DOI: 10.48011/asba.v2i1.1451

Mobile Robot Navigation in a Cluttered Environment via Visual Predictive Control

Adrien Durand-Petiteville * **Viviane Cadenat** **

* *Universidade Federal de Pernambuco UFPE, Departamento de Engenharia de Mecânica, s/n, Av. da Arquitetura, Recife, PE, 50740-550, Brazil (e-mail: adrien.durandpetiteville@ufpe.br).*
** *Univ de Toulouse, UPS, LAAS, F-31400, Toulouse, France CNRS, LAAS, 7 avenue du colonel Roche, F-31400 Toulouse, France (e-mail: cadenat@laas.fr)*

Abstract: This paper presents a Visual Predictive Controller scheme for a differential drive robot navigating in a cluttered environment. We introduce an analytic model predicting the future state for this specific system Moreover, constraints guaranteeing the convergence of the control law, and avoiding occultations and collisions with obstacles are presented. A large set of results obtained in simulations highlights the interest and efficiency of the approach.

Keywords: Mobile Robotics, Navigation, Visual Servoing, Model Predictive Control, Obstacle avoidance.

1. INTRODUCTION

Image-Based Visual Servoing (IBVS), a sub-division of visual servoing, aims at controlling the motion of a camera mounted on a robotic system. To do so, the control law is designed in the image space and relies on the minimization of an error between the values of some visual features computed in the current image and their values of reference, corresponding to the task to achieve Chaumette and Hutchinson (2006). Thus, there is no need to estimate the robot localization in a global frame and it is sufficient to track the visual features in the image. This method is usually chosen for its reactivity, the absence of localization, and its large stability margins Chaumette (1998).

When using an IBVS scheme, the control law is computed in the image space, usually leading to an efficient convergence of the visual features towards the reference ones. However, the camera pose is not taken into account by the control law, making the trajectory entirely dependent of the number and types of visual features used to define the task (points, lines, moments, etc Chaumette (2004)). It is then difficult to predict the trajectory or to provide any guarantee regarding the camera pose during the servoing in order to avoid collisions or occultations. Moreover, the evolution of the visual features in the image and the camera trajectory can be incompatible and induce a failure of IBVS. For example, it is well known that a classical IBVS controller cannot achieve a pure rotation around a set of point visual features Chaumette (1998). In order to overcome these issues, several works propose to use hybrid methods, such as Malis et al. (1999) or Corke and Hutchinson (2001), where some of the camera degrees of freedom are controlled with an IBVS scheme and the remaining ones with a control law expressed in the Cartesian space. However, these approaches only allow dealing with one specific problem at the time and do not present any global guarantee regarding the camera pose and trajectory.

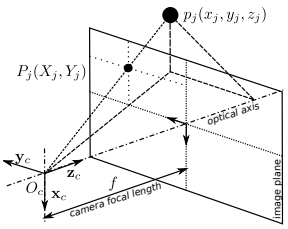
Visual Predictive Control (VPC) Allibert et al. (2010) is the fusion between Image-Based Visual Servoing (IBVS) Chaumette and Hutchinson (2006) and Nonlinear Model Predictive Control (NMPC) Grüne and Pannek (2017). The task to achieve is defined by a cost function expressed in the image space. It is the sum over a prediction horizon of the difference between the predicted visual features and the desired ones. Then, the command is obtained by minimizing at each iteration the cost function. The minimization process is performed by a numerical solver and is usually subject to constraints. The obtained control scheme thus combines the advantages of IBVS, *i.e.,* reactivity, absence of metric localization, and large stability margins Chaumette (1998), with the ones of NMPC, *i.e.,* ability to explicitly deal with constraints such as feature visibility, collision with obstacles, or control inputs boundaries.

Over the last decade, VPC schemes have mostly been developed for robotic system made of a camera mounted on a robotic arm, such as in Allibert et al. (2010), Assa and Janabi-Sharifi (2014) and Qiu et al. (2019). In Copot et al. (2012) and Lazar et al. (2012), the proposed VPC schemes are slightly modified and use image moments as visual features. In Wang et al. (2012) and Hajiloo et al. (2016), the visual servoing system is represented as a polytopic linear parameter-varying system and offers interesting properties regarding the robustness of the controller. In Flécher et al. (2019), the VPC scheme is used to simultaneously control two robotic arms sharing a same workspace. In addition to robotic arms, other systems were considered such as a flying camera in Heshmati-alamdari et al. (2014) and Mcfadyen et al. (2014), a mobile robot in Ke et al. (2017), and a fixed-wing aerial vehicle in Lee et al. (2011). These works focus on the design of the VPC scheme subject

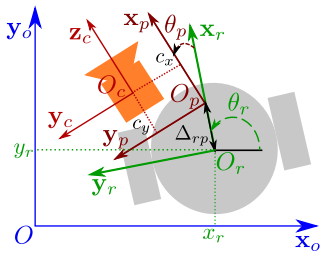
to control inputs and field of view constraints, usually for small displacements of the camera between the initial and desired poses, which limits the impact of the prediction errors and the need for a large prediction horizon. Furthermore, the constraints due to external elements, which requires accurate prediction models, are not taken into account. Finally, there is no guarantee regarding the system convergence given that the feasibility problem and the terminal constraint are not addressed.

In this work, it is proposed a VPC scheme for a differential drive robot navigating in a cluttered environment. We first present the generic VPC framework and the main parameters, as well as the usual local and global models Allibert et al. (2010) used to predict the future states. We then show that it is possible to obtain a more accurate local model when considering a differential drive robot. Next, we focus on a set of constraints allowing to successively and safely navigate in a cluttered environment. The first constraint is the terminal constraint which allows to check the feasibility of the problem and that can be used to guarantee the convergence of the control law. The second constraint is used to keep the landmark of interest in the field of view of the camera, and the last one allows to avoid the obstacles lying in the scene. The last part of the paper presents a large set of results obtained in simulations. These results show the impact of the main parameters of a VPC scheme and the usefulness of the previously mentioned constraint to deal with a cluttered environment. We conclude the paper by giving an overview of what are the main challenges in order to implement an efficient VPC scheme on a robot.

