, $t_z = 340$mm, $(u\theta)_x = -70$dg, $(u\theta)_y = -35$dg, $(u\theta)_z = 110$dg).

We first present the elementary visual tasks (section V-A) and the avoidance law (section V-B) used during the

experiments. Four elementary tasks have been chosen, using the object center of gravity in the image, the angle of one diagonal, the second order moments and the third order moments respectively. The robot has moreover to avoid its joint limits. The results of an execution using the stack without avoidance, and with the proposed method are then provided in part V-C.

A. Four elementary tasks to constraint the six DOF

The task functions $\mathbf{e_i}$ used in the remainder of the text are computed from the visual features [7]:

$$\mathbf{e_i} = \mathbf{s_i} - \mathbf{s_i^*} \tag{22}$$

where $\mathbf{s_i}$ is the current value of the visual features for task $\mathbf{e_i}$ and $\mathbf{s_i^*}$ their desired value. The interaction matrix $\mathbf{L_{s_i}}$ related to $\mathbf{s_i}$ is defined so that $\dot{\mathbf{s}}_\mathbf{i} = \mathbf{L_{s_i}}\mathbf{v}$, where $\mathbf{v}$ is the kinematic camera screw. From (22), it is clear that the interaction matrix $\mathbf{L_{s_i}}$ and the task jacobian $\mathbf{J_i}$ are linked by the relation:

$$\mathbf{J_i} = \mathbf{L_{s_i}}\mathbf{M}\mathbf{J_q} \tag{23}$$

where the matrix $\mathbf{J_q}$ denotes the robot jacobian ($\dot{\mathbf{r}} = \mathbf{J_q}\dot{\mathbf{q}}$) and $\mathbf{M}$ is the matrix that relates the variation of the camera velocity $\mathbf{v}$ to the variation of the chosen camera position parametrization $\mathbf{r}$ ($\mathbf{v} = \mathbf{M}\dot{\mathbf{r}}$).

In order to have a better and easier control over the robot trajectory, approximatively decoupled tasks are chosen. As explained in the previous parts, there is no need to choose them perfectly independent, thanks to the redundancy formalism. We have used visual features derived from the image moments. At each iteration, let $P_i = (x_i, y_i)$ be the position of the points in the image. The moment $m_{i,j}$ of the image is defined by

$$m_{i,j} = \sum_{k=1}^{N} x_k^i y_k^j \tag{24}$$

The first task $\mathbf{e_g}$ is based on the position of the center of gravity. The features are based on the first order moments:

$$(x_g, y_g) = (\frac{m_{10}}{m_{00}}, \frac{m_{01}}{m_{00}}) \tag{25}$$

The second task $\mathbf{e_Z}$ uses the centered moments of order 2 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. Since the considered object is discrete, discrete centered moments of second order are used, as proposed in [21]. The third task $\mathbf{e}_\alpha$ mainly rotates the camera around the optical axis, so that the object will be correctly oriented in the image. It uses a combination of the three moments of second order to realize a mainly decoupled motion [3]. The last task $\mathbf{e_R}$ uses third order moments to decouple v_x from ω_y and v_y from ω_x. The reader is invited to refer to [3] for more details.

B. Joint limits avoidance

The experiments of avoidance using the GPM have been realized using a joint limits avoidance. The cost function is defined directly in the articular space. It reaches its maximal value near its joint limits, and the gradient is nearly zero far from the limits.

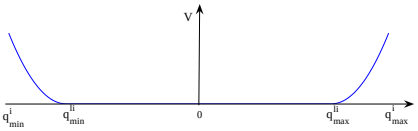

Fig. 2. Potential field of the joint limits avoidance for an articulation

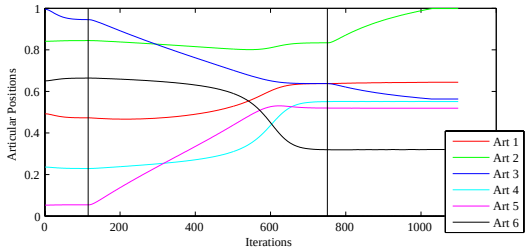

Fig. 3. Articular trajectories using the stack without any avoidance law. The second articulation (z-motion of the camera) reaches its limit during the execution of the third task (task $\mathbf{e_z}$). (each vertical line corresponds to the addition of a new task in the stack.)

