Visual Servoing Using Model Predictive Control to Assist Multiple Trajectory Tracking

Nicolas Cazy, Pierre-Brice Wieber, Paolo Robuffo Giordano, François Chaumette

Visual Servoing Using Model Predictive Control to Assist Multiple Trajectory Tracking

Nicolas Cazy, Pierre-Brice Wieber, Paolo Robuffo Giordano, and François Chaumette

***Abstract*— We propose in this paper a new active perception scheme based on Model Predictive Control under constraints for generating a sequence of visual servoing tasks. The proposed control scheme is used to compute the motion of a camera whose task is to successively observe a set of robots for measuring their position and improving the accuracy of their localization. This method is based on the prediction of an uncertainty model (due to actuation and measurement noise) for determining which robot has to be observed by the camera. Simulation results are presented for validating the approach.**

I. INTRODUCTION

Visual servoing is a very effective method for controling the robot motion for realizing tasks of all sorts using feedback from visual sensors [1]. However, an important constraint of using cameras is their limited field of view. This constraint is usually mitigated by imposing that the target features are always kept within the field of view. Avoiding that visual features leave the camera field of view is taken into account in [2]. Various works on visual trajectory planning that take into account occlusions and feature loss avoidance have also been realized [3]–[7].

However, strictly imposing the visibility of all targets is not always possible nor desirable. One way to perform visual servoing when the necessary visual information is missing is to simply predict how this information would evolve when no measurement is available [8], [9]. This solution is limited, however, since the mismatch between the prediction model and the real measurements is bound to increase with time. In this respect, we propose in this paper to monitor this discrepancy for guaranteeing that it stays between predefined bounds. This requires ensuring that the necessary visual information can be collected regularly, when needed. For this, we consider a classical eye-in-hand setup where the camera view can be modified at will. We model the drifts between real and modeled positions and predict their evolution over a future time window. We set bounds on these drifts, and ensure that they will not be exceeded thanks to new camera measurements, using a Model Predictive Control scheme. This control strategy has been already used for visual servoing handling constraints for different applications in [10]–[15].

N. Cazy is with Inria at Irisa, Campus de Beaulieu, 35042 Rennes Cedex, France nicolas.cazy@inria.fr

P.-B. Wieber is with Inria at Inria Rhône-Alpes, France pierre-brice.wieber@inria.fr

P. Robuffo Giordano is with the CNRS at Irisa and Inria, Campus de Beaulieu, 35042 Rennes Cedex, France paolo.robuffo_giordano@irisa.fr

F. Chaumette is with Inria at Irisa, Campus de Beaulieu, 35042 Rennes Cedex, France francois.chaumette@inria.fr

As an example, we propose to consider one camera carried by a UAV hovering over ground at a constant height. This UAV is subject to a drift between its real and its modeled position. We consider that the only way to correct the UAV/camera position consists in observing a landmark on the ground whose position is known. We also consider a set of mobile ground robots which have to follow some planned trajectories. These trajectories are far from each other and from the landmark so that the robots and the landmark cannot be seen by the flying camera at the same time. The UAV will then have to decide when to look at which robot and when to look at the landmark for making sure that the visual information, necessary for the ground robots to realize their task, is regularly collected.

A link can be established between our method and the self-triggered control in the sense that the quantity of information sent by the camera to the ground robots is intentionally reduced [16]. Indeed at each sampling instant the knowledge of the UAV and of the robots dynamics allows to predict the camera velocity without any measurement. Furthermore, our proposed method can be considered as an instance of active perception [17]. As discussed in [18], the proposed control scheme has to decide online the control inputs and sensor measurement strategies in order to minimize the uncertainty in the ground robot localization. Multiple works have already been conducted in this sense: for instance, in the context of object search, a mobile robot generates its path thanks to a web-based knowledge which delivers a customized execution plan in [19]. A robot path is determined on current knowledge of the object location encoded by a probability distribution computed during the search process in [20]. Object recognition is obtained by predicting the optimal viewpoints in [21], [22]. Finally, a grasping method of deformable object by a robot arm using depth information is proposed in [23]. In this paper, we exploit a model of the uncertainty evolution of the ground robots and UAV position to determine the sequence of camera visual servoing tasks to correct model drifts one after the other. The principle of active perception is thus focused on the control of the camera such that mobile ground robots and UAV correct their drift.

Section II starts by describing the tasks assigned to mobile ground robots, as well as the model errors which can be corrected by camera measures. Section III reviews the considered Model Predictive Control scheme which is used to realize the desired camera task. Simulation results are presented in Sect. IV. Finally, Sect. V concludes the paper and discusses about future works.

II. Modeling

In this paper, we consider N_r mobile robots that have to follow some desired trajectories on the ground, one camera embedded on a UAV stabilized at a constant height, and a fixed landmark on the ground whose position is known in the world frame. The robot and camera are modeled as kinematic systems with velocity inputs. We also assume that each real position drifts from its respective model because of non-idealities such as actuation noise. This noise represents the fact that the robots and the UAV/camera do not react perfectly to their control schemes due to, e.g., modeling errors or delays. The UAV and camera are modeled as simple kinematic systems in order to focus on the the characteristics of the model predictive control scheme presented in the Section III. Future extensions of this work will consider more realistic dynamical models for the considered robots.

In our settings, the only way to rectify these drifts is to correct the modeled positions by measuring the robots (for the robot models) and landmark (for the camera model) positions. The problem is that, in general, only one robot or the landmark can be seen by the camera at once. Moreover, we choose to force the camera to stay focused on one robot or on the landmark as long as the other models do not need to be corrected. This is meant to minimize the camera total displacement. To determine which model should be corrected, the uncertainty of each position is propagated in order to quantify the amount of drifts. According to these uncertainties, the camera controller is able to manage the priority of which model must be corrected.

A. Representation

We consider that the camera and the robots have 2 DOFs each, namely their Cartesian coordinates on a plane. More precisely, we set $\boldsymbol{c} = \left(c_x\,,\,c_y\right)$ and $\boldsymbol{r}_i = \left(r_{ix}\,,\,r_{iy}\right)$ respectively as camera and robot i $(\forall i = 1, ..., N_r)$ coordinates in a world frame (in red and orange on Fig. 1). We set also $\boldsymbol{m} = \left(m_x\,,\,m_y\right)$ as the motionless landmark used for the camera model correction, whose coordinates in the world frame are supposed known (in black in Fig. 1).

