IEEE International Conference on Robotics and Automation, ICRA 2000
San Francisco, California, April 2000

A New Redundancy-based Iterative Scheme for Avoiding Joint Limits Application to visual servoing

François Chaumette, Éric Marchand

IRISA - INRIA Rennes
Campus de Beaulieu,
35042 Rennes Cedex, France
Email {Francois.Chaumette, Eric.Marchand}@irisa.fr

Abstract

We propose in this paper new redundancy-based solutions to avoid robot joint limits of a manipulator. We use a control scheme based on the task function approach. We first recall the classical gradient projection approach and we then present a far more efficient method that relies on the iterative computation of motion that does not affect the task achievement and ensures the avoidance problem. We apply this new method in a visual servoing application and we demonstrate on various real experiments the validity of the approach.

1 Introduction

Within a reactive context, planning a robot trajectory is not always possible. If the control law computes a motion that exceeds the robot joint limits, the specified task will not be achieved. Control laws taking into account the region of space located in the vicinity of these joint limits have thus to be considered.

In order to avoid joint limits, Chang and Dubey [1] have proposed a method based on a weighted least norm solution for a redundant robot. This method does not try to maximize the distance of the joints from their limits but it dampens any motion in their direction. Thus, it avoids unnecessary self-motion and oscillations. Another approach has been used by Nelson and Khosla [7] and applied to visual servoing. It consists in minimizing an objective function which realizes a compromise between the main task and the avoidance of joint limits. During the execution of the task, the manipulator moves away from its joint limits and singularities. However, such motions can produce important perturbations in the visual servoing since they are generally not compatible with the specified task. Another approach known as Gradient Projection Method (e.g., [5, 8]) uses robot redundancy and has been widely used to solve joint limits problems. It relies on the evaluation of a cost function seen as a performance criterion function of the joints position. The gradient of this function, projected onto the null space of the main task Jacobian, is used to produce the motion necessary to minimize as far as possible the specified cost function. The main advantage of this method wrt. [7, 1] is that, thanks to the choice of adequate projection operator, the joint limits avoidance process has **no** effect on the main task: avoidance is performed under the constraint that the main task is realized.

In this paper we first recall how the gradient projection approach can be used to avoid joint limits [8, 6]. Unfortunately, it appears that the success of this method relies on a parameter (the amplitude of the secondary task wrt. the main task) that has to be precisely tuned in order to ensure the joint avoidance process. We show that, if badly chosen, the task may fail. We therefore propose an original and far more efficient solution to the joint limits avoidance problem. It consists in generating automatically camera motions compatible with the main task by iteratively solving a system of linear equations. The advantage of this method is that it ensures to stop any motion that moves the robot in the neighborhood of its joint limits.

To validate our approach, we apply the proposed method to a visual servoing problem. Visual servoing [4, 3, 2] is a closed loop reacting to image data. As in the general case, if the control law computes a motion that exceeds a joint limit, visual servoing fails. This specific problem has been already considered in the literature [7, 6]. In a previous paper [6], we considered an extension of the Gradient Projection Method. In this paper, we apply the proposed framework to a vision-based positioning task.

The next section of this paper recalls the approaches proposed in [6] to avoid joint limits. In Section 3 we present the original iterative method. In Section 4 we quickly present the visual servoing framework and we give, in Section 5, real experimental results dealing with positioning tasks. These results have been obtained using an eye-in-hand system composed of a camera mounted on the end-effector of a six d.o.f. robot.

2 Avoiding joint limits using task function approach

A robotic task can be seen as the regulation to zero of a task function [8] defined by:

$$\mathbf{e} = \mathbf{J}_1^+ \mathbf{e_1} + \beta(\mathbf{I_n} - \mathbf{J}_1^+ \mathbf{J_1})\mathbf{e_2} \quad (1)$$

where

- $\mathbf{e_1}$ is the main task to be achieved that induces m independent constraints on the n robot joints (with $m < n$).

