Visual Predictive Control Scheme for a Mobile Robot Navigating in a Cluttered Environment

Adrien Durand-Petiteville, Viviane Cadenat

Visual Predictive Control Scheme for a Mobile Robot Navigating in a Cluttered Environment

Adrien Durand-Petiteville[1] and Viviane Cadenat[2]

***Abstract*— This work focuses on the use of Visual Predictive Control to drive a mobile robot in a cluttered environment. To efficiently and safely achieve the navigation task, one proposes to modify a classical VPC scheme by (i) relaxing some control input boundaries to extend the feasibility set and guarantee the stability, (ii) adding obstacle constraint over the pieces of trajectory, and (iii) modifying the calculated solution to avoid local minima. Simulated results highlight the efficiency of the proposed approach.**

I. Introduction

In this work, it is proposed to use a Visual Predictive Control (VPC) scheme to make a differential driven mobile robot navigate in an environment containing *a priori* unknown static obstacles. VPC [1] is the fusion between Image-Based Visual Servoing (IBVS) [2] and Nonlinear Model Predictive Control (NMPC) [3] [4]. The obtained control scheme thus combines the advantages of IBVS, *i.e.,* reactivity and absence of metric localization [5], with the ones of NMPC, *i.e.,* ability to explicitly deal with constraints such as collision with obstacles or control inputs boundaries. Over the last decade, the interest for VPC-based controllers has grown and several schemes were designed to control different robotic systems: a camera mounted on a robotic arm [6] [7], a flying camera [8] [9], a mobile robot [10] or a fixed-wing aerial vehicle [11]. More advanced schemes were proposed such as in [12] where the robotic arm is represented by a polytopic linear parameter-varying system, in [13] and [14] where image moment visual features are used, or in [15] where the prediction scheme used to control a robotic arm is obtained by integrating the acceleration of the visual features. Despite the variety of approaches, stability related issues are not considered. Indeed, it is well established that NMPC schemes, and therefore VPC, only guarantee a stable closed-loop when considering an infinite prediction horizon [3]. For solutions relying on a finite prediction horizon, which is the case for all the mentioned works, two main classes of approach are identified to guarantee stability. The first one enforces stability by adding a zero terminal equality constraint at the end of the prediction horizon [16], [17]. The second one relies on the quasi-infinite horizon method [18], which consists in adding a terminal penalty term to the cost function and a terminal region constraint. Both are determined off-line such that the modified cost function gives an upper bound on the infinite horizon cost and guarantees a decrease in the cost function. This second class of solution does not seem to be appropriate to the navigation problem. Indeed, the obstacles are detected during the navigation and the constraints related to obstacle avoidance might be updated at every iteration. It is then impossible to determine a terminal region at the beginning of the navigation. Moreover, the quasi-infinite horizon methods requires that there exists a known control law, local to the terminal region, that stabilizes the system and satisfies the constraints. One more time, it is impossible to prove the existence of such a local control law without knowing the constraints in the terminal region.

In this work, it is then proposed to guarantee the stability by adding a zero terminal equality constraint. For such an approach, the size of the feasibility set, set for which there exists a trajectory reaching the goal while dealing with the constraints, depends on both the length of the prediction horizon and the boundaries of the control inputs. Large prediction horizons leading to large computational times, one proposes to enlarge the feasibility region by taking action on the control input boundaries. Thus, in this work one defines two sets of constraints for the control inputs. The first constraints correspond to the actual boundaries of the system and are applied on the first part of the prediction horizon, whereas the second ones are relaxed and applied to the rest of the prediction horizon. Thus, the command applied to the robot, which is in the first part of the prediction horizon, respects the actuators boundaries, while the feasible region is enlarged by the use of relaxed boundaries. However, in order to preserve an efficient and safe navigation system while relaxing the control input boundaries, it is required to deal with two issues. First, the constraints applied to the system, *e.g.,* to avoid collision with obstacles, have to be checked along the predicted trajectories and not only for the predicted states as it usually done. Indeed, due to possible large commands, the predicted trajectory between two states can be sufficiently large to pass through an obstacle without violating the state-centered collision constraints. Next, one has to deal with the suboptimality of the computed solution. Indeed, predictive control schemes of complex systems usually rely on numerical solvers computing a local solution, *i.e.,* the computed trajectory is sub-optimal. It will be shown that the insertion of relaxed constraints increases the solution suboptimality, and leads to possible navigation failures in case the robot reaches a local minima. Thus, in addition to the insertion of relaxed constraints, one presents in this work two other contributions: (i) a constraint to avoid collisions along the robot trajectory, and (ii) a method based on equivalent control vectors [19] to modify the obtained

[1]Adrien Durand-Petiteville is with the Mechanical Engineering Department, Federal University of Pernambuco, Recife, Brazil adrien.durandpetiteville@ufpe.br

[2]Viviane Cadenat is with LAAS-CNRS and Université Paul Sabatier Toulouse III, Toulouse, France cadenat@laas.fr

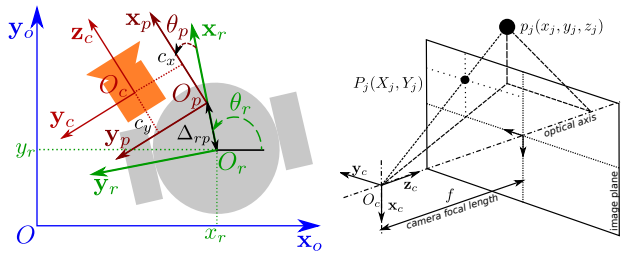
trajectory and reduce the solution suboptimality.