2. SYSTEM MODELING



(1.1) Camera pinhole model



(1.2) Differential robot model

Figure 1. System model

In this work, a pinhole camera is controlled via a VPC scheme. To model the system, an orthonormal frame $\mathbf{F}_c(O_c, \mathbf{x}_c, \mathbf{y}_c, \mathbf{z}_c)$ represents the camera, and the focal length is denoted by f (see Fig. 1.1). The camera pose χ_c is expressed in the world frame $\mathbf{F}_o(O, \mathbf{x}_o, \mathbf{y}_o, \mathbf{z}_o)$. Finally, the camera kinematic screw $\mathbf{\Gamma}^{F_c}_{O_c/F_o}$, calculated at point O_c with respect to $\mathbf{F}_0$ and expressed in $\mathbf{F}_c$, can be decomposed as follows:

$$\mathbf{\Gamma}^{\mathbf{F}_c}_{O_c/\mathbf{F}_o} = \mathbf{\Gamma} = \left[V^{\mathbf{F}_c}_{\mathbf{x}_c} \; V^{\mathbf{F}_c}_{\mathbf{y}_c} \; V^{\mathbf{F}_c}_{\mathbf{z}_c} \; \Omega^{\mathbf{F}_c}_{\mathbf{x}_c} \; \Omega^{\mathbf{F}_c}_{\mathbf{y}_c} \; \Omega^{\mathbf{F}_c}_{\mathbf{z}_c} \right]^T \quad (1)$$

where $V^{\mathbf{F}_c}_{\mathbf{x}_c}$, $V^{\mathbf{F}_c}_{\mathbf{y}_c}$ and $V^{\mathbf{F}_c}_{\mathbf{z}_c}$ are respectively the linear velocities along $\mathbf{x}_c$, $\mathbf{y}_c$ and $\mathbf{z}_c$ expressed in $\mathbf{F}_c$. Following the same idea, $\Omega^{\mathbf{F}_c}_{\mathbf{x}_c}$, $\Omega^{\mathbf{F}_c}_{\mathbf{y}_c}$ and $\Omega^{\mathbf{F}_c}_{\mathbf{z}_c}$ are respectively the angular velocities along $\mathbf{x}_c$, $\mathbf{y}_c$ and $\mathbf{z}_c$ expressed in $\mathbf{F}_c$.

The camera is embedded on a differential robot equipped with a pan-platform. Let define $\mathbf{F}_r(O_r, \mathbf{x}_r, \mathbf{y}_r, \mathbf{z}_r)$ the robot frame and $\mathbf{F}_p(O_p, \mathbf{x}_p, \mathbf{y}_p, \mathbf{z}_p)$ the platform frame (see Fig. 1.2). 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 $\mathbf{x}_r$, O_p the pan-platform center of rotation and Δ_{rp} the distance between the robot reference point O_r and O_p. Moreover, with x_r and y_r the coordinates of the point O_r in $\mathbf{F}_o$, one defines a the mobile base state as $\chi_r = [x_r, y_r, \theta_r]^T$. The control input 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 $\mathbf{F}_r$. Thus, it is possible to obtain the following kinematic model for the mobile base:

$$\begin{bmatrix} \dot{x}_r \\ \dot{y}_r \\ \dot{\theta}_r \end{bmatrix} = \begin{bmatrix} \upsilon \cos(\theta_r) \\ \upsilon \sin(\theta_r) \\ \omega_r \end{bmatrix} \quad (2)$$

The camera being embedded on a differential robot equipped with a pan-platform, it only has three degrees of freedom. For this reason, 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 a reduced camera state and kinematic screw as follows:

$$\bar{\chi}_c = [x_c, y_c, \theta_c]^T \quad (3)$$

$$\begin{aligned} \overline{\mathbf{\Gamma}}^{\mathbf{F}_c}_{O_c/\mathbf{F}_o} &= \overline{\mathbf{\Gamma}} = \left[V^{\mathbf{F}_c}_{\mathbf{y}_c}, V^{\mathbf{F}_c}_{\mathbf{z}_c}, \Omega^{\mathbf{F}_c}_{\mathbf{x}_c} \right]^T \\ &= \begin{bmatrix} \dot{x}_c \\ \dot{y}_c \\ \dot{\theta}_c \end{bmatrix} = \begin{bmatrix} \upsilon \cos(\theta_r) - \omega_r \Delta_{rp} \sin(\theta_r) \\ \upsilon \sin(\theta_r) + \omega_r \Delta_{rp} \cos(\theta_r) \\ \omega_r + \omega_p \end{bmatrix} \end{aligned} \quad (4)$$

Finally, 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. Therefore, the visual data are represented by a $2N_v$ dimensional vector $\mathbf{S}$. A 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 1.1).

3. VISUAL PREDICTIVE CONTROL

In this section, we first recall the VPC framework and the main parameters impacting the controller behavior. Then, we present three constraints that have to be taken into account in the VPC problem in order to guarantee the convergence of the closed-loop system and the safety of the robot.

3.1 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 equal 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, one 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 $\mathbf{S}^*$. Note that the proposed cost function is the one traditionally used for VPC schemes, but it does not represent the only choice.