- $\mathbf{e_2}$ is a secondary task.
- $\mathbf{J_1^+}$ and $\mathbf{I_n - J_1^+ J_1}$ are two projection operators which guarantee that the camera motion due to the secondary task is compatible with the constraints involved by $\mathbf{e_1}$. $\mathbf{J_1} = \frac{\partial \mathbf{e_1}}{\partial \mathbf{q}}$ is the $m \times n$ full rank Jacobian matrix of task $\mathbf{e_1}$. Each column of $\mathbf{I_n - J_1^+ J_1}$ belongs to Ker $\mathbf{J_1}$, which means that the realization of the secondary task will have no effect on the main task ($\mathbf{J_1}(\mathbf{I_n - J_1^+ J_1})\mathbf{e_2} = 0, \ \forall \mathbf{e_2}$). However, if errors are introduced in $\mathbf{J_1}$, $\mathbf{I - J_1^+ J_1}$ no more exactly belongs to Ker $\mathbf{J_1}$, which will induce perturbations on $\mathbf{e_1}$ due to the secondary task. Let us finally note that, if $\mathbf{e_1}$ constrains all the n degrees of freedom of the manipulator (i.e., $m = n$) , we have $\mathbf{I_n - J_1^+ J_1} = 0$. It is thus impossible in that case to consider any secondary task.
- β is a scalar which sets the amplitude of the control law due to the secondary task. Tuning this scalar has proved to be a non trivial issue. We will see latter on how to consider efficiently this problem.

To make $\mathbf{e}$ decreases exponentially and then behaves like a first order decoupled system, we get:

$$\dot{\mathbf{q}} = -\lambda \mathbf{e} - \beta(\mathbf{I_n - J_1^+ J_1})\frac{\partial \mathbf{e_2}}{\partial t} \tag{2}$$

where:

- $\dot{\mathbf{q}}$ is the joint velocity given as input to the robot controller;
- λ is the proportional coefficient involved in the exponential decrease of $\mathbf{e}$;

The most classical way to solve the joint limits avoidance problem is to define the secondary task as the gradient of a cost function h_s ($\mathbf{e_2} = \frac{\partial h_s}{\partial \mathbf{q}}$). This cost function must reach its maximal value near a joint limits and its gradient must be equal to zero when the cost function reaches its minimal value [8]. Several cost functions h_s which reflect this desired behavior have been presented in [8, 1, 6]. We briefly recall the most efficient cost function proposed in [6].

Activation thresholds of the secondary task are defined by $\tilde{\mathbf{q}}_{i_{min}}$ and $\tilde{\mathbf{q}}_{i_{max}}$ such that:

$$\begin{array}{l} \tilde{\mathbf{q}}_{i_{min}} = \mathbf{q}_{i_{min}} + \rho\,(\mathbf{q}_{i_{max}} - \mathbf{q}_{i_{min}}) \\ \tilde{\mathbf{q}}_{i_{max}} = \mathbf{q}_{i_{max}} - \rho\,(\mathbf{q}_{i_{max}} - \mathbf{q}_{i_{min}}) \end{array} \tag{3}$$

where $0 < \rho < 1/2$ (typically, $\rho = 0.1$). The cost function is thus given by (see Figure 1):

$$h_s = \frac{1}{2}\sum_{i=1}^{n} \frac{s_i^2}{\mathbf{q}_{i_{max}} - \mathbf{q}_{i_{min}}} \tag{4}$$

$$\text{where} \quad s_i = \begin{cases} \mathbf{q}_i - \tilde{\mathbf{q}}_{i_{max}} & \text{if } \mathbf{q}_i > \tilde{\mathbf{q}}_{i_{max}} \\ \mathbf{q}_i - \tilde{\mathbf{q}}_{i_{min}} & \text{if } \mathbf{q}_i < \tilde{\mathbf{q}}_{i_{min}} \\ 0 & \text{else} \end{cases} \tag{5}$$

and components of $\mathbf{e_2}$ and $\frac{\partial \mathbf{e_2}}{\partial t}$ take the form:

$$\mathbf{e_{2}}_i = \begin{cases} \frac{(\mathbf{q}_i - \tilde{\mathbf{q}}_{i_{max}})}{(\mathbf{q}_{i_{max}} - \mathbf{q}_{i_{min}})} & \text{if } \mathbf{q}_i > \tilde{\mathbf{q}}_{i_{max}} \\ \frac{(\mathbf{q}_i - \tilde{\mathbf{q}}_{i_{min}})}{(\mathbf{q}_{i_{max}} - \mathbf{q}_{i_{min}})} & \text{if } \mathbf{q}_i < \tilde{\mathbf{q}}_{i_{min}} \\ 0 & \text{else} \end{cases} \quad , \quad \frac{\partial \mathbf{e_2}_i}{\partial t} = 0 \tag{6}$$

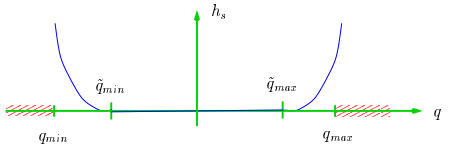

Figure 1: Evolution of the cost function wrt. joint position

This cost function is similar to the Tsai's manipulability measure used in [7]. It is however more simple since it directly sets the activation thresholds with ρ. Let us finally note that, in all cases, $\mathbf{e_2}$ and $\frac{\partial \mathbf{e_2}}{\partial t}$ are continuous, which will ensure a continuous control law.

