Visual predictive control of robotic arms with overlapping workspace

Emile Le Flecher, Adrien Durand-Petiteville, Viviane Cadenat, Thierry Sentenac

Visual predictive control of robotic arms with overlapping workspace

Le Flécher E.[1,2][a], Durand-Petiteville A[3], Cadenat V.[1,2] Sentenac T.[1]
[1]*CNRS, LAAS, 7 avenue du colonel Roche, F-31400 Toulouse, France*
[2]*Univ de Toulouse, UPS, F-31400, Toulouse, France*
[3]*Mechanical Department of Federal University of Pernambuco UFPE, Recife, Brazil*
{*eleflech,cadenat,sentenac*}*@laas.fr ajsd@cin.ufpe.br*

Keywords: Agricultural robotics, multi-arms, Non-linear Model Predictive Control, Image Based Visual Servoing

Abstract: This paper deals with multi-arms fruits picking in orchards. More specifically, the goal is to control the arms to approach the fruits position. To achieve this task a VPC strategy has been designed to take into account the dynamic of the environment as well as the various constraints inherent to the mechanical system, visual servoing manipulation and shared workspace. Our solution has been evaluated in simulation using on PR2 arms model. Different models of visual features prediction have been tested and the entire VPC strategy has been run on various cases. The obtained results show the interest and the efficiency of this strategy to perform a fruit picking task.

1 Introduction

To meet the increasing food demands from a growing world population, agriculture will need to double its production by 2050. Alternatively, it will need to reduce its environmental impact to fight against climate change and avoid to harm soils. Among the solutions presented in (Foley et al., 2011; Grift et al., 2008), robotics has been designated as one of the most promising strategies.

Agriculture robotics may operate in various environments such as open fields, orchards, green houses, ... In this work, one focuses on robots performing weeding, spraying, trimming, or harvesting in orchards. The autonomous achievement of any of these tasks requires the same capacities for the robotic system: (i) autonomously navigate in an orchard; (ii) detect and localize a set of natural landmarks of interest; (iii) approach the landmarks with some robotic manipulators; (iv) manipulate/interact with the landmarks of interest without damaging them. After addressing the autonomous navigation problem in (Durand-Petiteville et al., 2017; Le Flecher et al., 2017), the focus is now on the landmark approach problem when considering a couple of robotic manipulators embedded on a mobile platform. This topic has already been addressed during the design of systems harvesting cucumbers (Van Henten et al., 2003) or tomatoes (Zhao et al., 2016b). Despite the promising results, the harvesting efficiency needs to be increased to meet the viability demand from farmers (Zhao et al., 2016a). To do so, one of the most adopted approach consists of embedding several robotics arms on the mobile platform, as it is done in (Zhao et al., 2016a) for a melon harvester. However, it is pointed in (Vougioukas et al., 2016) that fruits are not uniformly distributed on commercial orchards trees. Indeed, to maximize the fruits accessibility, growers use specific trees with high fruits density, thus facilitating the manual harvesting. For this reason, during the design of a harvesting system, it seems then interesting to consider the use of several arms whose workspace overlap. Thus, even if the arms do not collaborate to manipulate one single object, they share a common workspace. It is then possible to use several arms to pick fruits in the same high density area, which decreases the overall harvesting time.

Regarding the control of the robotics arms, it seems interesting to focus on reactive controllers. Thus, when considering the harvesting problem, a tree is a highly dynamic environment because of the flexibility of the branches and the weight of the fruits: each time a fruit is harvested, the local distribution of fruits and branches is modified. It is then proposed to use an image based visual servoing (IBVS) scheme for its reactive capabilities (Chaumette and Hutchinson, 2006). This kind of controller will track the fruit

[a] https://orcid.org/0000-0002-2683-8675

of interest and use its coordinates in the image to control the arm. However, IBVS schemes working in the image space, they do not provide any guarantee regarding the trajectory of the camera in the workspace. This represents a major issue when considering arms sharing a common workspace. A well adopted solution to take into account constraints on position, velocity, or workspace is the model predictive control scheme. Indeed, when minimizing the cost function, the solver can deal with numerous constraints (Diehl and Mombaur, 2006). Thus, to obtain a reactive controller dealing with constraints, it is proposed to design an IBVS-based NMPC (nonlinear model predictive control) controller, also named Visual Predictive Control (VPC). For this particular case, the cost function is defined as the error between the image coordinates of the visual features representing the landmark of interest at the current pose and the desired one.