The VPC problem is then given by:

$$\overline{\mathbf{Q}}^*(.) = \min_{\overline{\mathbf{Q}}(.)} \left(J_{N_p}(\mathbf{S}(k), \overline{\mathbf{Q}}(.)) \right) \tag{5}$$

with

$$J_{N_p}(\mathbf{S}(k), \overline{\mathbf{Q}}(.)) = \sum_{p=k+1}^{k+N_p} [\hat{\mathbf{S}}(p) - \mathbf{S}^*]^T [\hat{\mathbf{S}}(p) - \mathbf{S}^*] \tag{6}$$

subject to

$$\hat{\mathbf{S}}(p+1) = g(\hat{\mathbf{S}}(p+1), \mathbf{Q}(p)) \tag{7a}$$

$$\hat{\mathbf{S}}(k) = \mathbf{S}(k) \tag{7b}$$

$$C(\overline{\mathbf{Q}}^*(.)) \leq 0 \tag{7c}$$

where k represents the current iteration at instant t_k and $\overline{\mathbf{Q}}(.) = [\mathbf{Q}(k), ..., \mathbf{Q}(k+N_p-1)]$. Equation (7a) gives the model $g(\hat{\mathbf{S}}(p+1), \mathbf{Q}(p))$ used to predict the future states. In the case of VPC, the state is defined by the visual features and several models are available. We will present some of them in section 4. Moreover, the predicted visual features rely on the last measured ones, as stated by equation (7b). Finally, equation (7c) is used to include the constraints in the optimization problem via nonlinear inequalities. These latter will be presented in 3.2.

Thus, solving (5) leads to the optimal sequence of control inputs $\overline{\mathbf{Q}}^*(.)$. 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 sequence of optimal control inputs is computed. This loop is repeated until the task is achieved.

3.2 Definition of the constraints

The terminal constraint: When using a VPC scheme, it is necessary to guarantee that the set of control inputs makes the camera converge towards the desired pose. Due to the fact that a VPC scheme minimizes the distance between a set of predicted visual features and the desired ones, there is no guarantee that the ultimate predicted features have converged towards the desired ones. Moreover, the prediction horizon might be too short or the constraints on the control inputs might be too restrictive to reach the goal Grüne and Pannek (2017). Thus, to check the feasibility, one adds a terminal constraint in the optimization problem.

The terminal constraint is defined as the error between the prediction of the visual feature coordinates $\hat{\mathbf{S}}(k+N_p)$ obtained at the end of the prediction horizon, and the desired ones $\mathbf{S}^*$ (see (8) where δ_{tc} is a user defined threshold). If the solver cannot compute a sequence of control inputs $\overline{\mathbf{Q}}^*(.)$ that respects this constraint, then the problem is not feasible, and there is no guarantee regarding the system convergence.

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

where δ_{tc} is a user defined threshold. Two remarks can be made concerning the use of the terminal constraint. First, none of the works studied during the literature review proposes to include a terminal constraint, despite its ease of implementation and utility. Moreover, authors usually weight the last predicted value based on the distance to the desired one. This method helps the solver to converge towards an optimal solution but it does not guarantee the convergence as the terminal constraint does. Second, the terminal constraint does not provide an absolute guarantee of the task realization. Indeed, in the case the predictions are strongly erroneous, the system cannot converge towards the real values of the desired states.

The occultation avoidance constraints: Visual servoing schemes, including IBVS and VPC, rely on visual features in the image corresponding to landmarks in the scene. If the landmarks are no more in the field of view of the camera, it becomes then impossible to extract the visual features and the servoing fails. It is then mandatory to guarantee their visibility at any moment of the navigation. In this work, it is proposed to include a set of constraints guaranteeing that the visual features obtained over the prediction horizon stay within the image boundaries. To achieve this aim, ones define the following set of constraints:

$$\begin{bmatrix} \hat{\mathbf{S}}(k) - S_{max} \\ \vdots \\ \hat{\mathbf{S}}(k+N_p) - S_{max} \\ S_{min} - \hat{\mathbf{S}}(k) \\ \vdots \\ S_{min} - \hat{\mathbf{S}}(k+N_p) \end{bmatrix} \leq 0 \tag{9}$$

where S_{max} and S_{min} are two $2N_v$ long vectors respectively containing the upper and lower boundaries of the image space.

The obstacle avoidance constraints: Visual servoing schemes do not offer any guarantee regarding possible collisions of the robotic system with elements of the scene. It is then mandatory to define a constraint managing the distance between the robotic system and the obstacles lying in the environment. In this work, it is proposed to only consider static, non-occulting [1], and circle-shaped obstacles. Each of the N_o obstacles present in the scene is defined by $x_{o_m}, y_{o_m}, r_{o_m}$, with x_{o_m}, y_{o_m} the coordinates of the center, r_{o_m} the radius, and $m \in [1, ..., N_o]$. The collision risk is managed thanks to the following constraint:

$$\begin{bmatrix} \delta_s - \sqrt{(\hat{x}(k) - x_{o_1})^2 + (\hat{y}(k) - y_{o_1})^2} \\ \vdots \\ \delta_s - \sqrt{(\hat{x}(k+N_p) - x_{o_1})^2 + (\hat{y}(k+N_p) - y_{o_1})^2} \\ \delta_s - \sqrt{(\hat{x}(k) - x_{o_{N_o}})^2 + (\hat{y}(k) - y_{o_{N_o}})^2} \\ \vdots \\ \delta_s - \sqrt{(\hat{x}(k+N_p) - x_{o_{N_o}})^2 + (\hat{y}(k+N_p) - y_{o_{N_o}})^2} \end{bmatrix} \leq 0 \tag{10}$$

[1] Small enough for the camera to perceive the target from any configuration

with $\delta_s = d_s + r_{o_m} + r_r$, where r_r represents the mobile base radius and d_s the safety distance around an obstacle. The predicted positions of the mobile base $\hat{x}(.)$ and $\hat{y}(.)$ are obtained by integrating equation (2).

4. PREDICTION MODELS

A VPC scheme relies on models predicting the future states based on a given sequence of control inputs. Here, one presents three models: the two first ones are extracted from the literature Allibert et al. (2010) whereas the third one is specifically computed for a differential robot.