The parameter β (see equation (1)) that sets the amplitude of the control law due to the secondary task is very important. Indeed, as pointed out in [1], if β is too small, the change in the configuration will occur when $(\mathbf{I - W^+ W})\mathbf{e_2}$ will become large wrt. the primary task. It may be too late and may produce some overshoot in the effector velocity. If β is too large, it will result in some oscillations. Therefore β is usually set based on trial and errors. We now propose a simple new solution to this important problem.

Tuning the influence of the secondary task The simplest solution to this problem is to compute automatically the minimum value of β to stop any motion on the axis the closest from its joint limits (if it is in the critical area). We first determine the component of the primary task $\mathbf{e_1}$ that moves the robot toward its joints limits. This can be done by performing a prediction step. Assuming that the robot is located in $\mathbf{q}(t)$, if we do not consider a secondary task, position $\mathbf{q}(t+1)$ is given by:

$$\mathbf{q}(t+1) = \mathbf{q}(t) + \dot{\mathbf{q}}\Delta t = \mathbf{q}(t) - \lambda \mathbf{J_1^+ e_1}\Delta t \tag{7}$$

If at least one of the axes is in the critical area, the goal is to choose the component k for which $q_k(t+1)$ is the closest from the joint limits and to compute β in order to stop any motion on this component (i.e., $\mathbf{q}_k(t+1) - \mathbf{q}_k(t) = 0$). Using (2), the constraint $\Delta \mathbf{q}_k = 0$ is equivalent to $\mathbf{e}_k = 0$ and using (1) leads to compute β as:

$$\beta = -\frac{\mathbf{J_1^+ e_1}_k}{(\mathbf{I_n - J_1^+ J_1})\mathbf{e_2}_k}$$

The considered joint is stopped but it does not move away its joint limits. The cost function $\mathbf{e_2}$ as defined in (4) is in fact useless. Furthermore, it does not ensure that another axis does not move toward its joint limits. We therefore propose in the next section a new redundancy-based approach to cope with these problems.

3 A new approach: Iterative computation of adequate motions

As seen in the previous section, a good solution to achieve the avoidance task is to cut any motion on axes that are in critical area or that moves the robot toward it. Considering that $\mathbf{q}_k$ is one of these axes, we have to compute a velocity $\dot{\mathbf{q}}_k = 0$. In the

previous paragraph, we considered such a condition but the result was to compute the minimum value of β (for all the axes) that ensures this task. It should be more interesting to compute such a gain on each axis. As described below, the proposed approach to achieve this goal relies on the resolution of a linear system. Another drawback of the previous approach is that, thanks to the new computed control law, other axes may enter in the critical area. In the new framework, this can be handled by applying the same algorithm iteratively.

A general task function that uses redundancy can be defined by[1]:

$$\mathbf{e} = \mathbf{J_1^+ e_1} + \sum_{i=1}^{n_0} \mathbf{B}_i \mathbf{E}_{\bullet i} \tag{8}$$

where

- $n_0 = \dim \text{Ker} \mathbf{J_1} = n - m$.
- $\sum_{i=1}^{n_0} \mathbf{B}_i \mathbf{E}_{\bullet i}$ defines motions that try to ensure that the robot will never encounter its joints limits. Within this term:
  - $\mathbf{E}$ is a basis of Ker $\mathbf{J_1}$ of dimension $n \times n_0$ (this insures that these motions have no effect on the main task).
  - $\mathbf{B}$ is a vector of gains that will be automatically computed.

Consider that axis $\mathbf{q}_k$ is in critical situation (i.e., $\mathbf{q}_k \leq \tilde{\mathbf{q}}_{k_{min}}$ or $\tilde{\mathbf{q}}_{k_{max}} \leq \mathbf{q}_k$). We determine vector $\mathbf{B}$ in order to stop the motion on axis k: $\Delta \mathbf{q}_k = 0$ (or $\dot{\mathbf{q}}_k = 0$). From equation (8), for each axis in critical situation we obtain:

$$\sum_{i=1}^{n_0} \mathbf{B}_i \mathbf{E}_{ki} = - \left[\mathbf{J_1^+ e_1} \right]_k \tag{9}$$

If we now consider the p axes in critical situation, we can define from equation (9) a linear system $\mathbf{FB} = \mathbf{S}$ where $\mathbf{F}$ is of dimension $p \times n_0$ while $\mathbf{S}$ and $\mathbf{B}$ are of dimension n_0. We have three possible cases:

- when $p > n_0$, we have more axes in critical situation than redundant axes. Of course in that case, the total efficiency of the method cannot be ensured.
- when $p = n_0$, there is only one solution but the problem can be solved.
- when $p < n_0$, the system features multiple solutions.