Similar control schemes have been already proposed. For example, in (Allibert et al., 2010) several prediction models are used to simulate the control of a 6 DOF camera subject to visibility constraints. In (Sauvee et al., 2006), the designed controller deals with torques, joints and image constraints and is evaluated in simulations. In (Hajiloo et al., 2015) an IBVS-based MPC controller using a tensor product model is developed. It also takes into account actuators limitations and visibility constraints.

In this paper, it is proposed to extend the VPC case to the control of two arms sharing a common workspace. Thus, in addition to drive the two end-effectors to their desired pose using the image space, the controllers have to guarantee the non-collision between both arms.

A non-linear predictive control strategy to perform a reactive control based on IBVS method is simulated on a robot equipped of multiple arms with eye-in-hand cameras. Each arm needs to achieve separate tasks sharing the same workspace. These tasks are subject to constraints such as the restricted field of view, the kinematic (joints limits in position and velocity) but also the self collision constraints.

The paper is organized as follows. In section II the robot model to estimate the location of visual features is defined. In section III, the control strategy is explained, as well as the cost function described and the diverse constraints designed. Section IV is dedicated to the simulation results obtained while section V summarizes and develops the expectations for the future.

2 Modeling

In this section, one first focuses on the modeling of the system, *i.e.,* the robotic arms, the cameras and the targets. Then, two models to predict the visual feature coordinates of a point are presented. One ends with a discussion regarding the viability of these models for the visual predictive control problem.

2.1 System modeling

The robotic system considered in this work is composed of n_a identical arms, each of them containing m_q revolute joints. A camera is attached to the end effector of each arm. To model, the system, let us first define a global frame $F_b = (O_b, \mathbf{x}_b, \mathbf{y}_b, \mathbf{z}_b)$ attached to the system base. Moreover, n_a frames $F_{c_i} = (O_{c_i}, \mathbf{x}_{c_i}, \mathbf{y}_{c_i}, \mathbf{z}_{c_i})$, with $i \in [1, ..., n_a]$, are used to represent the pose of the camera attached to the i^{th} arm.

One denotes q_{ij} and $\dot{q}_{ij}$, with $j \in [1, ..., m_q]$, respectively the angular position and velocity of the j^{th} joint of the i^{th} arm. Thus, for the i^{th} arm, one obtains the configuration vector $Q_i = [q_{i1}, q_{i2}, ..., q_{im_q}]^T$ and the control input vector $\dot{Q}_i = [\dot{q}_{i1}, \dot{q}_{i2}, ..., \dot{q}_{im_q}]^T$. Finally, we define for the whole system $Q = [Q_1^T, ..., Q_{n_a}^T]^T$ and $\dot{Q} = [\dot{Q}_1^T, ..., \dot{Q}_{n_a}^T]^T$.

The cameras mounted on the end effectors are modeled using the pinhole model. Thus, the projection matrices H_{I_i/c_i} mapping the 3D coordinates (x, y, z) of a point in the camera frame F_{c_i} to its 2D projection (X, Y) on the image plan F_{I_i} is defined as follows:

$$\begin{bmatrix} X \\ Y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} f/z & 0 & 0 & 0 \\ 0 & f/z & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{1}$$

In this work, the arms are controlled to make each camera reach a pose defined by the coordinates of point visual features in the image space. To do so, one uses n_a landmarks made of four points. When considering the i^{th} camera/target couple, the coordinates of the projection of each point on the image is denoted $S_{il} = [X_{il}, Y_{il}]^T$, with $i \in [1, ..., n_a]$ and $l \in [1, 2, 3, 4]$. Thus, the visual feature vector for each camera is defined as $S_i = [S_{i1}, S_{i2}, S_{i3}, S_{i4}]^T$, whereas the system one is given by $S = [S_1^T, ..., S_{n_a}^T]^T$. In the same way, a vector for the visual feature desired coordinates of each camera is defined as $S_i^* = [S_{i1}^*, S_{i2}^*, S_{i3}^*, S_{i4}^*]^T$.