We set also $\boldsymbol{c}_m$ and $\boldsymbol{r}_{im}$ as the *modeled* coordinates of the camera and the robots (in blue on Fig. 1). These models are introduced for two reasons: the first one lies in the structure of the employed model predictive control scheme (see Section III) in wich future positions of the robots and the camera are determined using a model. The second one lies in the fact that, when the robots and the landmark are outside the camera field of view, no measurement is available to determine precisely the robot and camera position. Thus, using a model allows to cope with the lack of direct measurements and approximate the unmeasurable positions. Obviously, when a measurement becomes available, the corresponding model can be corrected accordingly.

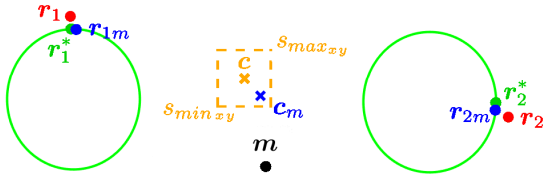


Fig. 1: Camera with limited field of view monitoring the trajectory of two robots on the ground (top view).

B. Dynamics

The dynamics of camera and robots is assumed to evolve according to this simple kinematic model:

$$\left\{\begin{array}{rcl} \dot{\boldsymbol{c}} & = & \boldsymbol{u}_c + n_c \\ \dot{\boldsymbol{r}}_i & = & \boldsymbol{u}_i + n_i \end{array}\right. ; \tag{1}$$

where:

- the velocities applied to the camera allow it to move parallel to the ground: $\boldsymbol{u}_c = \left(u_{cx}\,,\,u_{cy}\right)$.
- the velocities applied to each robot allow them to move on the ground: $\boldsymbol{u}_i = \left(u_{ix}\,,\,u_{iy}\right)$.
- n_c and n_i are additive Gaussian noises applied to the camera and robots velocities:

$$n \sim \mathcal{N}(0, \sigma^2), \tag{2}$$

with σ^2 being the variance. They define the error between the applied and the actual velocity.

As discussed in Sect. II-A, the modeled positions of camera and robots are the only information always available. Following (1), the dynamics of the modeled coordinates are defined in a discrete time form at each instant t_k as:

$$\left\{\begin{array}{rcl} \boldsymbol{c}_m(t_{k+1}) & = & \boldsymbol{c}_m(t_k) + \tau \boldsymbol{u}_c(t_k) \\ \boldsymbol{r}_{im}(t_{k+1}) & = & \boldsymbol{r}_{im}(t_k) + \tau \boldsymbol{u}_i(t_k) \end{array}\right. \tag{3}$$

with τ being the sampling time. From (3), the modeled coordinates can be predicted over a prediction horizon N as:

$$\left\{\begin{array}{rcl} \overrightarrow{\boldsymbol{c}_m} & = & \mathbf{A}\boldsymbol{c}_m(t_k) + \mathbf{B}\overrightarrow{\boldsymbol{u}_c} \\ \overrightarrow{\boldsymbol{r}_{1m}} & = & \mathbf{A}\boldsymbol{r}_{im}(t_k) + \mathbf{B}\overrightarrow{\boldsymbol{u}_i} \end{array}\right. \tag{4}$$

where :

- the sets of the camera and robot positions are:

$$\left\{\begin{array}{rcl} \overrightarrow{\boldsymbol{c}_m} & = & \left(\boldsymbol{c}_m(t_{k+1})\,,\,\ldots\,,\,\boldsymbol{c}_m(t_{k+N})\right) \in \mathbb{R}^{2N\times 1} \\ \overrightarrow{\boldsymbol{r}_{im}} & = & \left(\boldsymbol{r}_{im}(t_{k+1})\,,\,\ldots\,,\,\boldsymbol{r}_{im}(t_{k+N})\right) \in \mathbb{R}^{2N\times 1} \end{array}\right. , \tag{5}$$

- the sets of the camera and robot inputs are:

$$\left\{\begin{array}{rcl} \overrightarrow{\boldsymbol{u}_c} & = & \left(\boldsymbol{u}_c(t_k)\,,\,\ldots\,,\,\boldsymbol{u}_c(t_{k+N-1})\right) \in \mathbb{R}^{2N\times 1} \\ \overrightarrow{\boldsymbol{u}_i} & = & \left(\boldsymbol{u}_i(t_k)\,,\,\ldots\,,\,\boldsymbol{u}_i(t_{k+N-1})\right) \in \mathbb{R}^{2N\times 1} \end{array}\right. , \tag{6}$$

- and

$$\mathbf{A} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ \vdots & \vdots \\ 1 & 0 \\ 0 & 1 \end{pmatrix} \in \mathbb{R}^{2N \times 2},$$

$$\mathbf{B} = \begin{pmatrix} \tau & 0 & 0 & 0 & \dots & 0 \\ 0 & \tau & 0 & 0 & \dots & 0 \\ \tau & 0 & \tau & 0 & \dots & 0 \\ 0 & \tau & 0 & \tau & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & \tau & 0 & \tau & \dots & \tau \end{pmatrix} \in \mathbb{R}^{2N \times 2N}. \quad (7)$$

This dynamic over a future horizon is used for the model predictive control scheme (see Section III).

We now describe how the velocities applied to the robots are computed to allow them to follow their desired trajectory.

C. Robot control

The desired trajectories to be followed by the robots are represented by a time-varying desired $\boldsymbol{r}_i^*(t)$ (in green on Fig. 1). The robots are controlled to track their desired trajectories by applying the simple control scheme:

$$\boldsymbol{u}_i = k(\boldsymbol{r}_i^* - \boldsymbol{r}_{im}) \quad (8)$$

with a scalar gain $k > 0$. It therefore appears that the drift between the real and modeled robot coordinates will also induce a drift between the real and the desired robot position. We see now how to correct the modeled position thanks to camera measurements.