In any case, a solution is given by $\mathbf{B}^* = \mathbf{F}^+ \mathbf{S}$ where $\mathbf{F}^+$ is obtained using a Singular Values Decomposition.

Let us consider more deeply the last configuration ($p < n_0$). If vector $\mathbf{B}^*$ is computed as $\mathbf{B}^* = \mathbf{F}^+ \mathbf{S}$, any motion on the p axes in critical situation are stopped. However, with the resulting control law:

$$\dot{\mathbf{q}} = -\lambda \left(\mathbf{J_1^+ e_1} + \sum_{i=1}^{n_0} \mathbf{B}_i^* \mathbf{E}_{\bullet i} \right) \tag{10}$$

[1] If $\mathbf{M}$ is a matrix, we note $\mathbf{M}_{\bullet i}$ its i^{th} column and $\mathbf{M}_{i\bullet}$ its i^{th} row.

other axes may enter in the critical area. This undesired situation can be handled. Indeed when $p < n_0$, the linear system features multiple solutions. $\mathbf{B}^*$ can in fact be chosen as:

$$\mathbf{B}^* = \mathbf{F}^+ \mathbf{S} + \sum_{j=1}^{n_1} \alpha_j (\mathbf{I_{n_0}} - \mathbf{F}^+ \mathbf{F})_{\bullet j} \tag{11}$$

where $\mathbf{I_{n_0}} - \mathbf{F}^+ \mathbf{F}$ is a basis of Ker $\mathbf{F}$ and $n_1 = \dim \text{Ker } \mathbf{F}$. The new motions involved by $\sum_j \alpha_j (\mathbf{I_{n_0}} - \mathbf{F}^+ \mathbf{F})_{\bullet j}$ are built in the kernel of the constraint (i.e., projected onto the null space of the constraint, the resulting have therefore still no effect on the main task). Replacing $\mathbf{B}^*$ by its value defined in (11), we get:

$$\begin{aligned} \dot{\mathbf{q}} \quad = \quad & -\lambda \Big[\mathbf{J_1^+ e_1} \\ & + \sum_{i=1}^{n_0} \left(\mathbf{F}^+ \mathbf{S} + \sum_{j=1}^{n_1} \alpha_j (\mathbf{I_{n_0}} - \mathbf{F}^+ \mathbf{F})_{\bullet j} \right)_i \mathbf{E}_{\bullet i} \Big] \end{aligned} \tag{12}$$

To determine the vector α, like in the previous case, we build a linear system considering that $\dot{\mathbf{q}}_k = 0$ for all the p' axes $\mathbf{q}_k$ that will enter the undesired area according to the prediction. After some rewriting, each line of the system is given by:

$$\begin{aligned} \sum_{j=1}^{n_1} \alpha_j \sum_{i=1}^{n_0} (\mathbf{I_{n_0}} - \mathbf{F}^+ \mathbf{F})_{ij} \mathbf{E}_{ki} \quad = \quad & -(\mathbf{J_1^+ e_1})_k \\ & - \left[\sum_{i=1}^{n_0} (\mathbf{F}^+ \mathbf{S})_i \mathbf{E}_{\bullet i} \right]_k \end{aligned} \tag{13}$$

As in the previous case, there are three possible case regarding the dimension of α. Here again, p new axes may enter in the critical area according to the new control value computed with the appropriate vector $\mathbf{B}^*$ and α. Therefore this last process is repeated iteratively (and can be repeated as long as $p + p' + p'' + \ldots < n_0$).

Remark: Let us note that the resulting control law is not continuous. Even if, in practice, the dynamic of the robot smoothes the velocity, coping with this issue is one of the perspective of this work.