2.2 Prediction models

A visual predictive control scheme requires to be able to predict the values of the visual features for a given sequence of control inputs. In (Allibert et al., 2010), the authors present two ways to perform the prediction step for a flying camera: the local and global approaches. This section is devoted to the presentation of both methods for the case of a 6 degrees of freedom camera embedded on a robotic arm.

2.2.1 The global model

The global approach is based on the robot geometric model and the camera projection matrix. The idea consists of computing the 3D coordinates of the features in the next camera frame, then to project them onto the camera image plan. To do so, let us consider a discrete time system with a sampling period T_s, where $t_{k+1} = t_k + T_s$, and the notation $.(k) = .(t_k)$. The configuration $Q_i(k+1)$ of the i^{th} arm, with $i \in [1, ..., n_a]$, at the next iteration obtained after applying the control inputs $\dot{Q}_i(k)$ is given by:

$$Q_i(k+1) = Q_i(k) + \dot{Q}_i(k)T_s \tag{2}$$

Knowing the arm configuration at instants t_k and t_{k+1}, it is then possible to compute the homogeneous transformation matrices between the camera frame F_{c_i} and the base one F_b.

$$\bar{o}_{F_b}(.) = H_{b/c_i}(.)\bar{o}_{F_{c_i}}(.) \tag{3}$$

where $\bar{o}$ are the homogeneous coordinates of a point feature expressed in the base or camera frame. Finally, coupling (3) with (1), one obtains a prediction of the coordinates of a point feature in the image space.

$$\begin{aligned} &S_{il}(k+1) = \\ &H_{I_i/c_i}(k+1)H^{-1}_{b/c_i}(k+1)H_{b/c_i}(k)H^{-1}_{I_i/c_i}(k)S_{il}(k) \end{aligned} \tag{4}$$

2.2.2 The local model

The local approach relies on the differential equation mapping the derivative of the visual feature coordinates $\dot{S}_{il}$ to the robot control input vector $\dot{Q}_i$. To obtain this equation, one uses the robot's Jacobian J_i, mapping the camera kinematic screw T_{c_i} and $\dot{Q}_i$ (5), and the interaction matrix L_{il}, mapping $\dot{S}_{il}$ to T_{c_i} (6).

$$T_{c_i} = J_i\dot{Q}_i \tag{5}$$

$$\dot{S}_{il} = L_{il}T_{c_i} \tag{6}$$

where

$$L_{il} = \begin{bmatrix} -\frac{1}{z_l} & 0 & \frac{X_l}{z_l} & X_lY_l & -(1+X_l^2) & Y_l \\ 0 & -\frac{1}{z_l} & \frac{Y_l}{z_l} & 1+Y_l^2 & -X_lY_l & X_l \end{bmatrix} \tag{7}$$

By combining equations (5) and (6), one obtains the following local model for each arm/camera couple:

$$\dot{S}_{il} = L_{il}\, J_i\, \dot{Q}_i \tag{8}$$

2.2.3 Usability of the models

When considering using one of the models on an experimental system, several aspects have to be taken into account. First, both models require a measure/estimate of the visual feature depth z (see equations (1) and (7)). However, when using a monocular camera, this value can only be retrieved by estimation. Another widely adopted option, consists of using a constant value, such as the depth value at the initial or desired pose (Chaumette and Hutchinson, 2006). The second issue is related to the integration of the local model to obtain the prediction equation. A first option consists of using an advanced numeric scheme to accurately compute the integration, whereas a second one relies on a linear approximation of the model, such as:

$$S_{il}(k+1) = S_{il}(k) + L_{il}J_i\dot{Q}_i(k)T_s \tag{9}$$

Thus, depending on the user choices regarding these issues, the prediction feature coordinates will be more or less accurate. In section 4, different configurations will be evaluated for our specific application in order to select the most appropriate and realistic solution.