D. Models correction

The camera coordinates in the world frame are given by:

$$\boldsymbol{s}_c = \boldsymbol{c} - \boldsymbol{m}, \quad (9)$$

while the robot coordinates in the camera frame are:

$$\boldsymbol{s}_i = \boldsymbol{r}_i - \boldsymbol{c}. \quad (10)$$

However because of image processing errors, camera measurements are subject to an additive noise n_s with same form as (2). This implies a slight error for each measurement $\bar{\boldsymbol{s}_c}$ and $\bar{\boldsymbol{s}_i}$ as

$$\begin{cases} \bar{\boldsymbol{s}_c} = \boldsymbol{s}_c + n_s \\ \bar{\boldsymbol{s}_i} = \boldsymbol{s}_i + n_s \end{cases}. \quad (11)$$

These coordinates are available as measurement only when the landmark and the robots appear in the camera field of view delimited by the visibility range $\boldsymbol{s}_{min} = \left(\boldsymbol{s}_{min_x}, \boldsymbol{s}_{min_y}\right)$ and $\boldsymbol{s}_{max} = \left(\boldsymbol{s}_{max_x}, \boldsymbol{s}_{max_y}\right)$ (orange lines in Fig. 1). A measurement allows obtaining either the camera model coordinates $\boldsymbol{c}_m$ or one robot model coordinates $\boldsymbol{r}_{im}$. Thus, using (9), (10), and (11), we can write:

$$\begin{cases} \boldsymbol{c}_m \leftarrow \bar{\boldsymbol{s}_c} + \boldsymbol{m} & \text{if } \boldsymbol{s}_{min} \leqslant \bar{\boldsymbol{s}_c} \leqslant \boldsymbol{s}_{max} \\ \boldsymbol{r}_{im} \leftarrow \bar{\boldsymbol{s}_i} + \boldsymbol{c}_m & \text{if } \boldsymbol{s}_{min} \leqslant \bar{\boldsymbol{s}_i} \leqslant \boldsymbol{s}_{max} \end{cases} \quad (12)$$

The camera model drift can then be corrected by the landmark observation, while the robots and therefore also their trajectories are corrected by robots observation.

In this section, we discussed the modeling about camera and robots position, and highlighted the importance of obtaining a camera measurement for correcting the camera and robot models (and, thus, improving the robot tracking performance). The main remaining issue is to develop an appropriate control scheme for the camera moving to the robots or to the landmark to correct the position models, which is the goal of the next section.

III. MODEL PREDICTIVE CONTROL SCHEME

In this section we design a model predictive control strategy for allowing the camera to move towards either the landmark or one of the robots. Our aim is to force the camera to switch from one target to another one as a function of the evolution of the position uncertainties. These uncertainties are computed from a model characterizing how the drifts evolve over time. This strategy can be considered as a particular instance of *active perception*, since the camera motion must be optimized online in order to gather information about the scene.

The use of this approach is justified by the fact that the camera cannot move instantaneously from one point to another. It is therefore necessary to anticipate the camera and robot motion exploiting the evolution of position uncertainties over a prediction horizon. In order to develop this control strategy, we need to define the overall system dynamics (which has already been done in Sect. II-B), a suitable cost function able to capture our control objective, and the constraints imposed to the system. A proper definition of these constraints provides the desired behavior of the camera.

A. Cost function

The cost function to be minimized is given by:

$$\min_{\overrightarrow{u_c} \in \mathbf{K}} J(\overrightarrow{\boldsymbol{u}_c}) \quad (13)$$

where $\mathbf{K}$ is the constraint domain described in the next section III-B, and

$$J(\overrightarrow{\boldsymbol{u}_c}) = \begin{array}{r} (\overrightarrow{\boldsymbol{s}_{cm}} - \overrightarrow{\boldsymbol{s}_{cm}^*})^T (\overrightarrow{\boldsymbol{s}_{cm}} - \overrightarrow{\boldsymbol{s}_{cm}^*}) \\ + \sum (\overrightarrow{\boldsymbol{s}_{im}} - \overrightarrow{\boldsymbol{s}_{im}^*})^T (\overrightarrow{\boldsymbol{s}_{im}} - \overrightarrow{\boldsymbol{s}_{im}^*}) \end{array}, \quad (14)$$

where $\overrightarrow{\boldsymbol{s}_{cm}}$ and $\overrightarrow{\boldsymbol{s}_{im}}$ are the sets of the corresponding models of $\overrightarrow{\boldsymbol{s}_c}$ and $\overrightarrow{\boldsymbol{s}_i}$ defined as:

$$\begin{cases} \boldsymbol{s}_{cm} = \boldsymbol{c}_m - \boldsymbol{m} \\ \boldsymbol{s}_{im} = \boldsymbol{r}_{im} - \boldsymbol{c}_m \end{cases} \quad (15)$$

Moreover, $\overrightarrow{\boldsymbol{s}_{cm}^*}$ and $\overrightarrow{\boldsymbol{s}_{im}^*}$ are the sets of the desired position of the landmark and the robot models in the camera frame along the prediction horizon. To obtain the landmark and robots in the center of the camera field of view, we set $\boldsymbol{s}_{cm}^*$ and $\boldsymbol{s}_{im}^*$ always null.

This cost function has been designed so that the camera keeps at equal distance from the landmark and the robots. However, this distance will be strongly disturbed by the constraints (21). Namely, these constraints will force the camera to move towards the landmark or one robot (as it will be explained in Sect. III-B.2.b).

B. Constraints

A great advantage of model predictive control is the possibility to handle constraints in a principled way. We will use this characteristic to influence the camera motion.

1) Camera inputs constraints: First, constraints on the control input are included in the domain $\mathbf{K}$ as:

$$\boldsymbol{u}_c^- \leqslant \boldsymbol{u}_c \leqslant \boldsymbol{u}_c^+ \tag{16}$$

to ensure the camera velocity will not exceed its physical limitations, with $\boldsymbol{u}_c^- = (u_{cx}^-,\ u_{cy}^-)$ and $\boldsymbol{u}_c^+ = (u_{cx}^+,\ u_{cy}^+)$. Obviously, these constraints are applied for all the set of inputs $\overrightarrow{\boldsymbol{u}_c}$.