The paper is organized as follows. First, the robot and visual features models as well as the equivalent control vector method are presented. In the next section, the VPC scheme and the set of constraints are introduced. Next, one describes our method to modify the obtained trajectory and improve the solution suboptimality. Finally, simulated results are presented to illustrate the proposed approach.

II. Preliminaries

In this section, one first presents the robot model, then the visual features and their prediction model that will be used in the VPC scheme. Finally, one recalls the equivalent command vector principle [19] used in section IV.

A. System Modeling



(a) Differential robot model

(b) Camera pinhole model

Fig. 1: System model

A pinhole camera embedded on a differential robot equipped with a pan-platform is controlled via a VPC scheme. Let define the world frame $\mathbf{F}_o(O, \mathbf{x}_o, \mathbf{y}_o, \mathbf{z}_o)$, the robot frame $\mathbf{F}_r(O_r, \mathbf{x}_r, \mathbf{y}_r, \mathbf{z}_r)$, the platform frame $\mathbf{F}_p(O_p, \mathbf{x}_p, \mathbf{y}_p, \mathbf{z}_p)$, and $\mathbf{F}_c(O_c, \mathbf{x}_c, \mathbf{y}_c, \mathbf{z}_c)$ the camera frame (see Fig. 1a). Moreover, the camera focal length is denoted by f (see Fig. 1b). Let θ_r be the direction of the robot with respect to $\mathbf{x}_o$, θ_p the direction of the pan-platform with respect to $\mathbf{x}_r$, and Δ_{rp} the distance between O_r and O_p.

The camera being embedded on a differential robot equipped with a pan-platform, it only has three degrees of freedom. Thus, the control input vector is given by $\mathbf{Q} = [\upsilon, \omega_r, \omega_p]^T$, where υ and ω_r are the mobile base linear and angular velocities, and ω_p is the pan-platform angular velocity with respect to $\mathbf{F}_r$. Moreover, with x_c and y_c the coordinates of the point O_c in $\mathbf{F}_o$ and $\theta_c = \theta_r + \theta_p$, one defines the camera state as:

$$\chi_c = [x_c, y_c, \theta_c]^T \tag{1}$$

B. The Visual Features

A VPC scheme relies on a landmark to control the camera. One assumes that this landmark can be characterized by N_v interest points which are extracted by an image processing. An interest point p_j, whose coordinates in the camera frame are given by (x_j, y_j, z_j), is represented by a point P_j whose coordinates are $S_j = (X_j, Y_j)$ in the image plane, with $j \in [1, ..., N_v]$ (see figure 1b). Therefore, the visual data are represented by a $2N_v$ dimensional vector $\mathbf{S} = [X_1, Y_1, ..., X_{N_v}, Y_{N_v}]^T$.

The robot is a sampled system whose inputs evolve at each instant $t = kT_s$, where T_s is the sampling time. Assuming that the inputs $\mathbf{Q}(t_1)$ are constant during the two instants t_1 and $t_2 = t_1 + T_s$, the following prediction model of the visual features between t_1 and t_2 has been obtained in [20]:

$$\begin{cases} X_j(t_2) = & \frac{z_j(t_1) X_j(t_1)}{z_j(t_2)} \\ Y_j(t_2) = & \frac{f}{z_j(t_2)} \Big\{ C_1 \cos(A) - C_2 \sin(A) \\ & + \Delta_{rp} \sin(\theta_p(t_2)) + \frac{\upsilon(t_1)}{\omega_r(t_1)} \cos(\theta_p(t_2)) - c_y \Big\} \\ z_j(t_2) = & C_1 \sin(A) + C_2 \cos(A) \\ & - \Delta_{rp} \cos(\theta_p(t_2)) + \frac{\upsilon(t_1)}{\omega_r(t_1)} \sin(\theta_p(t_2)) - c_x \end{cases} \tag{2}$$

where:

$$\begin{cases} A = & \Big(\omega_r(t_1) + \omega_p(t_1)\Big) T_s \\ C_1 = & \frac{Y_j(t_1) z_j(t_1)}{f} - \Delta_{rp} \sin(\theta_p(t_1)) - \frac{\upsilon(t_1)}{\omega_r(t_1)} \cos(\theta_p(t_1)) + c_y \\ C_2 = & z_j(t_1) + \Delta_{rp} \cos(\theta_p(t_1)) - \frac{\upsilon(t_1)}{\omega_r(t_1)} \sin(\theta_p(t_1)) + c_x \end{cases}$$