4.1 Global model

The global prediction model consists in de-projecting a point from the initial image to the initial camera frame ($\mathbb{R}^2 \rightarrow \mathbb{R}^3$), computes its coordinates in the prediction camera frame ($\mathbb{R}^3 \rightarrow \mathbb{R}^3$), and finally projects this point in the prediction image ($\mathbb{R}^3 \rightarrow \mathbb{R}^2$). By defining the projection matrix $\mathbf{H}_{i/c}$ and the homogeneous matrix $\mathbf{H}_{c(t_1)/c(t_2)}$ between two camera poses at instants t_1 and t_2 such as:

$$\begin{bmatrix} X_j \\ Y_j \\ z_j \\ 1 \end{bmatrix} = \begin{bmatrix} f/z_j & 0 & 0 & 0 \\ 0 & f/z_j & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_j \\ y_j \\ z_j \\ 1 \end{bmatrix} = \mathbf{H}_{i/c} \begin{bmatrix} x_j \\ y_j \\ z_j \\ 1 \end{bmatrix} \tag{11}$$

$$\mathbf{H}_{c(t_1)/c(t_2)} = \begin{bmatrix} \mathbf{R}_{c(t_1)/c(t_2)} & \mathbf{T}_{c(t_1)/c(t_2)} \\ 0_{1\times 3} & 1 \end{bmatrix} \tag{12}$$

where $\mathbf{R}_{c(t_1)/c(t_2)}$ and $\mathbf{T}_{c(t_1)/c(t_2)}$ are respectively a 3×3 rotation matrix and a 3×1 translation vector between $\mathbf{F}_c(t_1)$ and $\mathbf{F}_c(t_2)$, one obtains the global prediction model:

$$\mathbf{P_j}(t_2) = \mathbf{H}_{i/c(t_2)} \mathbf{H}^{-1}_{c(t_1)/c(t_2)} \mathbf{H}^{-1}_{i/c(t_1)} \mathbf{P_j}(t_1) \tag{13}$$

4.2 Local model

It relies on the mapping between the visual features evolution and the camera kinematic screw via the interaction matrix, classically given for a point P_j by Chaumette and Hutchinson (2006):

$$\mathbf{J}_i = \begin{bmatrix} \frac{-f}{z_j} & 0 & \frac{X_j}{z_j} & \frac{X_j Y_j}{f} & -(f + \frac{X_j^2}{f}) & Y_j \\ 0 & \frac{-f}{z_j} & \frac{Y_j}{z_j} & (f + \frac{X_j^2}{f}) & \frac{-X_j Y_j}{f} & X_j \end{bmatrix} \tag{14}$$

The local model is thus given by the integration between the current instant and the prediction one of the following equation:

$$\dot{\mathbf{S}}_j = \mathbf{J}_i \mathbf{\Gamma} \tag{15}$$

4.3 3 DOF local model

In this paper, it is proposed to predict the visual features using a local model in the case of a 3 DOF camera. In other words, equation (15) is modified to include the robotic system carrying the camera. Let define the robot Jacobian by Durand-Petiteville (2012):

$$\mathbf{J}_r = \begin{bmatrix} -\sin(\theta_p) & \Delta_{rp}\cos(\theta_p) + c_x & c_x \\ \cos(\theta_p) & \Delta_{rp}\sin(\theta_p) - c_y & -c_y \\ 0 & -1 & -1 \end{bmatrix} \tag{16}$$

where c_x and c_y are the coordinates of O_c along axes $\mathbf{x}_p$ and $\mathbf{y}_p$ (see figure 1.1), and the reduced interaction matrix, *i.e.,* for a 3 DOF camera, by

$$\overline{\mathbf{J}}_i = \begin{bmatrix} \frac{X_j}{z_j} & \frac{X_j Y_j}{f} & -(f + \frac{X_j^2}{f}) \\ \frac{Y_j}{z_j} & (f + \frac{X_j^2}{f}) & \frac{-X_j Y_j}{f} \end{bmatrix} \tag{17}$$

It is then possible to rewrite (15) such as:

$$\dot{\mathbf{S}}_j = \overline{\mathbf{J}}_i \overline{\mathbf{\Gamma}} = \overline{\mathbf{J}}_i \mathbf{J}_r \mathbf{Q} \tag{18}$$

The robot is a discrete system whose inputs evolve at each instant $t = kT_s$, where T_s is the sampling time. By assuming that the inputs $\mathbf{Q}(t_1)$ are constant during the two instants t_1 and $t_2 = t_1 + T_s$, it is then possible to solve (18) between t_1 and t_2. After some computations (see Folio and Cadenat (2008)), one obtains:

$$\begin{cases} X_j(t_2) = \frac{z_j(t_1) X_j(t_1)}{z_j(t_2)} \\ Y_j(t_2) = \frac{f}{z_j(t_2)} \Big\{ C_1 \cos(A) - C_2 \sin(A) \\ \qquad\qquad + \Delta_{rp} \sin(\theta_p(t_2)) + \frac{v(t_1)}{\omega_r(t_1)} \cos(\theta_p(t_2)) - c_y \Big\} \\ z_j(t_2) = C_1 \sin(A) + C_2 \cos(A) \\ \qquad\qquad - \Delta_{rp} \cos(\theta_p(t_2)) + \frac{v(t_1)}{\omega_r(t_1)} \sin(\theta_p(t_2)) - c_x \end{cases} \tag{19}$$

where:

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

This model is a closed-form expression that can be used to predict exactly the coordinates of the visual features. Moreover, it does not require any advanced/complex operation, offering a low computational cost.

Remark: *The three models rely on the z coordinate of the visual features. Thus, to accurately predict the values of the visual features, it is mandatory to estimate it or to measure it with a sensor such as a 3D camera. Using a constant value, which is widely used in classical IBVS Chaumette and Hutchinson (2006), might lead to large prediction errors, especially for the navigation problem where the value of z has huge variations.*

5. RESULTS

In this section, we present results obtained simulating a VPC servoing for a differential drive robot equipped with a camera. The robotic system and the environment were implemented using the C++ language. They were tested on an Intel Core i7-8750H CPU running at 2.20GHz. To minimize the cost function, we rely on the SQP solver from the NLopt nonlinear-optimization package Johnson (2020). Finally, the robot state and visual feature coordinates are simulated using the equations presented in section 2.

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 control variables are bounded as follows: $0 \leq v \leq 0.4M/s$, $-0.1rad/s \leq \omega_r \leq 0.1rad/s$, and $-0.1rad/s \leq \omega_p \leq 0.1rad/s$.