2) Relative positions constraints: Secondly, the propagation of the positions uncertainty over a future time is modeled. This propagation is used to select which model needs to be corrected. Then, to force the camera to move to the selected model, constraints on the camera position relative to the landmark and to the robot positions are included in the domain $\mathbf{K}$.

a) Uncertainties: As introduced in Sect. II-B, the velocity applied to camera and robots is subject to noise in (1), which causes a drift between the real and the modeled position. Knowing the variance of that noise, it is possible to model the drift propagation over time. For this, we introduce the position uncertainty p_j where $j = 0, ..., N_r$, and $j = 0$ represents the camera.

Two situations are involved to determine the uncertainty propagation. Firstly, when a model is corrected thanks to an observation by the camera, a minimum value can be assigned to its uncertainty:

$$p_j(t_k) = \sigma_s^2 \quad \text{if} \quad \boldsymbol{s}_{min} \leqslant \bar{\boldsymbol{s}_j}(t_k) \leqslant \boldsymbol{s}_{max}, \tag{17}$$

which corresponds to the variance σ_s^2 of the measurement error (11). Secondly when a model is not corrected, the propagation of the uncertainty, exlploiting the dynamics (1), is defined as [24]:

$$p_j(t_{k+1}) = p_j(t_k) + \tau^2\sigma_j^2. \tag{18}$$

Propagating over the future horizon N, we obtain:

$$\left\{ \begin{array}{lcl} p_j(t_{k+N}) & = & \sigma_s^2 \ \text{ if } \ \boldsymbol{s}_{min} \leqslant \bar{\boldsymbol{s}_j}(t_k) \leqslant \boldsymbol{s}_{max} \\ p_j(t_{k+N}) & = & p_j(t_k) + N\tau^2\sigma_j^2 \ \text{ otherwise} \end{array} \right. . \tag{19}$$

By predicting the position uncertainty over the prediction horizon, we can define which model has to be corrected. For this, a condition C_j is associated to each uncertainty as:

$$\begin{array}{l} C_j \Leftrightarrow \\ \left(\begin{array}{lcl} & p_j(t_{k+N}) & \geqslant \ p_{max_j} \\ \text{and} & p_j(t_{k+N}) - p_{max_j} & > \ p_l(t_{k+N}) - p_{max_l} \end{array} \right) \\ \text{or} \\ \left(\begin{array}{lcl} & p_j(t_{k+N}) & < \ p_{max_j} \\ \text{and} & p_j(t_{k+N}) & < \ p_l(t_{k+N}) \end{array} \right) \\ \forall l \neq j,\ \forall j = 0, ..., N_r \end{array} \tag{20}$$

The uncertainty threshold p_{max_j}, which corresponds to the uncertainty value to not exceeded for allowing the correction of model j will be described in Sect. III-B.2.c. More precisely:

- If the uncertainty p_j is larger than p_{max_j} over the prediction horizon and if the difference between p_j and p_{max_j} is the largest among all the other uncertainties, the condition is held. This ensures that the model whose uncertainty exceeds the most the threshold over the prediction horizon has to be corrected.
- Or if all uncertainties are lower than their threshold, and if the uncertainty p_j is lower than all the other ones over the prediction horizon, the condition is held. This ensures that either the landmark or one of the robots is always corrected even when no uncertainty exceeds its threshold over the prediction horizon. This choice minimizes the camera displacement since it will remain focused on the same target as long as no other target needs to be corrected.

The prediction of uncertainties allows the control strategy to determine which model should be corrected by the observations of the camera. We now describe how to inject this information in the model predictive control.

b) Tolerances: Tolerances represent the set of distances $\overrightarrow{\delta_j}$ between the limits of the camera field of view and the modeled coordinates of the landmark/robot in the camera frame. The constraints that they define are added to the domain $\mathbf{K}$:

$$\boldsymbol{s}_{min} - \overrightarrow{\delta_j} \leqslant \overrightarrow{\boldsymbol{s}_{jm}} \leqslant \boldsymbol{s}_{max} + \overrightarrow{\delta_j}\ , \quad \forall j = 0, ..., N_r. \tag{21}$$

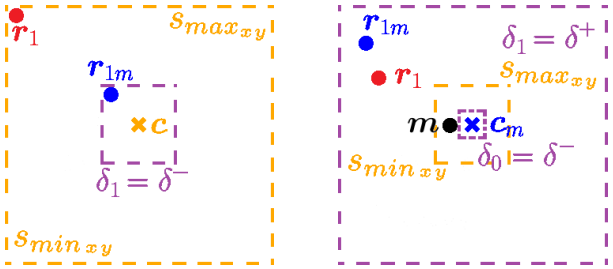
We set δ^- and δ^+ as respectively a low and a high tolerance value. They are assigned to the tolerances $\overrightarrow{\delta_j}$ following the conditions defined in (20):

$$\left\{ \begin{array}{lcll} \overrightarrow{\delta_j} & = & \delta^- & \text{if} \quad C_j \\ \overrightarrow{\delta_j} & = & \delta^+ & \text{otherwise} \end{array} \right. . \tag{22}$$

The idea consists in having a low value δ^- for moving the camera to the model that must be corrected. For instance on the Figure 2a, the tolerance δ_1 is low, the camera moves then to the first robot. However, even if δ^- is low, we cannot guarantee that the real position will be in the camera field of view at the end of the camera motion because of the drifts from the models. This is why the value δ^- is set to a negative value to allow the robot or the landmark to be in the camera field of view despite the drifts (see Figure 2a). Obviously, the camera can move to only one robot or to the landmark at a time. This is why a high value of tolerance δ^+ is assigned to the other models for ensuring that the system tolerates that they keep far from the camera field of view. Thus, on Figure 2b, while the tolerance assigned to the camera δ_0 is low, the tolerance assigned to the robot δ_1 is high, which allows the camera to move to the landmark and to keep far from the robot.

The constraints defined by (21) allow the control scheme to force the camera to move towards the landmark or one

robot when it is needed. δ^- and δ^+ are assigned according to the conditions in (20) depending on the evolution of uncertainties p_j and on the uncertainty threshold p_{maxj} that is described now.