The robot lower and upper joint limits for each axis i are denoted $\bar{\mathbf{q}}^i_{\min}$ and $\bar{\mathbf{q}}^i_{\max}$. The robot configuration $\mathbf{q}$ is acceptable if, for all i, $\mathbf{q}_i \in [\bar{\mathbf{q}}^{\ell i}_{\min}, \bar{\mathbf{q}}^{\ell i}_{\max}]$, where $\bar{\mathbf{q}}^{\ell i}_{\min} = \bar{\mathbf{q}}^i_{\min} + \rho\bar{\mathbf{q}}^i$, $\bar{\mathbf{q}}^{\ell i}_{\max} = \bar{\mathbf{q}}^i_{\max} - \rho\bar{\mathbf{q}}^i$, $\bar{\mathbf{q}}^i = \bar{\mathbf{q}}^i_{\max} - \bar{\mathbf{q}}^i_{\min}$ is the length of the domain of the articulation i, and ρ is a tuning parameter, in $[0, 1/2]$ (typically, $\rho = 0.1$). $\bar{\mathbf{q}}^{\ell i}_{\min}$ and $\bar{\mathbf{q}}^{\ell i}_{\max}$ are activation threshold. In the acceptable interval, the avoidance force should be zero. The cost function is thus given by (see Fig. 2) [4]:

$$V(\mathbf{q}) = \frac{1}{2}\sum_{i=1}^{n} \frac{{\delta_i}^2}{\Delta\bar{\mathbf{q}}^i} \tag{26}$$

where

$$\delta_i = \begin{cases} \mathbf{q}_i - \bar{\mathbf{q}}^{\ell i}_{\min}, & \text{if } \mathbf{q}_i < \bar{\mathbf{q}}^{\ell i}_{\min} \\ \mathbf{q}_i - \bar{\mathbf{q}}^{\ell i}_{\max}, & \text{if } \mathbf{q}_i > \bar{\mathbf{q}}^{\ell i}_{\max} \\ 0, & \text{else} \end{cases}$$

C. Results

When the three first tasks are completed, the robot is in one corner of its available domain, near the joint limit corresponding to the motion along the optical axis. Using the four visual tasks presented above into a classical servo fails because of the too large displacement between initial and desired position and the closeness of the joint limits. We present quickly the results obtained with the stack alone (13), then with the stack using the simple avoidance law (16). Without any avoidance, the robot reaches its joint limit quickly, during the third task, while DOF remain available (see Fig.3). This joint limit is avoided using the simple GPM. Instead of going back along the optical-axis, the robot moves along its x and y axes and completes the task. Therefore, the joint limit can not be avoided when the stack is full, because no DOF are available to project the gradient. The execution failed during the execution of the last elementary task $\mathbf{e_R}$ (see Fig. 4).

We present now the results obtained with the complete method (see Fig. 5). During the execution of the last task, the stack controller detects that the joint limit will be reached on axis 2. The controller chooses to remove the task $\mathbf{e_z}$, and the task $\mathbf{e_R}$ is completed using the obtained DOF. The task $\mathbf{e_z}$ is then put back in the stack, and the robot reaches the desired position.

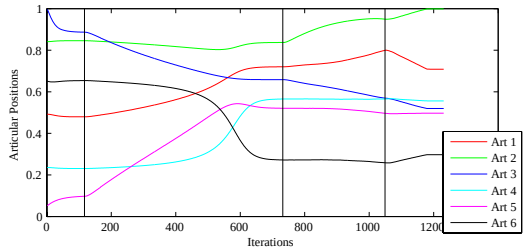

Fig. 4. Articular trajectories using the stack with the GPM but without stack controller. The limit is avoided during the execution of the third task ($\mathbf{e_z}$) by moving along the axis 1. The robot reaches its joint limit during the last task, when no DOF are available any more.