In the figures, the current robotic system is represented in dark blue. A plain orange line represents the path performed by the mobile base, whereas a dashed orange one is used for the predicted path of the camera. The camera pose to reach is symbolized by a red triangle and the landmark is represented by the red points. Obstacles are represented by plain green circles and the safety boundaries by pointed green circles. Finally, in the figures representing the evolution of the visual features, green dots are the initial values, red dots the final values, and blue ones are the desired values.

For the first set of tests presented in figure 2, the environment is free of obstacles and the terminal constraint is not used. From its initial pose, the camera has to reach the desired one, represented by the red triangle in the figures. Three different configurations are evaluated (see table 1): Ω_1 with $N_p = 40$ and $N_c = 1$ (figures 2.1, 2.2, 2.3 and 2.4), Ω_2 with $N_p = 40$ and $N_c = 2$ (figures 2.5, 2.6, 2.7 and 2.8), and Ω_3 with $N_p = 40$ and $N_c = 40$ (figures 2.9, 2.10, 2.11 and 2.12). With Ω_1, the solver does not succeed in finding a sequence of control inputs allowing to reach the desired pose from the initial one (Fig. 2.1). The robot moves using the obtained control inputs (Fig. 2.2 and 2.3) and ends up in a local minima (Fig. 2.4). With Ω_2, the desired pose is not reached by the predicted state neither from the initial pose (Fig. 2.5), nor during the first steps (Fig. 2.6). However, after a couple of steps, the solver finds a sequence of control inputs reaching the desired pose (Fig. 2.7), allowing the system to achieve the navigation task (Fig. 2.8). Finally, with Ω_3, the computed sequence of control inputs allows reaching the desired pose from the initial pose (Fig. 2.9) and during the whole navigation (Fig. 2.10 and 2.11). Thus, the camera is driven to the desired location (Fig. 2.12). For the three configurations, the prediction horizon $N_p = 40$ is large enough to be able to reach the desired pose from the initial one. However, when $N_c = 1$, the solver has to find the unique sequence of control inputs achieving the task. In the presence of local minima, it is very challenging for a local solver such as SQP to find this unique solution. By increasing N_c, we increase the number of sequences achieving the task. Thus, the solver will more likely find one of these sequences. Another solution consists in changing the initial values of the control in order to start the minimization close to a desired minima. This technique is known as warm start Grüne and Pannek (2017).

For the second set of simulations, a terminal constraint is included in the VPC scheme. This constraint is used to check and to guarantee that the control vectors allow to reach a neighborhood of the desired pose. For the initial configuration, if the solver cannot find a solution not violating the terminal constraint, it is then necessary to modify the controller parameters. In this simulation, we use the configuration Ω_4 with $N_p = 40$, $N_c = 20$ and the terminal constraint is activated (see table 1). As it can be seen in figures 3.1 and 3.2, the predicted states always end in the neighborhood of the desired pose. Thus, the visual features converge towards their desired values (Fig. 3.3) and the control variables stay within the defined boundaries (Fig. 3.4).

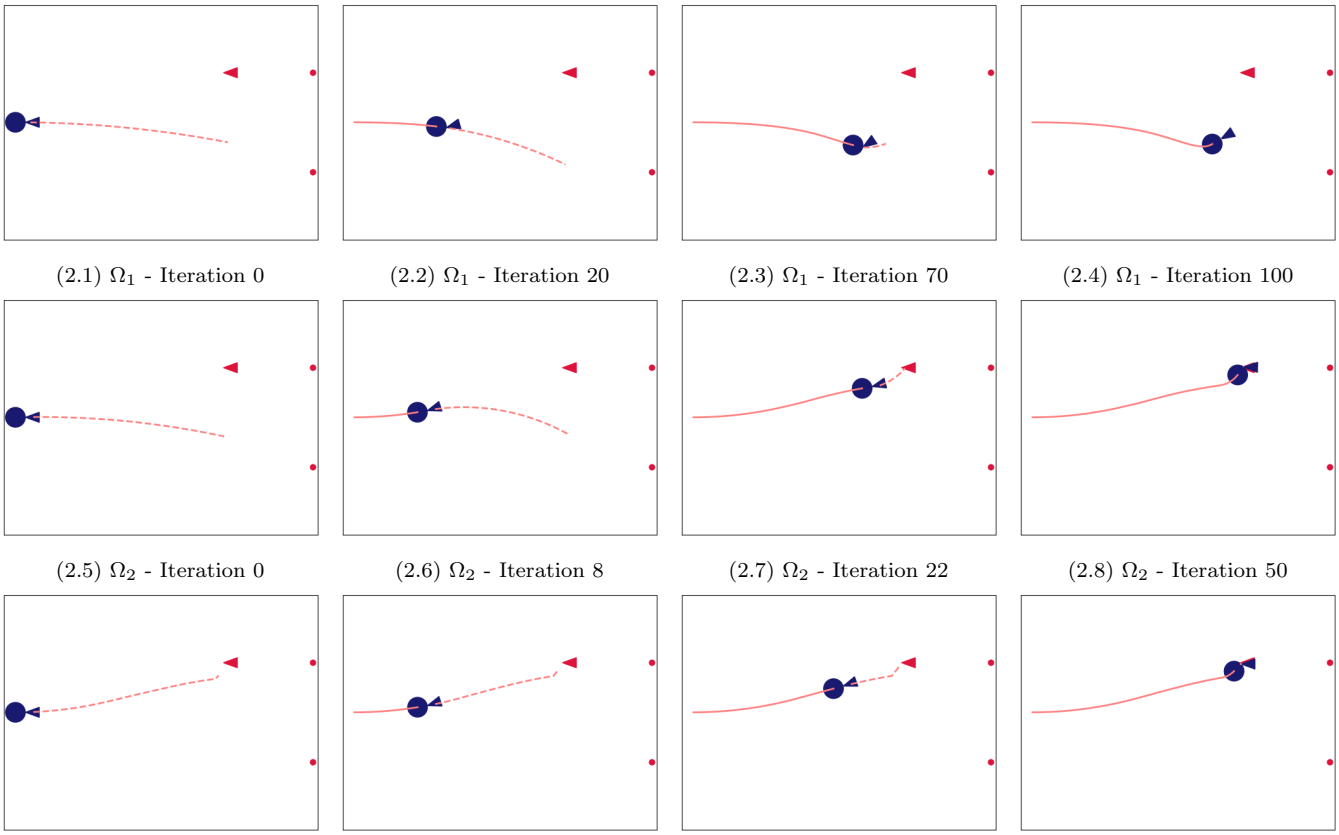
The third set of simulations aims at showing an example of the occultation constraint use. We consider two configurations (see table 1): Ω_5 where $N_p = N_c = 40$ and no occultation constraint is used, and Ω_6 where $N_p = N_c = 40$ and an occultation constraint is added. Figures 4.1 and 4.2 represent the nominal case without an occultation constraint. Figures 4.1 and 4.2 show the new behavior of the robot when an occultation constraint is added. Indeed, one can see in figure 4.2 that the visual features do not overpass the boundary represented by the dotted red line.