Moving away from the joint limits. The presented framework provides a complete solution to ensure that, if a solution exists, the joints in critical situation will not encounter their limits. It could also be interesting, like in the previous section, to generate a motion that moves the joint away from its limits. This can be simply achieved by introducing a cost function (as proposed in equation (4)) within equation (8) that becomes:

$$\mathbf{e} = \mathbf{J_1^+ e_1} + (\mathbf{I} - \mathbf{J_1^+ J_1}) \mathbf{e_2} + \sum_{i=1}^{n_0} \mathbf{B}_i \mathbf{E}_{\bullet i} \tag{14}$$

The new linear system to be solved is then given by:

$$\sum_{i=1}^{n_0} \mathbf{B}_i \mathbf{E}_{ki} = - \left[\mathbf{J_1^+ e_1} + (\mathbf{I_n} - \mathbf{J_1^+ J_1}) \mathbf{e_2} \right]_k \tag{15}$$

4 Application to visual servoing

We applied the proposed method to an image-based visual servoing problem. Let us denote $\mathbf{P}$ the set of selected visual features used in the visual servoing task. To ensure the convergence of $\mathbf{P}$ to its desired value $\mathbf{P_d}$, we need to know the interaction matrix (or image Jacobian) $\mathbf{L_P^T}$ defined by the classical equation [2] :

$$\dot{\mathbf{P}} = \mathbf{L_P^T T_c} \tag{16}$$

where $\dot{\mathbf{P}}$ is the time variation of $\mathbf{P}$ due to the camera motion $\mathbf{T_c}$.

Control laws in visual servoing are generally expressed in the operational space (i.e., in the camera frame), and then computed in the articular space using the robot inverse Jacobian. However, in order to combine a visual servoing with the avoidance of joint limits, we have to directly express the control law in the articular space. Indeed, manipulator joint limits are defined in this space. This leads to the definition of a new interaction matrix such that:

$$\dot{\mathbf{P}} = \mathbf{H_P}\ \dot{\mathbf{q}} \tag{17}$$

Since we have $\mathbf{T_c} = \mathbf{J(q)}\ \dot{\mathbf{q}}$, where $\mathbf{J(q)}$ is nothing but the robot Jacobian, we simply obtain:

$$\mathbf{H_P} = \mathbf{L_P^T}\ \mathbf{J(q)} \tag{18}$$

If l visual features are selected, the dimension of $\mathbf{H_P}$ is $l \times n$. If the visual features are independent, the rank m of $\mathbf{H_P}$ is equal to l, otherwise $l > m$. The vision-based task $\mathbf{e_1}$ is then defined by:

$$\mathbf{e_1} = \mathbf{C}(\mathbf{P} - \mathbf{P_d}) \tag{19}$$

where $\mathbf{P_d}$ is the desired value of the selected visual features, $\mathbf{P}$ is their current value (measured from the image at each iteration of the control law), and $\mathbf{C}$, called combination matrix, has to be chosen such that $\mathbf{J_1} = \mathbf{CH_P}$ is full rank along the desired trajectory $\mathbf{q_r}(t)$. It can be defined as $\mathbf{C} = \mathbf{W}{\mathbf{H}^+}_{\mathbf{P}|\mathbf{P}=\mathbf{P_d}}$, where $\mathbf{W}$ is a full rank $m \times n$ matrix such that Ker $\mathbf{W}$ = Ker $\mathbf{H_P}$ (see [8][2] for more details). If $\mathbf{H_P}$ is full rank m, we can set $\mathbf{W} = \mathbf{H_P}$, then $\mathbf{C} = \mathbf{I}$ and $\mathbf{J_1} = \mathbf{H_P}$ is a full rank $m \times n$ matrix. If rank m of $\mathbf{H_P}$ is less than l, we have $\mathbf{J_1} = \mathbf{W}\mathbf{H_P^+}\mathbf{H_P}$ which is also a full rank $m \times n$ matrix.

We then can use the framework presented in the previous sections.

5 Experimental results

All the joint limits avoidance approaches presented in this paper have been implemented on an experimental testbed composed of a CCD camera mounted on the end effector of a six degrees of freedom cartesian robot. The implementation of the control law as well as the image processing runs on an Ultra SPARC. Each iteration is done in 100ms.

Positioning task. The specified visual task consists in a gazing task. If $\mathbf{P} = (X, Y)$ describes the position in the image of the projection of the center of gravity of an object, the goal is to observe this object at the center of the image: $\mathbf{P_d} = (0, 0)$. In the presented experiments, the initial robot position is located in the vicinity of three joint limits ($\mathbf{q}_1$, $\mathbf{q}_2$, $\mathbf{q}_3$) while $\mathbf{q}_5$ is located near the threshold $\tilde{\mathbf{q}}_{5_{min}}$. If none strategy is used to avoid joint limits, the visual task fails. On all the plots, joint positions are normalized between [-1;1], where -1 and 1 represent the joint limits.