C. Equivalent Control Vector

In order to modify the solution provided by the solver, one will use the equivalent control vector method presented in [19]. It consists in computing the smallest sequence of control inputs connecting two states of a system, in our case the camera states. It has been shown in [19] that the considered robotic system is controllable in one step. Thus, it exists a constant control input vector $\tilde{\mathbf{Q}}_{t_1|t_2}$, named equivalent control vector, allowing to reach in one step any camera state $\chi_c(t_2)$ from $\chi_c(t_1)$, with $t_2 > t_1$. This equivalent control vector allows to link two images without the need of intermediate ones as it is illustrated in Fig. 2. Defining $\Delta_{x_c} = x_c(t_2) - x_c(t_1)$ and $\Delta_{y_c} = y_c(t_2) - y_c(t_1)$,

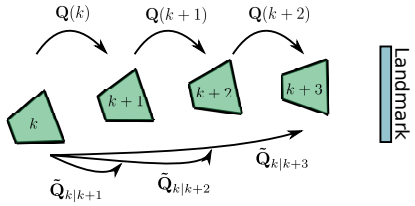


Fig. 2: Example of equivalent control vectors

the equations computing each element of $\tilde{\mathbf{Q}}_{t_1|t_2}$ have been identified as [19]:

$$\begin{aligned} \tilde{\upsilon} &= \sqrt{\tilde{\omega}_r^2 \left(\frac{\Delta_{x_c}^2 + \Delta_{y_c}^2}{4 \sin^2(\eta_{C1})} - \Delta_{rp}^2 \right)} \quad \tilde{\omega}_r = \frac{2}{T_s} \arctan(\gamma) \\ \tilde{\omega}_p &= \frac{1}{T_s} \left[\theta_p(t_2) - \theta_p(t_1) + \theta_r(t_2) - \theta_r(t_1) + \tilde{\omega}_r T_s \right] \end{aligned} \tag{3}$$

with $\gamma = -\frac{\Delta_{x_c} \sin(\theta_r(t_1)) + \Delta_{y_c} \cos(\theta_r(t_1))}{2\Delta_{rp} + \Delta_{x_c} \cos(\theta_r(t_1)) + \Delta_{y_c} \sin(\theta_r(t_1))}$, $\eta_{C1} = \frac{\omega_r(k) T_s}{2}$ and $\eta_{C2} = \frac{2\theta_r(k) + \omega_r(k) T_s}{2}$. By using the formulas (3), it is now possible to obtain the equivalent control vector $\tilde{\mathbf{Q}}_{t_1|t_2} = [\tilde{\upsilon}, \tilde{\omega}_r, \tilde{\omega}_p]^T$ linking the image at instant t_2 from the image at t_1.

III. Visual Predictive Control

A. The VPC Scheme

A VPC scheme consists in coupling NMPC with IBVS. On the one hand, similarly to NMPC, it consists of computing an optimal control sequence $\overline{\mathbf{Q}}^*(.)$ that minimizes a cost function J_{N_p} over a prediction horizon of N_p steps while taking into account a set of user-defined constraints $\mathbf{C}(\overline{\mathbf{Q}}^*(.))$. The optimal control sequence is of length N_c, which represents the control horizon. In other words, the N_c^{th} first predictions are computed using independent control inputs, while the remaining ones are all obtained using a unique control input equals to the last element of $\overline{\mathbf{Q}}^*(.)$. On the other hand, similarly to IBVS, the task to achieve is defined as an error in the image space. To do so, ones defines $\mathbf{S}$ as the vector containing the coordinates of N_v visual features and $\mathbf{S}^*$ as the one containing their reference values. In this work, one uses points as visual features, and in this particular case $\mathbf{S} = [X_1, Y_1, ..., X_j, Y_j, .., X_{N_v}, Y_{N_v}]^T$. Finally, the cost function to minimize is defined as the sum of the quadratic error between the visual feature coordinates vector $\hat{\mathbf{S}}(.)$ predicted over the horizon N_p and the desired ones $\mathbf{S}^*$. The optimal problem is then defined as follows:

$$\overline{\mathbf{Q}}^*(.) = \underset{\overline{\mathbf{Q}}(.)}{\min} \left(J_{N_p}(\mathbf{S}(k), \overline{\mathbf{Q}}(.)) \right) \tag{4}$$

with

$$J_{N_p}(\mathbf{S}(k), \overline{\mathbf{Q}}(.)) = \sum_{p=k+1}^{k+N_p} [\hat{\mathbf{S}}(p) - \mathbf{S}^*]^T [\hat{\mathbf{S}}(p) - \mathbf{S}^*] \tag{5}$$