3 Controller design

In this section the control of the arms motion to reach their own targets and compounding necessary constraints are presented. To achieve these goals, NMPC control has been implemented to find the optimal inputs vector which minimizes a cost function subjects to constraints. To do so, we first present the cost function to be minimized in order to reach the desired targets. Then, the expression of each constraint will be defined.

3.1 Visual Predictive Control

The proposed VPC scheme couples NMPC (Nonlinear Model Predictive Control) with IBVS (Image-Based Visual Servoing). Thus, similarly to NMPC, it consists of computing an optimal control sequence $\bar{\dot{Q}}(.)$ 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 $C(\bar{\dot{Q}}(.))$. Moreover, similarly to IBVS, the task to achieve is defined as an error in the image space. Thus, the cost function to

minimize is the sum of the quadratic error between the visual feature coordinates $\hat{S}(.)$ predicted over the horizon N_p and the desired ones S^*. It then possible to write the VPC problem as:

$$\bar{\dot{Q}}(.) = \min_{\dot{Q}(.)} \left(J_{N_P}(S(k), \dot{Q}(.)) \right) \tag{10}$$

with

$$J_{N_P}(S(k), \dot{Q}(.)) = \sum_{p=k+1}^{k+N_p-1} [\hat{S}(p) - S^*]^T [\hat{S}(p) - S^*] \tag{11}$$

subject to

$$\hat{S}(.) = (4) \text{ or } (8) \tag{12a}$$

$$\hat{S}(k) = S(k) \tag{12b}$$

$$C(\bar{\dot{Q}}(.)) \leq 0 \tag{12c}$$

As mentioned in the previous section, the visual feature coordinates can be predicted either using the global or local models (equation (12a)). To compute the prediction, one uses the last measured visual features (equation (12b)). Finally, equation (12c) defines the set of constraints to be taken into account when minimizing the cost function (11).

Thus, solving equation (10) leads to the optimal sequence of control inputs $\bar{\dot{Q}}(.)$. In this work, and as it is usually done, only the first element $\bar{\dot{Q}}(1)$ of the sequence 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, *e.i.,* the cost function (11) is vanished.

3.2 Constraints

As seen in the previous section, it is possible to add a set of constraints in the minimization problem. In this section, one presents the constraints related to the geometry of the robot, the velocity of the joints, and the field of view of the camera. One finishes with defining a set of constraints that guarantee the non-collision of the arms despite sharing a common workspace.

3.2.1 System constraints

The joint velocity lower and upper bounds are respectively denoted $\dot{Q}_{min}$ and $\dot{Q}_{max}$. They are two $N_{\dot{Q}}$ long vector, with $N_{\dot{Q}} = n_a m_q N_p$. They allow to define the lower and upper limit of each joint of each arm over the prediction horizon. Consequently, the constraint on $\dot{Q}(.)$ is described in the domain $\kappa = [\dot{Q}_{min}, \dot{Q}_{max}]$, and the velocity constraint is then written as:

$$\begin{bmatrix} \dot{Q} - \dot{Q}_{max} \\ \dot{Q}_{min} - \dot{Q} \end{bmatrix} \leq 0 \tag{13}$$