Gradient Projection Approach. We performed a set of experiments using the cost function defined in Section 3 with various value of the β coefficient. The goal is reached with $\beta > 0.1$. If β is too large (e.g., $\beta = 2$), it results in oscillation and too important motions. If β is too small (typically $\beta = 0.05$), the motion generated by the main task in the direction of the joint limits is not enough compensated by the secondary task. As pointed out in [1], tuning β is therefore performed based on trial and error. This solution is not acceptable.

Fig. 2 depicts results obtained using the approach proposed in Section 2. β is automatically computed. Motion on axis $\mathbf{q}_1$ has been stopped since it is the closest from its joint limit.

Results with the iterative approach. The following results deal with various experiments considering or not the iterative process (i.e., the iterative evaluation of the vector α) and including or not a cost function $\mathbf{e_2}$.

To consider the behavior of our algorithm, we show on Figure 3 the behavior of the robot on the 5th axis. The threshold $\tilde{\mathbf{q}}$ is -0.93. On the first plot *'no iteration'* the α vector is not estimated using the iterative approach and no cost function has been added in (8). The computed motion moves axis $\mathbf{q}_5$ toward its joint limit until it crosses the threshold. Then, since it is in the critical area, $\mathbf{B}^*$ is computed in order to stop any motion on this axis. On the second plot *'no iter with cost function'*, a cost function has been added (as defined in equation (4)). As in the previous case, the robot crosses the threshold which increases the cost function. This results in a motion in the opposite direction. Once in the safe area, the cost function is null and motion is again directed toward the joint limit. This predictable behavior results in oscillations around the threshold. In both cases, the motions produced to avoid other joint limits ($\mathbf{q}_1$ and $\mathbf{q}_2$) have generated an undesired behavior: axis $\mathbf{q}_5$ that was not in the critical area enters this area. Let us note that such a behavior can be also observed when we consider a gradient projection approach.

The iterative method has been built to cope with this problem as can be seen on plot *'iterations* and *'iterations with cost function'*. In that case $\mathbf{B}^*$ is computed in order to avoid that a new axis moves toward the threshold. In fact when predictions considering $\alpha = 0$ show that $\mathbf{q}_5$ will enter in the critical area (just after iteration 55), another solution is proposed. As explained in the previous section, this adequate new solution is proposed by computing iteratively motion in the kernel of the constraints. As can been seen, the motion on $\mathbf{q}_5$ allows the robot not to enter the critical area (plot *'iterations*). The motion is even directed on the opposite direction if a cost function is considered (plot *'iterations with cost function'*).

The full behavior of a similar experiment considering the iterative approach is reported on Figure 4 (without $\mathbf{e_2}$) and 5 (with $\mathbf{e_2}$). Behavior is quite similar except dealing with the motion on the axes $\mathbf{q}_1$ and $\mathbf{q}_3$ that are in the critical area at the beginning of the experiment and that are only stopped in the first case while, in the second case they moves away toward the threshold.