subject to

$$\hat{\mathbf{S}}(k+1) = f(\hat{\mathbf{S}}(k), \mathbf{Q}(k)) \tag{6a}$$

$$\hat{\mathbf{S}}(k) = \mathbf{S}(k) \tag{6b}$$

$$C(\overline{\mathbf{Q}}^*(.)) \leq 0 \tag{6c}$$

where $\overline{\mathbf{Q}}(.) = [\mathbf{Q}(k), ..., \mathbf{Q}(k+N_p-1)]$. The prediction function $f(\hat{\mathbf{S}}(k), \mathbf{Q}(k))$ in equation (6a) corresponds to the prediction model given in equation (2). Moreover, the predicted visual features rely on the last measured ones, as stated by equation (6b). Finally, equation (6c) is used to include the constraints in the optimization problem. The set of constraints is presented in the following sections.

Remark 1: Solving equation (4) leads to the optimal sequence $\overline{\mathbf{Q}}^*(.)$. As it is usually done, only the first element $\overline{\mathbf{Q}}^*(1)$ is applied to the system. At the next iteration, the minimization problem is restarted, and a new $\overline{\mathbf{Q}}^*(.)$ is computed. This loop is repeated until the task is achieved.

Remark 2: Numerical solvers require an initial value for the vector to optimize. In this work, the results of the previous optimization are used as the initial guess of the current one.

B. The Zero Terminal Equality Constraint

In this work, the stability of the VPC scheme relying on a finite prediction horizon is achieved by adding a zero terminal equality constraint. It is defined as the error between the prediction of the visual feature $\hat{\mathbf{S}}(k+N_p)$ obtained at the end of the prediction horizon, and the desired ones $\mathbf{S}^*$.

$$||\hat{\mathbf{S}}(k+N_p) - \mathbf{S}^*|| = 0 \tag{7}$$

The equality constraint (7) being almost impossible to achieve, one uses the following inequality constraint:

$$||\hat{\mathbf{S}}(k+N_p) - \mathbf{S}^*|| - \delta_{tc} \leq 0 \tag{8}$$

where δ_{tc} is a user defined threshold sufficiently small to impact the optimization process similarly to the equality while offering an efficient implementation of the constraint.

The respect of this constraint at each iteration guarantees the recursive feasibility, *i.e.,* there exists a trajectory from the current state leading to the desired one. If the solver cannot compute a $\overline{\mathbf{Q}}^*(.)$ that fulfills this constraint, then the prediction horizon is too short and/or the constraints on the control inputs are too restrictive to reach the goal [4]. This issue is addressed in the following section.

C. The Input Constraints

The constraints applied to the control input vector are usually in the form of boundaries. They allow to take into account the physical limits of the actuators. However, for a given number of prediction steps N_p, they also limit the size of the feasibility set. To overcome this issue, it is proposed to divide the set of control input constraints into two subsets. In the first one, the boundaries correspond to the actual limits of the actuators and are named tight boundaries. In the second one, the boundaries are relaxed and do not have any physical sense. These boundaries are called relaxed or extended. Thus, one obtains the following inequality constraints:

$$\begin{bmatrix} \mathbf{Q}(i) - \mathbf{Q}_{u|t} \\ \mathbf{Q}_{l|t} - \mathbf{Q}(i) \end{bmatrix} \leq 0, \text{ if } 1 \leq i \leq N_c - N_r$$

$$\begin{bmatrix} \mathbf{Q}(i) - \mathbf{Q}_{u|r} \\ \mathbf{Q}_{l|r} - \mathbf{Q}(i) \end{bmatrix} \leq 0, \text{ if } N_c - N_r < i \leq N_c \tag{9}$$

where $i \in [1, ..., N_c]$, N_r is the number of prediction steps with relaxed boundaries, $\mathbf{Q}_{l|t}$ and $\mathbf{Q}_{u|t}$ are respectively the lower and upper tight boundaries corresponding to the actuators limits, and $\mathbf{Q}_{l|r}$ and $\mathbf{Q}_{u|r}$ are respectively the lower and upper relaxed boundaries. Thus, the command applied to the robot, which belongs to the first part of the prediction horizon, respects the actuators boundaries, while the feasible set is enlarged by the use of extended boundaries. Note that, ideally one would like to set the relaxed constraints to infinity but the majority of the solvers requires finite boundaries.

D. The Obstacle Avoidance Constraints

To perform a safe navigation, constraints can be used to avoid collision with the obstacles in the vicinity of the robot. Traditionally, the avoidance is obtained by guaranteeing a minimal distance between one or several points of the obstacles and the centroid of the robot for each predicted state. When considering small displacements between two

states, this approach is sufficient to avoid collisions. However, the use of relaxed input boundaries may lead to large displacements, requiring then to check the risk of collision along the trajectories, and not only for the predicted states.