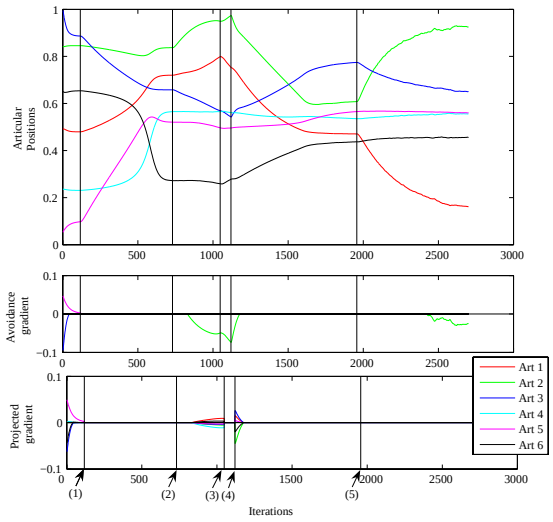

Fig. 5. During the execution of the first task (before event (1)), the avoidance is activated. Since there are many DOF left, the gradient is almost unchanged by the projection operator (gradient and projected gradient are almost equal). The joint limit avoidance is then activated during the task 3 (after event (2)). The obstacle is avoided using the DOF available. Not many DOF remain free, so the gradient is strongly modified by the projection. Therefore, the joint limit is avoided. During the last task, the obstacle can not be avoided (event (3)). No DOF are available, the projected gradient is thus null even if the gradient is not null. The task $\mathbf{e_z}$ is removed from the stack (event (4)). The obstacle is avoided. The task $\mathbf{e_z}$ is finally put back in the stack (event (5)). At the end of the execution, the avoidance is activated once more, but no DOF are available. The projected gradient is null. Therefore, the robot can complete the task despite the closeness of the joint limit. The stack controller does not remove any task from the stack.

VI. Conclusion

In this paper, we proposed a new method to avoid undesirable configurations such as obstacles or joint limits. Far from the desired position, the robot is underconstrained, and the remaining DOF are used for avoidance. When not enough DOF are available, a stack controller has been designed to remove the elementary task from the stack which will free the most appropriate DOF for the avoidance. This control scheme has been validated on a 6-DOF robotic platform. The robot had to reach the desired position from a distant initial position. The classical servoing scheme drive the robot into its joint limits, but using the proposed method, the execution ended properly.

Further work is necessary to validate experimentally this approach when taking into account several additional constraints, such as occlusion, obstacle and joint limits avoidance simultaneously. The stack controller should also be able to choose automatically which task to put in the stack without any *a priori* as supplied here. The robot will thus be able to choose which elementary task to use among a very large amount of possible tasks.

References

[1] P. Baerlocher and R. Boulic. An inverse kinematic architecture enforcing an arbitrary number of strict priority levels. *The Visual Computer*, 6(20), 2004.

[2] 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.

[3] F. Chaumette. Image moments: a general and useful set of features for visual servoing. *IEEE Trans. on Robotics*, 20(4):713–723, August 2004.

[4] F. Chaumette and E. Marchand. A redundancy-based iterative scheme for avoiding joint limits: Application to visual servoing. *IEEE Trans. on Robotics and Automation*, 17(5):719–730, Octobre 2001.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

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

[10] 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.

[11] A. Liegeois. Automatic supervisory control of the configuration and behavior of multibody mechanisms. *IEEE Trans. on Systems, Man and Cybernetics*, 7(12):868–871, December 1977.

[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] N. Mansard and F. Chaumette. Tasks sequencing for visual servoing. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS'04)*, Sendai, Japan, November 2004.

[14] 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, Leuven, Belgium, May 1998.

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

[16] 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.

[17] 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.

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

[19] B. Siciliano and J-J. Slotine. A general framework for managing multiple tasks in highly redundant robotic systems. In *ICAR'91*, pages 1211 – 1216, 1991.

[20] 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.

[21] 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.