(a) (b)

Fig. 2: Tolerance principle: The camera field of view is represented by orange dotted lines. (a) δ_1 represented in magenta is negative. The camera $\boldsymbol{c}$ retrieves the robot $\boldsymbol{r}_1$ in its field of view despite the drift from the model $\boldsymbol{r}_{1m}$. (b) δ_0 is negative, and δ_1 is high. The camera has moved to the landmark and its model is corrected, while the robot is allowed to stay far away from the field of view.

c) Uncertainty threshold: As described above, δ^- has to be negative to allow the real robot or the landmark to be in the camera field of view despite the drifts from the models. However, whatever the value chosen for δ^- the drift distance between the real and the modeled position will exceed $|\delta^-|$ after several iterations. If that would occur, the real position would be outside the camera field of view and the model would not be corrected. The solution consists to correct the model before the distance $|\delta^-|$ is reached by the drift.

For this, we define $Nmin_j$ as the number of iterations required by the drift distance to reach $|\delta^-|$ from a null value. Thus, by propagating the uncertainty p_j from the minimum value defined by (17) over $Nmin_j$ iterations we obtain the uncertainty threshold p_{max_j}, namely the uncertainty reached when the drift distance attains $|\delta^-|$ from a null value. Finally, using this threshold in (20), we ensure that the drift will be corrected before reaching $|\delta^-|$. Using (19), the uncertainty threshold p_{max_j} is given by:

$$p_{max_j} = \sigma_s^2 + Nmin_j \tau^2 \sigma_j^2. \tag{23}$$

To understand how we compute $Nmin_j$, we firstly introduce the drift distances of the camera and the robots:

$$\begin{cases} e_0 &= ||\boldsymbol{c} - \boldsymbol{c}_m|| \\ e_i &= ||\boldsymbol{r}_i - \boldsymbol{r}_{im}|| \end{cases} \forall i = 1, \dots, N_r, \tag{24}$$

with e_0 being the distance of the camera from its model, and e_i the distance between the robot i and its model. Because both drift ditances of the camera and the robot i increase together, it appears that the distance $|\delta^-|$ can be reached by the sum of these two drifts after a minimal number of iterations. As an example illustrated in Figure 3, we considere the worst case which can be considered. Because $\delta_1 = \delta^-$, the camera represented in orange moves to the robot in red while this one follows its trajectory represented in green. We can see the drifts between their real and their model positions, and that the sum $e_0 + e_1$ reachs the distance $|\delta^-|$ after $Nmin_1$ iterations.

The exact values of e_0 and e_i cannot be exactly computed, so we need to further explore the considered model of noise (2) which represents the drift evolution. As well known, it follows the normal distribution whose probability density establishes that 99.7% of noise values are distributed between $-3\sigma_j$ and $+3\sigma_j$. The drift distance can then be confidently modeled by the propagation of $3\sigma_j$ at each iteration. From this, the corresponding uncertainty is propagated as $9\sigma_j^2$ at each iteration. $Nmin_j$ is then the ration between the square of the limit distance $|\delta^-|^2$ and the uncertainty $9\sigma_j^2$.

Finally, two different $Nmin_j$ are computed. The first $Nmin_i$ is linked to the robot i. As we saw at the beginning of this paragraph, the distance $|\delta^-|$ is reached by the sum of drift distances of the camera and the robot i. The corresponding minimal number of iterations can then be quantified as:

$$Nmin_i = \frac{|\delta^-|^2}{9\tau^2(\sigma_c^2 + \sigma_i^2)}. \tag{25}$$

Finally, $Nmin_i$ is injected in (23) to compute the uncertainty threshold of the robot i.

The second $Nmin_0$ is linked to the camera. The goal is to ensure that all models of the robots can be corrected despite the camera drift. This means that we must ensure that the model of the camera is corrected as much as the model of the robot which has the largest drift. Therefore $Nmin_0$ is equal to the $Nmin_i$ with the lowest value:

$$Nmin_0 = \min(Nmin_i) \tag{26}$$

$Nmin_0$ can then be injected in (23) to compute the uncertainty threshold of the camera.

We described in this part how to compute the uncertainty thresholds p_{max_j} in order to use it in the conditions (20). We now describe how to compute properly the value of the prediction horizon N to ensure that a satisfactory behavior will always be obtained.

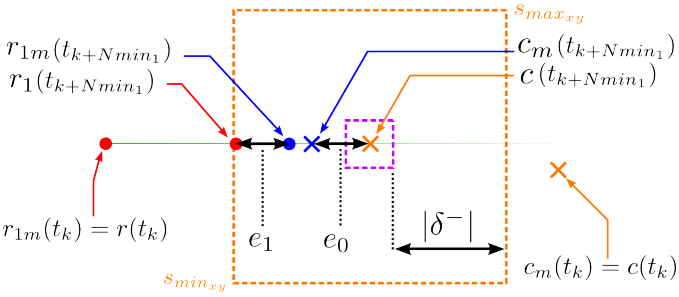


Fig. 3: The robot $\boldsymbol{r}_1$ in red follows the trajectory represented in green. The camera $\boldsymbol{c}$ and its field of view are in orange. At $t = t_k$, both models are corrected. At $t = t_{k+Nmin1}$, the robot is in the camera field of view despite the sum of errors caused by robot and camera drifts $e_0 + e_1$.

C. Prediction Horizon

To guarantee that the models can be corrected despite errors introduced by noises, we must ensure that no uncertainty exceeds its corresponding threshold. This can be completed by a suitable computation of the prediction horizon N.

The camera cannot move instantaneously from a position to another. The prediction horizon must thus be greater than or equal to the number of iterations required to complete the longest path taken by the camera according to the minimum of its maximum speed:

$$N \geqslant \frac{d_{max}}{\tau u_{cmin}} \tag{27}$$

where $u_{cmin} = \min(|u_{cx}^-|, |u_{cy}^-|, |u_{cx}^+|, |u_{cy}^+|)$ to ensure that the lower camera speed is taken into account, and d_{max} is the longest path taken by the camera to reach robots and landmark. It can be seen here that the achievement of the proposed control scheme does not depend on the number of robots N_r, but rather on d_{max} whose value depends on the considered initial configurations (an example is presented in Section IV-A.3).

We can then use N computed in (27) to determine if the desired behavior can be realized without any loss of information. Indeed, the drift distance (24) could exceed the distance defined by $|\delta^-|$ before the camera has time to reach the position. We can then validate the system settings if:

$$N \geqslant \min(Nmin_j). \tag{28}$$

IV. SIMULATION RESULTS

Simulations results are presented in this section. All coordinates described in the following are expressed in meters.