In this work, the control inputs are considered constant between two sampling times. Thus, for the given robotic system, the pieces of trajectory performed between two consecutive states are either a segment when $\omega_r = 0$ or an arc of circle when $\omega_r \neq 0$. The constraint to avoid collision consists then in guaranteeing a minimal distance between the points representing the obstacle and a segment or an arc of circle. For N_o points C_m representing the obstacles, with $m \in [1, ..., N_o]$, the set of constraints can be written as:

$$\delta_c - \Delta(C_m, \hat{\chi}(n)|\hat{\chi}(n+1)) \leq 0 \tag{10}$$

where $n \in [1, ..., N_p - 1]$ and $\Delta(C_m, \hat{\chi}(n)|\hat{\chi}(n+1))$ is the shortest distance between C_m and the piece of trajectory between two consecutive predicted states $\hat{\chi}(n)$ and $\hat{\chi}(n+1)$. Finally, δ_c is a user-defined distance preventing collisions.

IV. Optimization of the Sub-Optimal Solution

The optimal problem being defined, it is necessary to compute a solution at each iteration, which is usually achieved by a numerical solver. For complex problems (nonlinear cost functions and constraints, non convex sets, large control input vectors), it is challenging to compute the global solution, and usually the solver provides a local solution.

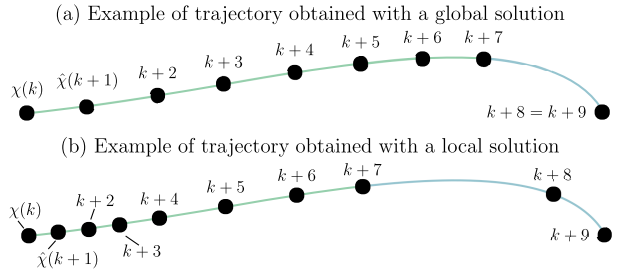


Fig. 3: Example of trajectories with $N_c = 9$ and $N_r = 2$. (a) Global solution: the first extended command allows reaching the goal and the second one is null. (b) Local solution: the two extended commands are used to reach the goal.

For a regulation problem, all the states of a global solution are as close as possible to the desired values while dealing with the constraints. Thus, in the case where there are more prediction steps than required to reach the goal, the unnecessary steps are the last ones and have null command values. For the proposed approach, it means that the global solution minimizes the use of the last steps with the relaxed constraints. They only allow completing the trajectory and are null when unnecessary (see figure 3). When only a local solution can be computed, there is no more guarantee the pieces of trajectory obtained with the relaxed constraints are minimal (see figure 3). In the worst case scenario, the first control inputs are null and the trajectory is only composed of the steps with the extended control inputs. In such a case, either the closely similar ones, the first command is null, or quasi null, and the robot is stuck in a local minima.

To take full advantage of the introduction of the relaxed constraints while dealing with such issues, one proposes to modify the obtained trajectory. The objective is to conserve the overall shape of the trajectory while modifying the sequence of control inputs to avoid local minima and improve performances. To achieve this aim, one introduces a new two steps method. First, using the equivalent control vectors one tries to merge some of the control inputs. Thus, the null commands are merged with other ones and local minima are avoided. Next, to compensate for the merged control inputs and to conserve the original shape of the trajectory, one creates new commands by breaking the extended commands into smaller pieces. These two steps are repeated for each element of the control input vector to improve the whole command vector. They are now presented in greater detail.

A. Commands Merging

In order to merge commands, one first computes the N_p equivalent control vectors $\tilde{\mathbf{Q}}_{k|k+i}$ between the initial camera pose at instant t_k and the N_p predicted ones at the predicted instants $t_k + i * T_s$, with $i \in [1, ..., N_p]$. Next, one needs to find among the $\tilde{\mathbf{Q}}_{k|k+i}$ respecting the boundaries $\mathbf{Q}_{l|t}$ and $\mathbf{Q}_{u|t}$ and the collision constraints, the one providing the largest piece of trajectory. It corresponds to the $\tilde{\mathbf{Q}}_{k|k+i}$ with the highest value for i among the ones dealing with the constraints. One denotes the highest value of i as N_m and one defines $\mathbf{Q}_M = \tilde{\mathbf{Q}}_{k|k+N_m}$. When $N_m > 1$, it means that $\mathbf{Q}_M$ merges the N_m first control inputs $\mathbf{Q}(k), ..., \mathbf{Q}(k+N_m-1)$.

Now that a merging command dealing with the constraints has been computed, it has to be included in the sequence $\overline{\mathbf{Q}}^*(.)$. Let first define N_s as the number of tight commands that are conserved in the merging process and N_z as the number of commands that disappear and need to be replaced.

$$\begin{aligned} N_s &= N_c - N_r - N_m \\ N_z &= N_m - 1 \end{aligned} \tag{11}$$

To include the merging command $\mathbf{Q}_M$, the control sequence $\overline{\mathbf{Q}}^*(.)$ is modified as follows (see figure 4.b for an example):

- The first element $\overline{\mathbf{Q}}^*(1)$ is equal to $\mathbf{Q}_M$.
- The N_s following elements $[\overline{\mathbf{Q}}^*(2), ..., \overline{\mathbf{Q}}^*(2+N_s-1)]$ are equal to $[\mathbf{Q}(k+N_m), ..., \mathbf{Q}(k+N_m+N_s-1)]$.
- The N_z following elements $[\overline{\mathbf{Q}}^*(2+N_s), ..., \overline{\mathbf{Q}}^*(1+N_s+N_z)]$ are null.
- The last N_r elements are not modified.