For the fourth set of simulations, two obstacles are included in the environment. The distance between the centers of the robot and the obstacles is known at any time. The safety distance d_s is setup to 0.1 m. Two different configurations are evaluated (see table 1): Ω_7 with $N_p = 60$ and $N_c = 60$ (figures 5.1, 5.2, 5.3 and 5.4), and Ω_8 with $N_p = 10$ and $N_c = 10$ (figures 5.5, 5.6, 5.7 and 5.8). With Ω_7, the prediction horizon is sufficiently large and the control horizon offers enough flexibility for the solver to compute a sequence of control inputs reaching the neighborhood of the desired state from the initial position (Fig. 5.1) while avoiding the obstacles. During the rest of the navigation (Fig. 5.2) the robot follows a path similar to the initial one and finally achieve the task. Indeed, the visual features converge towards the desired values (Fig. 5.3). With Ω_8, the prediction horizon is too short to obtain predicted states reaching the desired pose from the initial one (Fig. 5.5). The terminal constraint is violated and the convergence is not guaranteed. However, the sequence of control inputs computed by the solver at each iteration allows the robot achieving task while avoiding the obstacles (Fig. 5.6). The visual features converge towards the desired values (Fig. 5.7). Because of the shorter prediction horizon in Ω_8 than in Ω_7, the robot less anticipates the obstacles. Indeed, it can be seen in figure 5.8, that the robot has to stop moving forward a couple of times (around iterations 25, 35 and 50), to orientate itself and avoid the obstacles. In figure 5.4, it only happens once (around iteration 30), showing a greater anticipation capacity on the part of the robot.

For the last set of simulations, the two obstacles remain but the robot sensing range is limited to 1 meter. Two different configurations are evaluated (see table 1): Ω_9 with $N_p = 10$ and $N_c = 10$ (figures 6.1, 6.2, 6.3 and 6.4), and Ω_{10} with $N_p = 40$ and $N_c = 40$ (figures 6.5, 6.6, 6.7 and 6.8). With Ω_9, the solver cannot comply with the terminal constraint from the initial pose and the convergence is not guaranteed (Fig. 6.1). However, the robot manages to drive towards the goal while discovering and avoiding the obstacles. When a new obstacle is detected (Fig. 6.2), the solver modifies the sequence of control inputs in order to avoid it (Fig. 6.3). Thus, it successfully achieves the navigation task 6.4). With Ω_{10}, the solver complies with the terminal constraint from the initial pose and the convergence is guaranteed (Fig. 6.5). However, one of the obstacles is too far to be detected and the predicted path passes through it. When the robot detects the obstacle (Fig. 6.6), the predicted path is modified and the robot avoids the obstacle (Fig. 6.7). Thus, the navigation task is achieved (Fig. 6.8) while guaranteeing the convergence and the avoidance of the obstacles.

Configuration	N_p	N_c	Terminal constraint	Occultation constraint	Obstacle constraint	Obstacles	Sensor range
Ω_1	40	1	No	No	No	0	0
Ω_2	40	2	No	No	No	0	0
Ω_3	40	40	No	No	No	0	0
Ω_4	40	20	Yes	No	No	0	0
Ω_5	40	40	Yes	No	Yes	1	∞
Ω_6	40	40	Yes	Yes	Yes	1	∞
Ω_7	60	60	Yes	No	Yes	2	∞
Ω_8	10	10	Yes	No	Yes	2	∞
Ω_9	10	10	Yes	No	Yes	2	1
Ω_{10}	40	40	Yes	No	Yes	2	1

Table 1.



(2.1) Ω_1 - Iteration 0 (2.2) Ω_1 - Iteration 20 (2.3) Ω_1 - Iteration 70 (2.4) Ω_1 - Iteration 100

(2.5) Ω_2 - Iteration 0 (2.6) Ω_2 - Iteration 8 (2.7) Ω_2 - Iteration 22 (2.8) Ω_2 - Iteration 50

(2.9) Ω_3 - Iteration 0 (2.10) Ω_3 - Iteration 8 (2.11) Ω_3 - Iteration 18 (2.12) Ω_3 - Iteration 50

Figure 2. Navigation in a free environment without terminal constraint

6. CONCLUSION

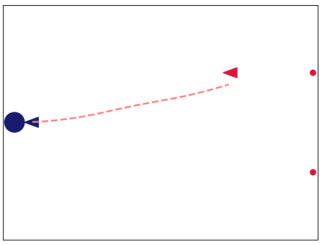
In this work we have presented the Visual Predictive Control framework for a mobile robot navigating in a cluttered environment. The control scheme relies on visual servoing coupled with model predictive control allows driving the camera towards a given pose while avoiding occultations of the landmark and collisions with the obstacles. Numerous results obtained in simulations show the impact of some of the parameters and highlight the efficiency of the approach. However, the presented results do not take into account the time required by the solver to minimize the cost function. Indeed, among the challenges arising when relying on a VPC scheme to navigate, the computing time seems to be a major one. Indeed, using large values for N_p and N_c leads to heavy computations not compatible with a short sampling time for a real time control in a dynamic environment. It seems then mandatory to develop a method allowing to quickly solve the optimization problem for large values for N_p and N_c.