A. Parameters

1) Initial configurations: The camera is initially located at $\boldsymbol{c}(t_0) = \big(0.7\,,\,0.6\big)$, the landmark at $\boldsymbol{m} = \big(0.5\,,\,0.5\big)$ and two robots at $\boldsymbol{r}_1(t_0) = \big(0.5\,,\,0.7\big)$ and $\boldsymbol{r}_2(t_0) = \big(1\,,\,0.7\big)$. We set also the model of the positions as being perfeclty estimated at the beginning of the simulation: $\boldsymbol{c}_m(t_0) = \boldsymbol{c}(t_0)$, $\boldsymbol{r}_{1m}(t_0) = \boldsymbol{r}_1(t_0)$ and $\boldsymbol{r}_{2m}(t_0) = \boldsymbol{r}_2(t_0)$. Uncertainties are at their minimum value at the initialization $p_j(t_0) = \sigma_s^2$.
The range of visibility is delimited by $\boldsymbol{s}_{min} = \big(-0.08\,,\,-0.08\big)$ and $\boldsymbol{s}_{max} = \big(0.08\,,\,0.08\big)$.
Concerning the noise added to the robots and the camera motion, the averages are null $\mu_c = \mu_1 = \mu_2 = 0$, and the standard deviations are $\sigma_c = 0.07$, $\sigma_1 = 0.09$ and $\sigma_2 = 0.08$. The noise added to the measures is defined by $\mu_s = 0$ and $\sigma_s = 0.005$. The maximum value for the tolerance is set to $\delta^+ = 2$, while the minimum value is $\delta^- = -0.08$ wich corresponds to the size of the field of view from the center of the image plane.
As for the camera control, the minimum and maximum values that can be reached are $\boldsymbol{u}_c^- = -2$m/s and $\boldsymbol{u}_c^+ = +2$m/s. And for the robots control, the gain is set at $k = 1$. The total simulation time is divided in 375 steps with sampling time of $\tau = 0.04$s.

2) Desired trajectories: The robot desired trajectories are

$$\begin{cases} \overrightarrow{\boldsymbol{r}_1^*} = \mathbf{A}\boldsymbol{r}_1^*(t_k) + \mathbf{B}\mathbf{E}\begin{pmatrix} 0.2\cos(t_k) \\ 0.2\sin(t_k) \end{pmatrix} \\ \overrightarrow{\boldsymbol{r}_2^*} = \mathbf{A}\boldsymbol{r}_2^*(t_k) + \mathbf{B}\mathbf{E}\begin{pmatrix} -0.2\cos(t_k) \\ -0.2\sin(t_k) \end{pmatrix} \end{cases}. \tag{29}$$

with $\boldsymbol{r}_1^*(t_0) = \boldsymbol{r}_1(t_0)$, $\boldsymbol{r}_2^*(t_0) = \boldsymbol{r}_2(t_0)$ and

$$\mathbf{E} = \begin{pmatrix} \cos(\tau) & -\sin(\tau) \\ \sin(\tau) & \cos(\tau) \\ \vdots & \vdots \\ \cos(N\tau) & -\sin(N\tau) \\ \sin(N\tau) & \cos(N\tau) \end{pmatrix} \in \mathbb{R}^{2N\times 2} \tag{30}$$

to obtain two circular paths repeated indefinitely. The radii of the circles are $R_1 = R_2 = 0.2$ and their centers are $\boldsymbol{C}_1 = \big(0.5\,,\,0.9\big)$ and $\boldsymbol{C}_2 = \big(1\,,\,0.5\big)$.

3) Prediction Horizon: To compute the prediction horizon, we first determine the longest path taken by the camera to correct the models. With the considered configurations, the longest path can be defined as:

$$d_{max} = d_m + d_r \tag{31}$$

with d_r the farthest distance between the two robots on their trajectory:

$$d_r = R_1 + R_2 + \sqrt{(C_{1_x} - C_{2_x})^2 + (C_{1_y} - C_{2_y})^2} \tag{32}$$

and d_m the distance between the landmark and the farthest robot from the landmark on its trajectory:

$$d_m = R_2 + \sqrt{(C_{2_x} - m_x)^2 + (C_{2_y} - m_y)^2} \tag{33}$$

Combining (27) and (31) we can then set $N = 22$. The condition (28) is then held, which means that the correction of the model is guaranteed.

B. Small prediction horizon

We first consider our control strategy with a very small prediction horizon $N = 1$ (see the first part of the video submitted as supplementary material). The numerical method chosen for minimizing J under the constraints $\mathbf{K}$ is available in the function *COBYLA* (Constrained Optimization BY Linear Approximation) [25] from the C++ library NLOPT [26]. This function is not the most efficient method to solve our optimization problem, but nevertheless allows to obtain simulation results which approach strongly the desired camera behavior. In this first simulation, the computation time required is approximately 1 ms for each function *COBYLA* call.

Figures 4a and 4b show the evolution of real s_x, s_y and desired s_x^* and s_y^* robot coordinates in the camera frame, while Figure 4c represents the evolution of the landmark coordinates. The camera field of view is depicted in orange while the evolution of the tolerances is in magenta (for the first robot), cyan (for the second robot) and black (for the landmark). We can thus visualize when the models have been corrected, that is when the blue and red curves are

between the two orange lines at the same time. Finally, Figure 4d shows the evolution of uncertainties together with the uncertainty thresholds.

It can be observed at $t = 0$ that the uncertainties do not exceed their thresholds. However, the one related to the camera is smaller than the other ones on the prediction horizon. Thanks to the conditions (20), δ^- is assigned to the landmark tolerance while δ^+ is assigned to the robots tolerance. Moreover, we can see on Figure 4c that $s_{cx}(t_0)$ and $s_{cy}(t_0)$ are not located in the interval described by $s_{min} - \delta_0$ and $s_{max} + \delta_0$. The camera must therefore move to the landmark to satisfy the constraint (21). The model of the camera is then corrected by the observation of the landmark.