B. Extraction of New Commands

The merging command being calculated and included, it is now proposed to extract tight commands from the relaxed ones to replace the null ones introduced in the previous step. The approach consists in extracting a piece of the trajectory obtained with the relaxed commands. One first defines two indices $u_n \in [N_c - Nr - Nz, ..., N_c - Nr]$ and $u_e = N_c - N_r + 1$ to respectively iterate over the null commands and the extended ones. Next, one defines a gain λ to extract from

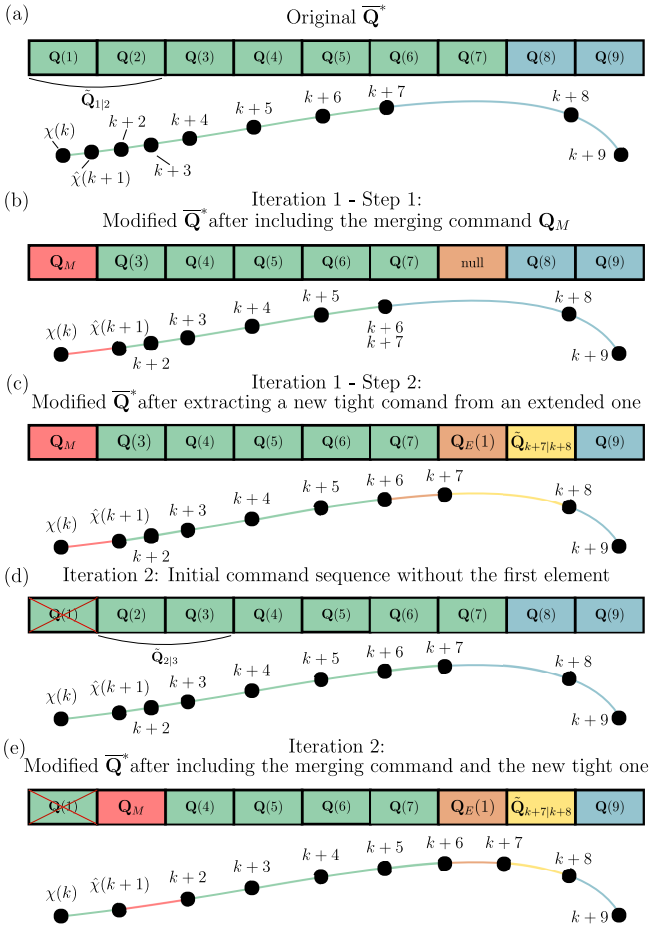


Fig. 4: Example of trajectory improvement. (a) Initial trajectory with $N_c = 9$ and $N_r = 2$. (b) The two first commands are merged, $N_m = 2$, $N_s = 5$, and $N_z = 1$. The merging command is included as the first element, the five remaining tight commands are copied and the seventh one is null. (c) The null command is replaced by a tight one extracted from the extended one. The extended command is updated. (d) Next iteration: the first element is removed from the merging process. (e) The second element is now the command merging the second and third commands. A tight command is included in the seventh element to compensate the merging one and the first extended command is updated. At this point the trajectory cannot be improved anymore.

the extended command the longest piece of trajectory lying within the tight bounds. It is computed as follows:

$$\lambda = max\left(\frac{\tau_\upsilon}{|\upsilon(u_e)|}, \frac{\tau_{\omega_r}}{|\omega_r(u_e)|}, \frac{\tau_{\omega_p}}{|\omega_p(u_e)|}\right) \tag{12}$$

where τ_υ, τ_{ω_r}, and τ_{ω_p} are the upper boundaries on υ, ω_r, and ω_p. If $\upsilon(u_e)$, $\omega_r(u_e)$ and $\omega_p(u_e)$ are null or within the tight boundaries, then the index u_e is incremented by one to consider the next extended control inputs. λ being calculated, one obtains the new commands as follows:

$$\mathbf{Q}_E(u_n) = \lambda \mathbf{Q}(u_e) \tag{13}$$

Finally, after extracting a new tight command from an extended one, one updates the extended command to conserve the original trajectory. To do so, one computes the new state $\hat{\chi}_c(k+u_n)$ obtained with $\mathbf{Q}_E(u_n)$. It is then possible to compute the equivalent control vector $\tilde{\mathbf{Q}}_{k+u_n|k+u_e}$ between this new state and the end of the trajectory piece obtained with the extended command. This equivalent control vector is used as the updated extended control input (see figure 4.c).