REFERENCES

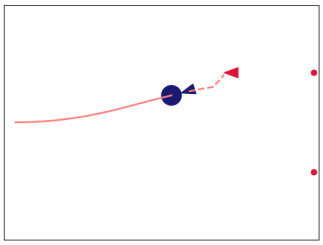
Allibert, G., Courtial, E., and Chaumette, F. (2010). Predictive control for constrained image-based visual servoing. *IEEE Trans. on Robotics*, 26(5), 933–939.

Assa, A. and Janabi-Sharifi, F. (2014). Robust model predictive control for visual servoing. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2715–2720.

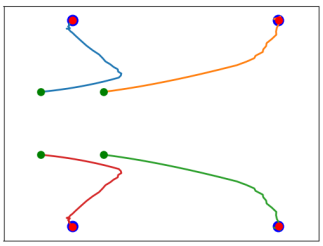
Chaumette, F. (1998). Potential problems of stability and convergence in image-based and position-based visual servoing. In D. Kriegman, G.. Hager, and A. Morse (eds.), *The Confluence of Vision and Control*, 66–78. LNCIS Series, No 237, Springer-Verlag.



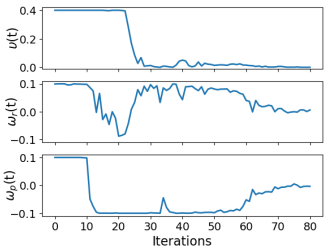
(3.1) Ω_4 - Iteration 0



(3.2) Ω_4 - Iteration 20

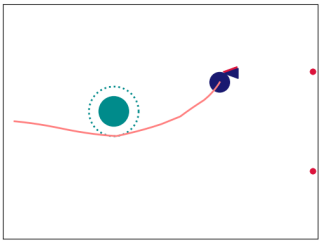


(3.3) Ω_4 - Visual features

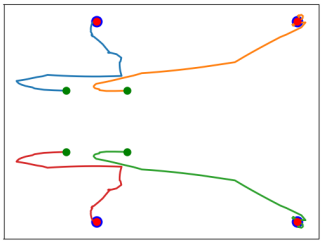


(3.4) Ω_4 - Control inputs

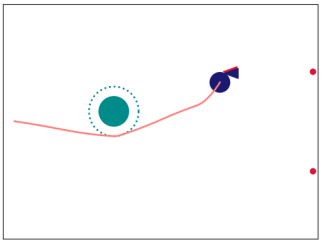
Figure 3. Navigation in a free environment with terminal constraint



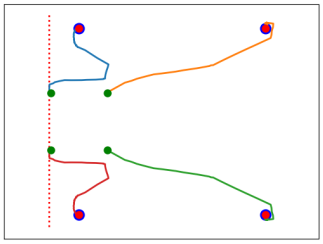
(4.1) Ω_5 without occultation constraint



(4.2) Ω_5 without occultation constraint - Visual features



(4.3) Ω_6 with occultation constraint



(4.4) Ω_6 with occultation constraint - Visual features

Figure 4. Navigation in a cluttered environment with and without occultation constraint

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

Chaumette, F. and Hutchinson, S. (2006). Visual servo control, part 1 : Basic approaches. *Robotics and Automation Mag.*, 13(4).

Copot, C., Lazar, C., and Burlacu, A. (2012). Predictive control of nonlinear visual servoing systems using image moments. *IET control theory & applications*, 6(10), 1486–1496.

Corke, P. and Hutchinson, S. (2001). A new partitioned approach to image-based visual servo control. *Robotics and Automation, IEEE Transactions on*, 17(4), 507 – 515.

Durand-Petiteville, A. (2012). *Navigation référencée multi-capteurs d'un robot mobile en environnement encombré*. Ph.D. thesis, Université Paul Sabatier-Toulouse III.

Flécher, E., Durand-Petiteville, A., Cadenat, V., and Sentenac, T. (2019). Visual predictive control of robotic arms with overlapping workspace. In *16th International Conference on Informatics in Control, Automation and Robotics (ICINCO 2019)*, 130–137.

Folio, D. and Cadenat, V. (2008). *Treating Image Loss by using the Vision/Motion Link: A Generic Framework*, chapter 4. IN-TECH.

Grüne, L. and Pannek, J. (2017). Nonlinear model predictive control. In *Nonlinear Model Predictive Control*, 45–69. Springer.

Hajiloo, A., Keshmiri, M., Xie, W., and Wang, T. (2016). Robust online model predictive control for a constrained image-based visual servoing. *IEEE Transactions on Industrial Electronics*, 63(4), 2242–2250. doi:10.1109/TIE.2015.2510505.

Heshmati-alamdari, S., Karavas, G.K., Eqtami, A., Drossakis, M., and Kyriakopoulos, K.J. (2014). Robustness analysis of model predictive control for constrained image-based visual servoing. In *2014 IEEE Int. Conf. on Robotics and Automation*, 4469–4474.

Johnson, S.G. (2020). The nlopt nonlinear-optimization package. URL `http://github.com/stevengj/nlopt`.

Ke, F., Li, Z., Xiao, H., and Zhang, X. (2017). Visual servoing of constrained mobile robots based on model predictive control. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 47(7), 1428–1438. doi:10.1109/TSMC.2016.2616486.

Lazar, C., Burlacu, A., and Copot, C. (2012). 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)*, 1458–1464. IEEE.

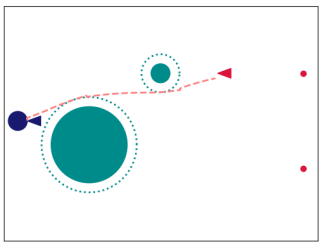
Lee, D., Lim, H., and Kim, H.J. (2011). Obstacle avoidance using image-based visual servoing integrated with nonlinear model predictive control. In *2011 50th IEEE Conference on Decision and Control and European Control Conference*, 5689–5694. IEEE.