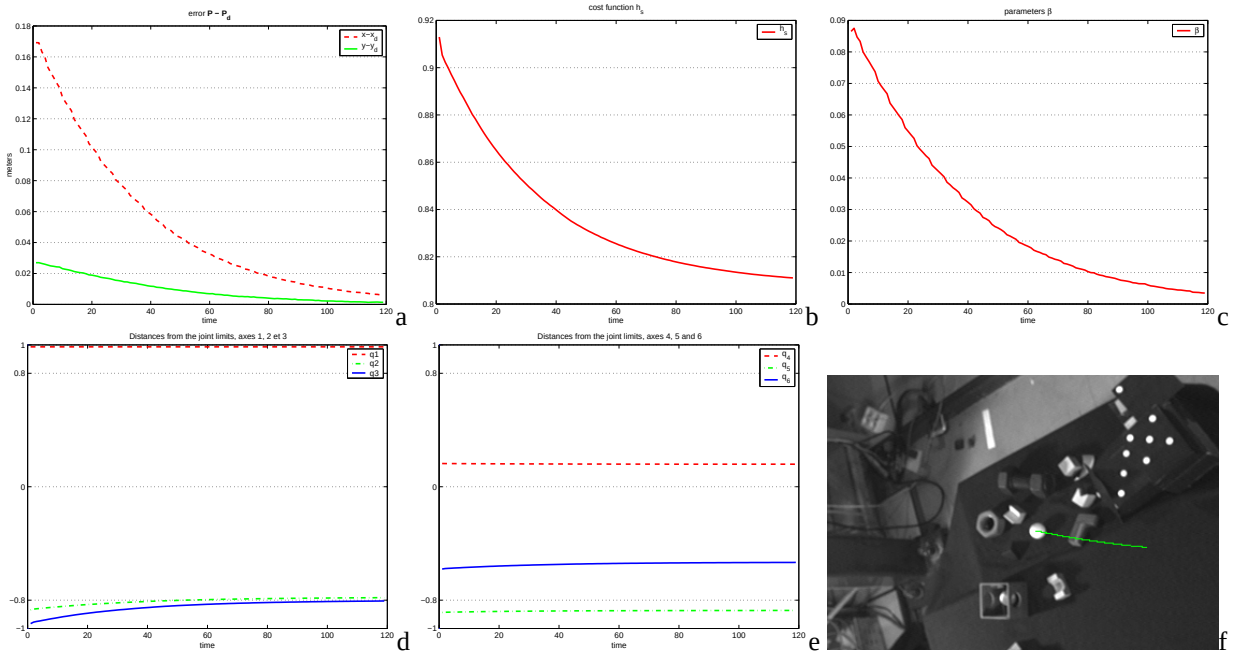

Figure 2: Gradient projection approach with automatic tuning of scalar β (a) errors $\mathbf{P} - \mathbf{P_d}$, (b) cost function h_s, (c) evolution of scalar β, (d) joint position $\mathbf{q}_1$, $\mathbf{q}_2$, $\mathbf{q}_3$, (e) joint position $\mathbf{q}_4$, $\mathbf{q}_5$, $\mathbf{q}_6$, (f) image trajectory

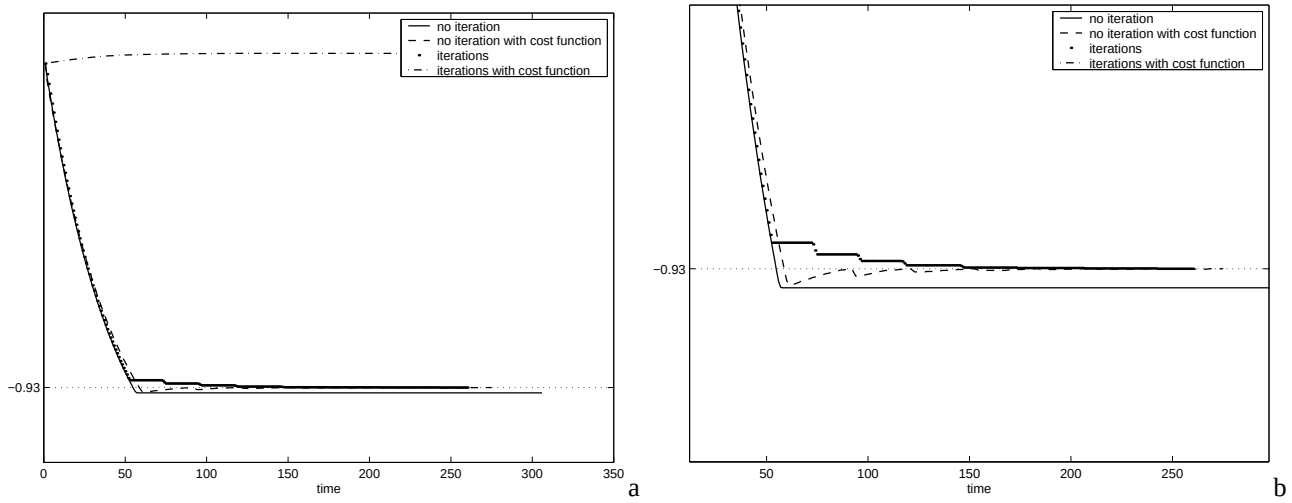

Figure 3: Behavior of the 5th axis of the manipulator considering various methods with the iterative approach ((b) is a close up of (a)).