The use of this method guarantees that the first command is non-null, which prevents local minima. Moreover, the two steps are repeated to process the whole control sequence. At each iteration the control sequence is updated by removing its first element from the merging process (see figure 4.d). Thus, the whole control sequence is improved. Although only the first command is applied, providing a modified control sequence as initial values to the next optimization process allows improving the next calculated trajectory.

Configuration	N_p	N_c	N_r	TC	OC	TI
Ω_1	15	15	0	No	Yes	No
Ω_2	15	15	5	Yes	Yes	No
Ω_3	15	15	5	Yes	Yes	Yes

TABLE I: Configuration description - TC: terminal constraint - OC: obstacle constraint - TI: trajectory improvement

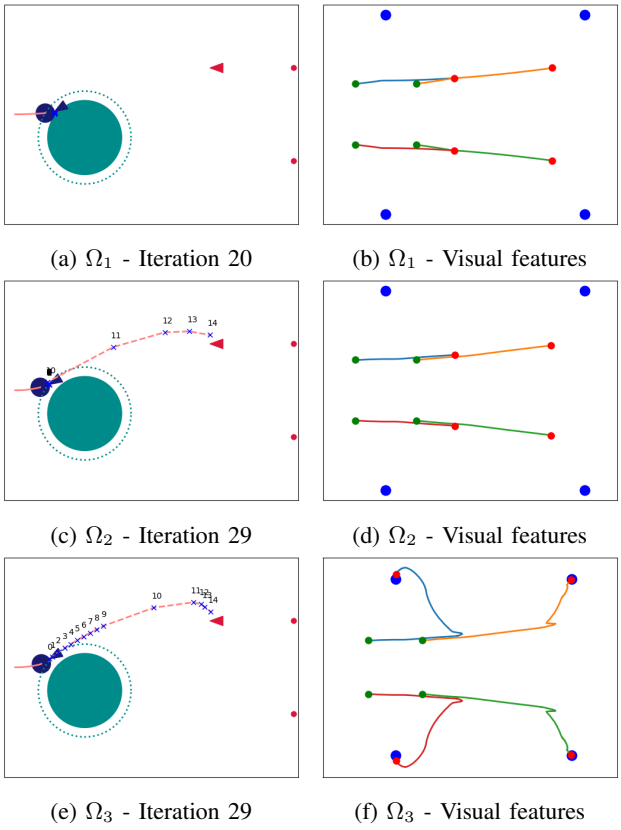
V. Results

In this section, we present the results obtained simulating a VPC servoing for a differential drive robot equipped with a camera[1]. The program was implemented using the C++ language and the cost function minimization was done with the SQP solver from the NLopt package [21].

In this work, we consider the depth of the visual features as known. Moreover, at the first step, the minimization problem is solved with a control vector equal to zero. For the next navigation steps, it is initialized with the results of the previous minimization. Finally, the tight boundaries are setup such as $0 \leq \upsilon \leq 0.4m/s$, $-0.1rad/s \leq \omega_r \leq 0.1rad/s$, and $-0.1rad/s \leq \omega_p \leq 0.1rad/s$, and the extended ones are ten times larger. In the figures, the robot is represented in dark blue, the path of the mobile base by a plain orange line, and the predicted path of the camera by a dashed orange line. The desired camera pose is symbolized by a red triangle and the landmark is represented by red points. The obstacle is represented by a plain green circle and the safety boundary by a pointed green circle. In the figures representing the visual features evolution, green dots are the initial values, red dots the final values, and blue ones are the desired values.

One considers the three configurations described in table I. For the first one Ω_1, the range covered by the prediction horizon does not allow reaching the desired state, *i.e.,* the stability is not guaranteed. The robot drives towards the obstacle, reaches a local minima and fails to make the visual features converge the desired ones (fig. 5a and 5b). With the second configuration Ω_2, the range covered by the prediction horizon allows reaching the desired state and dealing with the terminal constraint, *i.e.,* stability is guaranteed when the

[1]A video is also available: https://youtu.be/u9KKZJ-MTXk



(a) Ω_1 - Iteration 20 (b) Ω_1 - Visual features

(c) Ω_2 - Iteration 29 (d) Ω_2 - Visual features

(e) Ω_3 - Iteration 29 (f) Ω_3 - Visual features

Fig. 5: Simulated results for the three configurations

global solution of the optimization problem is calculated. However, in this example only a local solution is computed (SQP solver) which is not sufficient to prevent the navigation failure. Indeed, the robot once again drives towards the obstacle, reaches a local minima and fails to make the visual features converge the desired ones (fig. 5c and 5d). At the state corresponding to the local minimum, the local solution offers a trajectory reaching the desired state, but mostly using the steps with extended constraints. The local minimum is due to the commands with tight constraints being null or quasi-null, preventing the robot to achieve the task. Finally, with Ω_3 the local minima is avoided by modifying the obtained trajectory. As it can be seen if figure 5e, the predicted trajectory is now made of steps with both tight and extended boundaries. The trajectory optimization efficiently minimized the use of steps with extended boundaries, which allowed the robot to achieve the navigation task (figure 5f).