In the same way, the camera must move to the first robot at $t = t_1$, when the prediction of the uncertainty $p_1(t_{k+N})$ reaches p_{max1}. We can see that p_1 exceeds p_{max1} while the camera moves to the first robot. However, the camera is still able to recover the robot in its field of view. Indeed, the drift distance of the first robot and the camera is not large enough for exceed the distance described by $|\delta^-|$.

This limit is however exceeded for the second robot at $t = t_2$, the minimum tolerance value δ^- is fixed to the second robot tolerance, while δ^+ is fixed to the landmark and the first robot until the end of the simulation. The robots are not assisted by the camera observations anymore and diverge from their desired trajectories. Thus, the blue and red curves diverge from green curves on Figures 4a and 4b.

The solution consists in properly computing the prediction horizon following the method described in Section III-C.

C. Large prediction horizon

We now apply our method with a prediction horizon set to $N = 22$ following the computation presented in Section IV-A.3 (see the last part of the video submitted as supplementary material). The computation time required is obviously greater than in the previous case. To be consistent with the sampling time τ, and for real-time issue, we have limited the resolution time of (13) to 40 ms. This is possible with the NLOPT library but may introduce a loss of accuracy in the solution.

As we can see on Figures 5a, 5b, 5c, the large prediction horizon allows the camera to switch from a target to another much earlier than in the previous simulation, resulting in an increase of the tolerances modification frequency. It causes a greater number of corrections achieved by the camera to satisfy the constraint (21).

It can also be observed Figure 5d that the uncertainties never exceed their threshold. The drift distance is never large enough for exceed the distance described by $|\delta^-|$. The camera can then always retrieve the robots in its field of view, resulting in the success of the trajectory tracking of the robots throughout the simulation thanks to the proper horizon prediction.

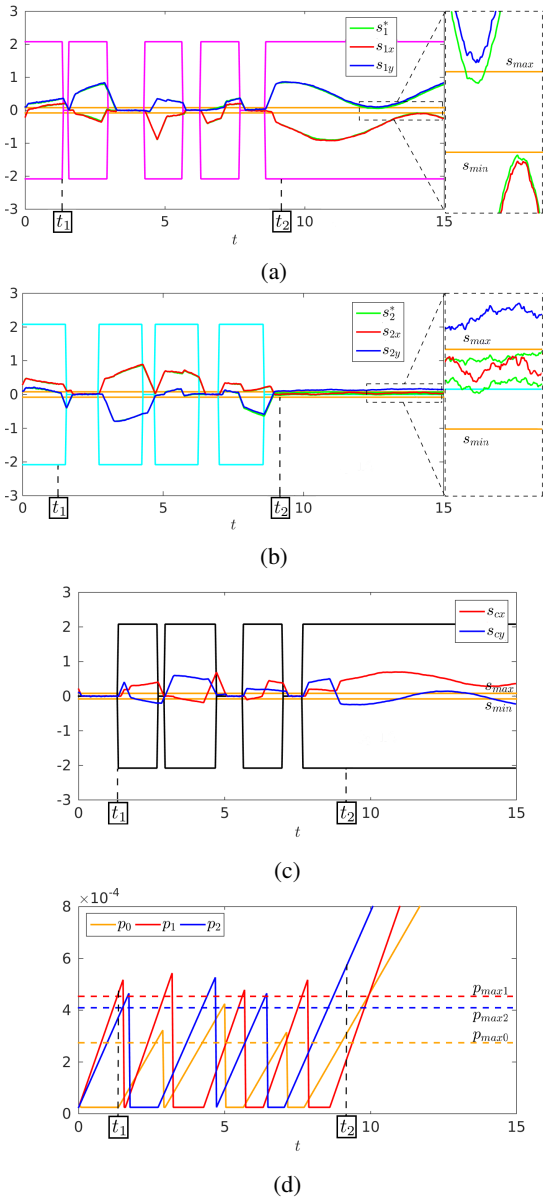


Fig. 4: Small prediction horizon: (a) and (b) show the desired and real coordinates of robots in the camera image plane, with $\boldsymbol{s}_{min} - \delta_i$ and $\boldsymbol{s}_{max} + \delta_i$ represented in magenta and cyan. (c) depicts the real coordinates of the landmark in the camera image plane, with $\boldsymbol{s}_{min} - \delta_0$ and $\boldsymbol{s}_{max} + \delta_0$ represented in black. (d) represents the uncertainties and their thresholds. At $t = t_2$, the camera is not able to observe the second robot in its field of view because of the small prediction horizon which does not anticipate the model errors.

V. CONCLUSION

We introduced in this paper a new camera control scheme to achieve a succession of visual servoing tasks by following the principle of active perception. This method is based on the model predictive control to anticipate the model errors over a future time horizon and, thanks to constraints, force the camera to move for correcting these errors. This method has been used to assist multiple robots subjected to noises to perform a trajectory tracking task. With the reported simulation results, we showed the effectiveness of our method and the importance of the prediction horizon in the parameters settings.

We applied this method in a 2D configuration. Future works will consider this method applied in more realistic