Malis, E., Chaumette, F., and Boudet, S. (1999). 2d 1/2 visual servoing. *IEEE Trans. on Rob. and Automation*, 15(2), 234–246.

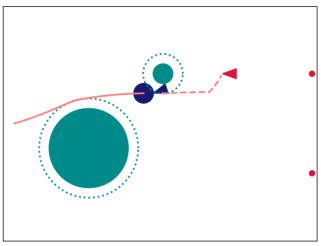
Mcfadyen, A., Corke, P., and Mejias, L. (2014). Visual predictive control of spiral motion. *IEEE Transactions on Robotics*, 30(6), 1441–1454.

Qiu, Z., Hu, S., and Liang, X. (2019). Model predictive control for uncalibrated and constrained image-based visual servoing without joint velocity measurements. *IEEE Access*, 7, 73540–73554.

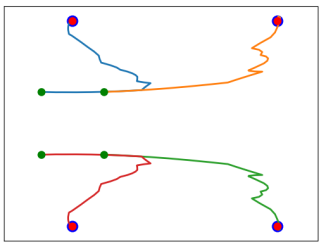
Wang, T., Xie, W., Liu, G., and Zhao, Y. (2012). Quasi-min-max model predictive control for image-based visual servoing. In *2012 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*, 98–103. doi:10.1109/AIM.2012.6265955.



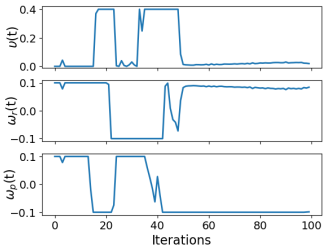
(5.1) Ω_7 - Iteration 0



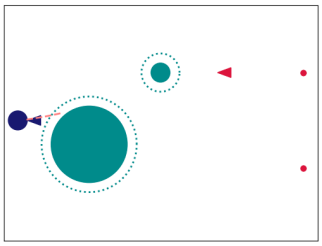
(5.2) Ω_7 - Iteration 42



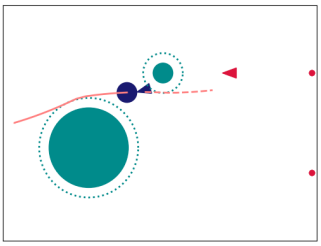
(5.3) Ω_7 - Visual features



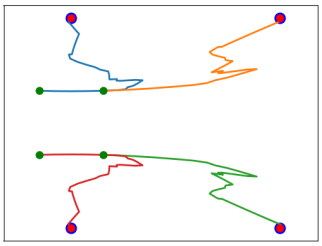
(5.4) Ω_7 - Control inputs



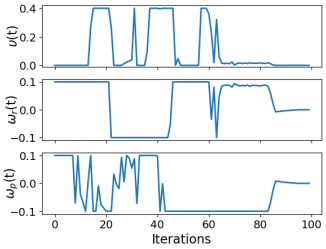
(5.5) Ω_8 - Iteration 0



(5.6) Ω_8 - Iteration 42

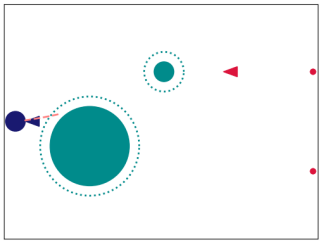


(5.7) Ω_8 - Visual features

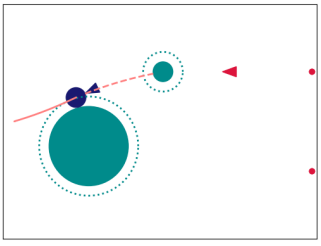


(5.8) Ω_8 - Control inputs

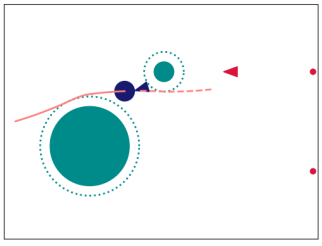
Figure 5. Navigation in a cluttered environment with an infinite sensor range



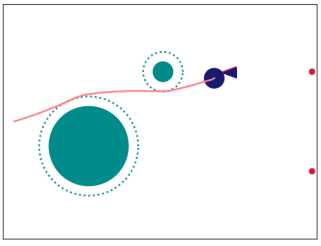
(6.1) Ω_9 - Iteration 0



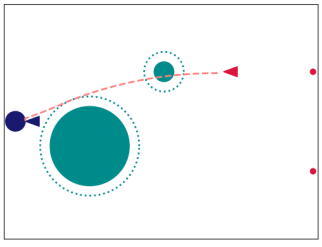
(6.2) Ω_9 - Iteration 23



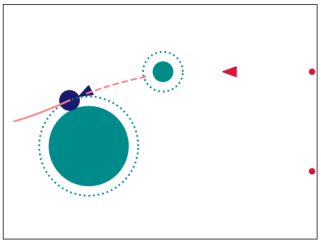
(6.3) Ω_9 - Iteration 42



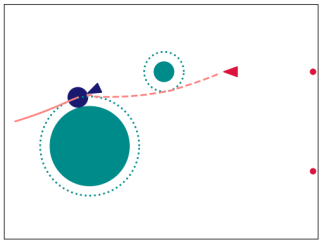
(6.4) Ω_9 - Iteration 100



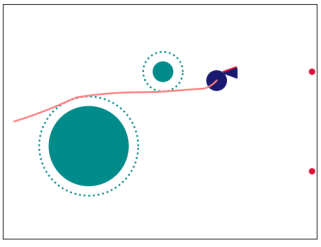
(6.5) Ω_{10} - Iteration 0



(6.6) Ω_{10} - Iteration 22



(6.7) Ω_{10} - Iteration 23



(6.8) Ω_{10} - Iteration 100

Figure 6. Navigation in a cluttered environment with a 1 meter sensor range