VI. Conclusion

In this work, one proposes to guarantee the stability of a VPC scheme by relaxing some of the control input boundaries. Moreover, an obstacle avoidance constraint over the pieces of trajectory is added. Finally, a two steps methods relying on the equivalent control input method allows to modify the obtained trajectory and to improve the solution suboptimality. Thus, the robot can efficiently reach the desired state while dealing with obstacles and without being stuck in local minima as shown in the result section. It is now planned to extend the navigation process by adding constraints dealing with visual feature occultation, and to implement and test our approach on a robotic system.

References

[1] G. Allibert, E. Courtial, and F. Chaumette, "Predictive control for constrained image-based visual servoing," *IEEE Trans. on Robotics*, vol. 26, no. 5, pp. 933–939, October 2010.

[2] F. Chaumette and S. Hutchinson, "Visual servo control, part 1 : Basic approaches," *Robotics and Automation Mag.*, vol. 13, no. 4, 2006.

[3] F. Allgower, R. Findeisen, Z. K. Nagy, *et al.*, "Nonlinear model predictive control: From theory to application," *Journal-Chinese Institute Of Chemical Engineers*, vol. 35, no. 3, pp. 299–316, 2004.

[4] L. Grüne and J. Pannek, "Nonlinear model predictive control," in *Nonlinear Model Predictive Control*. Springer, 2017, pp. 45–69.

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

[6] A. Assa and F. Janabi-Sharifi, "Robust model predictive control for visual servoing," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Sep. 2014, pp. 2715–2720.

[7] A. Paolillo, T. S. Lembono, and S. Calinon, "A memory of motion for visual predictive control tasks," in *International Conference on Robotics and Automation*, no. CONF, 2020.

[8] S. Heshmati-alamdari, G. K. Karavas, A. Eqtami, M. Drossakis, and K. J. Kyriakopoulos, "Robustness analysis of model predictive control for constrained image-based visual servoing," in *2014 IEEE Int. Conf. on Robotics and Automation*, May 2014, pp. 4469–4474.

[9] A. Mcfadyen, P. Corke, and L. Mejias, "Visual predictive control of spiral motion," *IEEE Transactions on Robotics*, vol. 30, no. 6, pp. 1441–1454, 2014.

[10] F. Ke, Z. Li, H. Xiao, and X. Zhang, "Visual servoing of constrained mobile robots based on model predictive control," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 47, no. 7, pp. 1428–1438, July 2017.

[11] D. Lee, H. Lim, and H. J. Kim, "Obstacle avoidance using image-based visual servoing integrated with nonlinear model predictive control," in *2011 50th IEEE Conference on Decision and Control and European Control Conference*. IEEE, 2011, pp. 5689–5694.

[12] A. Hajiloo, M. Keshmiri, W. Xie, and T. Wang, "Robust online model predictive control for a constrained image-based visual servoing," *IEEE Transactions on Industrial Electronics*, vol. 63, no. 4, pp. 2242–2250, April 2016.

[13] C. Copot, C. Lazar, and A. Burlacu, "Predictive control of nonlinear visual servoing systems using image moments," *IET control theory & applications*, vol. 6, no. 10, pp. 1486–1496, 2012.

[14] C. Lazar, A. Burlacu, and C. Copot, "Unified point and image moment features for image based predictive visual servoing systems," in *2012 13th International Conference on Optimization of Electrical and Electronic Equipment (OPTIM)*. IEEE, 2012, pp. 1458–1464.

[15] F. Fusco, O. Kermorgant, and P. Martinet, "Integrating features acceleration in visual predictive control," *IEEE Robotics and Automation Letters*, 2020.

[16] D. Q. Mayne and H. Michalska, "Receding horizon control of nonlinear systems," in *Proceedings of the 27th IEEE Conference on Decision and Control*. IEEE, 1988, pp. 464–465.

[17] S. a. Keerthi and E. G. Gilbert, "Optimal infinite-horizon feedback laws for a general class of constrained discrete-time systems: Stability and moving-horizon approximations," *Journal of optimization theory and applications*, vol. 57, no. 2, pp. 265–293, 1988.

[18] H. Chen and F. Allgöwer, "A quasi-infinite horizon nonlinear model predictive control scheme with guaranteed stability," *Automatica*, vol. 34, no. 10, pp. 1205–1217, 1998.

[19] A. Durand-Petiteville, "Navigation référencée multi-capteurs d'un robot mobile en environnement encombré," Ph.D. dissertation, Université Paul Sabatier-Toulouse III, 2012.

[20] D. Folio and V. Cadenat, *Treating Image Loss by using the Vision/Motion Link: A Generic Framework*. IN-TECH, 2008, ch. 4.

[21] S. G. Johnson, "The nlopt nonlinear-optimization package," 2020. [Online]. Available: http://github.com/stevengj/nlopt