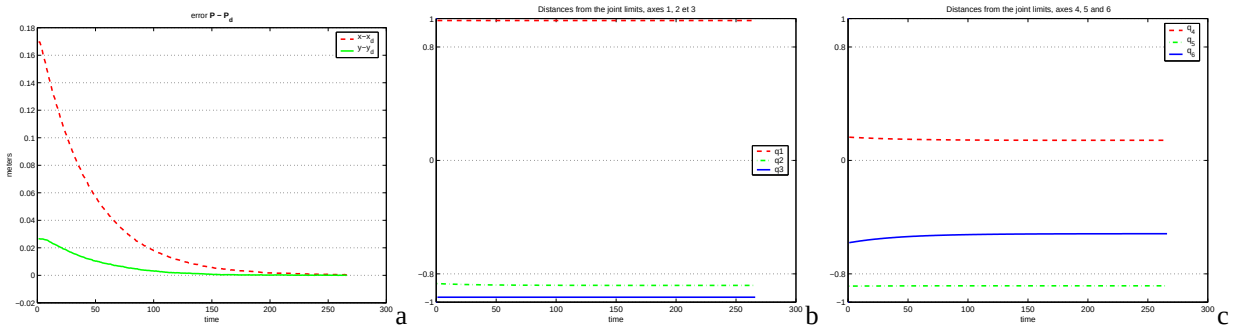

Figure 4: Iterative approach with $\mathbf{e_2} = 0$ in (8). (a) errors $\mathbf{P} - \mathbf{P_d}$, (b) joint position $\mathbf{q}_1$, $\mathbf{q}_2$, $\mathbf{q}_3$, (c) joint position $\mathbf{q}_4$, $\mathbf{q}_5$, $\mathbf{q}_6$

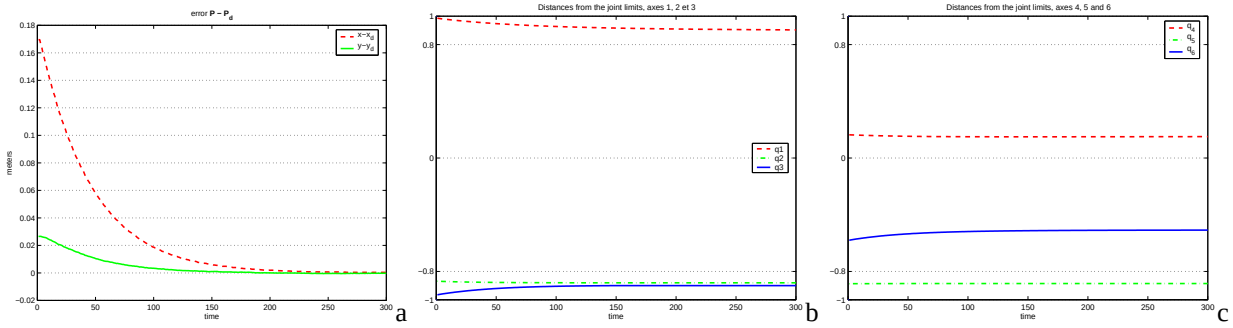

Figure 5: Iterative approach considering a cost function $\mathbf{e_2}$ in (8). (a) errors $\mathbf{P} - \mathbf{P_d}$, (b) cost functions (c) joint position $\mathbf{q}_1$, $\mathbf{q}_2$, $\mathbf{q}_3$, (d) joint position $\mathbf{q}_4$, $\mathbf{q}_5$, $\mathbf{q}_6$

6 Conclusion

We have proposed an original method to avoid the joint limits of a manipulator. It consists in generating automatically camera motions compatible with the main task by iteratively solving a system of linear equations. This new approach is far more efficient than the classical gradient projection method. It avoids unnecessary motions, and unlike gradient projection methods, it guarantees the joint limits avoidance: an axis in critical area will be at least stopped and even moved outside this area. For axes outside critical area, it ensures that they will never enter it (if such a solution does exist). We have demonstrated on real experiments within a visual servoing context the validity of our approach. Let us finally note that this new approach may be used for other problems where gradient projection approach are classically used, such as obstacle avoidance [8].

Acknowledgments. The authors wish to thank Viviane Cadenat from LAAS (Toulouse) for her careful reading and remarks on the paper.

References

[1] T.-F. Chang and R.-V. Dubey. A weighted least-norm solution based scheme for avoiding joints limits for redundant manipulators. *IEEE Trans. on Robotics and Automation*, 11(2):286–292, April 1995.

[2] 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.

[3] K. Hashimoto. *Visual Servoing : Real Time Control of Robot Manipulators Based on Visual Sensory Feedback*. World Scientific Series in Robotics and Automated Systems, Vol 7, World Scientific Press, Singapor, 1993.

[4] 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.

[5] A. Liegeois. Automatic supervisory control of the configuration and behavior of multibody mechanisms. *IEEE Trans. on Syst., Man and Cybern.*, SMC-7(12):245–250, March 1977.

[6] E. Marchand, F. Chaumette, and A. Rizzo. Using the task function approach to avoid robot joint limits and kinematic singularities in visual servoing. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, IROS'96*, volume 3, pages 1083–1090, Osaka, Japan, November 1996.

[7] B. Nelson and P.K. Khosla. Strategies for increasing the tracking region of an eye-in-hand system by singularity and joint limits avoidance. *Int. Journal of Robotics Research*, 14(3):255–269, June 1995.

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