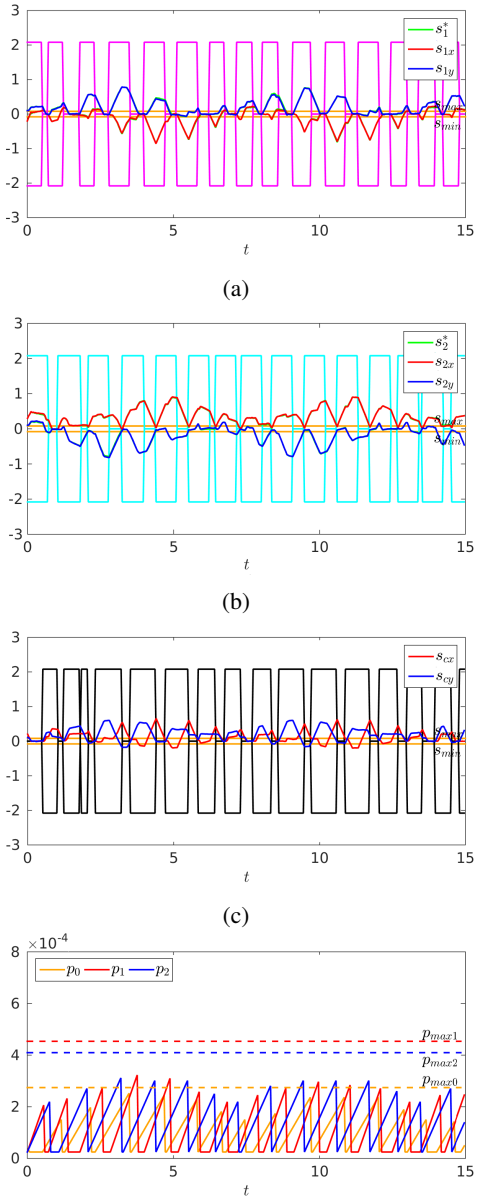


Fig. 5: Large prediction horizon: Compared to Fig. 4, the tolerance modification frequency increases thanks to the large horizon prediction, resulting by a greater number of model correction to achieve by the camera, which allows to assist indefinitely the robots trajectory tracking.

applications. This will involve considering more complex models for the UAV and robotics dynamics, as well as a more realistic propagation of the state estimation uncertainty. Morover, we will also consider different possible numerical solvers for solving the proposed minimization problem in order to meet real-time constraints.

ACKNOWLEDGMENTS

This work was partly funded by the Oseo Romeo 2 Project.

REFERENCES

[1] F. Chaumette and S. Hutchinson, "Visual servo control, part i: Basic approaches," *IEEE Robotics and Automation Magazine*, vol. 13, 2006.
[2] G. Morel, T. Liebezeit, J. Szewczyk, S. Boudet, and J. Pot, "Dynamic sensor planning in visual servoing," in 6^{th} *Int. Symp. on Experimental Robotics*, 2000, pp. 99–108.
[3] E. Marchand and G. Hager, "Dynamic sensor planning in visual servoing," in *ICRA 98*, pp. 1988–1993.
[4] Y. Mezouar and F. Chaumette, "Path planning for robust image-based control," *IEEE Trans. on Robotics and Automation*, vol. 18, 2002.
[5] A. Hafez, A. Nelakanti, and C. Jawahar, "Path planning approach to visual servoing with feature visibility constraints: a convex optimization based solution," in *IROS 2007*.
[6] T. Shen and G. Chesi, "Visual servoing path planning for cameras obeying the unified model," *Advanced Robotics*, vol. 26, 2012.
[7] M. Kazemi, K. Gupta, and M. Mehrandezh, "Randomized kinodynamic planning for robust visual servoing," *IEEE Trans. on Robotics*, vol. 29, no. 5, pp. 1197–1211, 2013.
[8] D. Folio and V. Cadenat, "Dealing with visual features loss during a vision-based task for a mobile robot," *Int. Journal of Optomechatronics*, vol. 2, no. 3, pp. 185–204, 2008.
[9] N. Cazy, C. Dune, P.-B. Wieber, P. Robuffo Giordano, and F. Chaumette, "Pose Error Correction For Visual Features Prediction," in *IROS 2014* - , Chicago, United States, Sept. 2014.
[10] T. Murao, T. Yamada, and M. Fujita, "Predictive visual feedback control with eye-in-hand system via stabilizing receding horizon approach," in *45th IEEE Conf. on Decision and Control*, dec. 2006, pp. 1758 –1763.
[11] M. Sauvée, P. Poignet, E. Dombre, and E. Courtial, "Image based visual servoing through nonlinear model predictive control," in *Int. Conf. on Decision and Control*, San Diego, CA, USA, December 13-15 2006.
[12] C. Lazar, A. Burlacu, and C. Copot, "Predictive control architecture for visual servoing of robot manipulators," in *IFAC World Congress*, Milano (Italy), August 28 - September 2 2011, pp. 9464–9469.
[13] G. Allibert, E. Courtial, and Y. Tour, "Real-time visual predictive controller for image-based trajectory tracking of mobile robot," in *17th IFAC World Congr.*, Seoul, Korea, July 2008.
[14] G. Allibert, E. Courtial, and F. Chaumette, "Predictive control for constrained image-based visual servoing," *IEEE Transaction on Robotics*, vol. 26, no. 5, pp. 933–939, October 2010.
[15] A. Assa and F. Janabi-Sharifi, "Robust model predictive control for visual servoing," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2014.
[16] C. Fiter, H. Omran, L. Hetel, and J.-P. Richard, "Tutorial on arbitrary and state-dependent sampling," in *13th European Control Conference*, 2014, pp. 1440–1445.
[17] R. Bajcsy, "Active perception," *Proceedings of the IEEE*, vol. 76, no. 8, pp. 996–1005, 1988.
[18] ——, "Active perception and exploratory robotics," *Artificial intelligence and information-control systems of robots*, 1989.
[19] L. Riazuelo, M. Tenorth, D. Marco, M. Salas, L. Mosenlechner, L. Kunze, M. Beetz, J. Tardos, L. Montano, and J. Montiel, "Roboearth web-enabled and knowledge-based active perception," in *IROS Workshop on AI-based Robotics*, 2013.
[20] K. Shubina and J. K. Tsotsos, "Visual search for an object in a 3d environment using a mobile robot," *Comput. Vis. Image Underst.*, vol. 114, no. 5, pp. 535–547, May 2010.
[21] B. Browatzki, V. Tikhanoff, G. Metta, H. Bülthoff, and C. Wallraven, "Active object recognition on a humanoid robot," in *ICRA*, 2012, pp. 2021–2028.
[22] Q. V. Le, A. Saxena, and A. Y. Ng, "Active perception: Interactive manipulation for improving object detection," *Standford University Journal*, 2008.
[23] G. Alenyà Ribas, F. Moreno-Noguer, A. Ramisa Ayats, and C. Torras, "Active perception of deformable objects using 3d cameras," in *Robot 2011: robótica experimental*, 2011, pp. 434–440.
[24] S. Thrun, W. Burgard, and D. Fox, *Probabilistic robotics*. MIT press, 2005.
[25] M. Powell, "Direct search algorithms for optimization calculations," *Acta numerica*, vol. 7, pp. 287–336, 1998.
[26] S. G. Johnson, "The nlopt nonlinear-optimization package, http://ab-initio.mit.edu/nlopt."