Mechanical limits follow the same principle. The lower and upper bounds of the angular values are denoted Q_{min} and Q_{max}. They are two N_Q long vector, with $N_Q = n_a m_q N_p$. These constraints are then given by:

$$\begin{bmatrix} Q - Q_{max} \\ Q_{min} - Q \end{bmatrix} \leq 0 \tag{14}$$

Finally, the visual constraints are defined by the image dimension. Ones denotes the bounds by S_{min} and S_{max}. They are two N_S long vector, with $N_S = 8 n_a N_p$. Thus, all the predicted visual features coordinates need to be kept into this space, and the constraints can then be written as:

$$\begin{bmatrix} S - S_{max} \\ S_{min} - S \end{bmatrix} \leq 0 \tag{15}$$

3.2.2 Overlapping workspace constraint

Due to the overlapping workspace, it is mandatory to guarantee the non-collision between the arms. To do so, one computes the shortest distance $d_{i,j|i',j'}$ between the j^{th} joint of the i^{th} arm and the j'^{th} joint of the i'^{th} arm. The closest distance is evaluated as the norm of the segment perpendicular to both assessed links as it is illustrated in figure 1. Finally, in order to avoid collision, one defines the following constraint:

$$\left[d_{min} - d \right] \leq 0 \tag{16}$$

where $d = [d_{1,1|2,1}, ..., d_{n_a-1,m_q|n_a,m_q}]$ and d_{min} is a predefined distance.

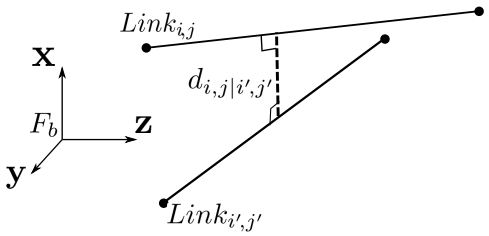


Figure 1: Computation of the shortest distance $d_{i,j|i',j'}$ between links L_{ij} and $L_{i'j'}$.

3.2.3 Terminal constraint

To make the system converge, it is mandatory to guarantee that the predictive control problem is feasible (Lars and Pannek, 2016). To check the feasability, one adds a final constraint in the optimization problem. It

is defined as the error between the prediction of the visual feature coordinates $\hat{S}(k+N_p-1)$ obtained at the end of the prediction horizon, and the desired ones S^*. If the solver cannot compute a sequence of control inputs $\bar{\dot{Q}}$ that respects this constraint, then the problem is not feasible, and there is no guarantee regarding the system convergence.

$$\left[\hat{S}(k+N_p-1)-S^*\right] \leq 0 \tag{17}$$

All the constraints are now defined and can be integrated into the non-linear function $C(\bar{\dot{Q}}(.))$:

$$C(\bar{\dot{Q}}) \leq 0 \Rightarrow \begin{bmatrix} \dot{Q}-\dot{Q}_{max} \\ \dot{Q}_{min}-\dot{Q} \\ Q-Q_{max} \\ Q_{min}-Q \\ S-S_{max} \\ S_{min}-S \\ \hat{D}_{min}-D \\ \hat{S}(k+N_p-1)-S^* \end{bmatrix} \leq 0 \tag{18}$$

Thus, the optimal input must belong to the domain defined by $C(\bar{\dot{Q}}(.))$.

4 Simulation

In this section, one presents the results obtained by simulating the control of two arms with MATLAB software. A 7 DOF model has been selected, based on the arms embedded on the PR2 robot. In these simulations the task consists of positioning the cameras attached to each end-effector with respect to two given targets. Each desired position is defined in the image space by $S_i = [x_1, y_1, x_2, y_2, x_3, y_3, x_4, y_4]$, which are the coordinates of four points representing the corresponding target. Moreover, the prediction horizon is set up at $N_p = 5$. Regarding the constraints, the angular velocity limits for the first prediction step are given by $\kappa = [-0.2\ 0.2]$ for each joint, whereas it is extended to $\kappa = [-5\ 5]$ for the rest of the prediction step. This allows to decrease the computation while guaranteeing a feasible control input for the first step. The size of the image is $S_{min} = [-4\ -4]$ and $S_{max} = [4\ 4]$. The angular constraints on each joint are setup as described in the PR2 documentation (Garage, 2012). Finally the minimum collision avoidance distance is fixed to $D_{min} = 0.1$. The initial coordinate of the visual features are computed based on the initial configuration of the robot given by $q_{init} = [0\ \frac{\pi}{2}\ 0\ 0\ 0\ 0\ 0\ \frac{\pi}{2}\ 0\ 0\ 0\ 0\ 0]^T$ for every simulation. Finally, one fixes $T_s = 50ms$.

4.1 Comparison between different prediction methods

The first set of simulation assesses the accuracy of different predictive models presented in section 2.2. To do so, one drives the arms from q_{init} to $q^* = [-\frac{\pi}{8}\ \frac{\pi}{2}\ 0\ 0\ 0\ 0\ 0\ \frac{\pi}{2}\ 0\ 0\ 0\ 0\ 0]^T$. During this motion, the coordinates of the visual features predicted over the prediction horizon, are compared with a ground truth. In this case, it consists of the prediction obtained with the global model computed with the real depth of each visual feature.

Figure 2 shows the evolution over the simulation of the quadratic error between the predicted coordinates of one visual feature and the ground truth. Six models are compared by combining the global model (solid lines), local Euler approximation (dashed lines) and local Runge Kutta approximation (dotted lines) with the depth values real z (black) and desired z^* (blue). It can be seen that the global models and the local Runge Kutta are significantly more accurate than the local Euler one. Indeed, using this latter prediction scheme introduces a large error, and might not allow the system to converge toward the desired configuration. In a second figure 3, the previous errors were summed over the simulation to offer a better view of model performances. When considering the use of the real depth, the global model is the most accurate. However, this result is obtained in a simulation, and its use in a real context would require an estimator of the depth. Thus, one now considers one more realistic case where the depth is approximated with the constant value z^*. In this case, the local Runge Kutta is the most accurate model. Thus, for the rest of the simulations the local Runge Kutta model with z^* is used to evaluate the performances in a more realistic scenario.

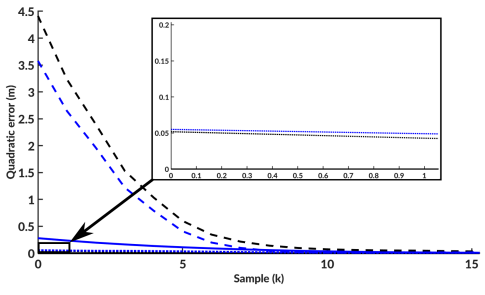


Figure 2: Quadratic errors of prediction over the simulation -Solid: Global model -Dashed: Local Euler -Dotted: Local Runge Kutta -Black: Real z -Blue: Desired z

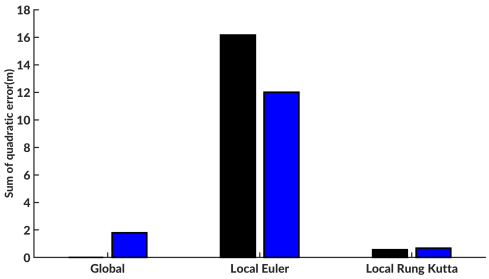
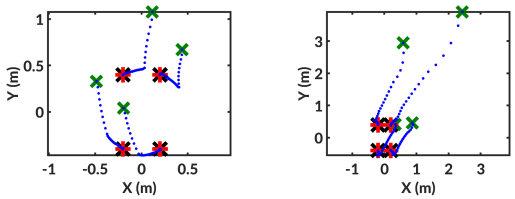


Figure 3: Sum of the quadratic errors of prediction for global, local Euler and local Runge-Kutta methods -Black: Real z -Blue: Desired z

4.2 First validation

For the second simulation, the system has to reach the desired configuration $q^* = [-\frac{\pi}{16}\ \frac{3\pi}{8}\ 0\ \frac{\pi}{4}\ 0\ \frac{\pi}{8}\ 0\ \frac{\pi}{16}\ \frac{\pi}{2}\ \frac{\pi}{4}\ \frac{\pi}{8}\ 0\ 0\ 0]^T$ from the initial one q_{init} while dealing with the previously presented constraints. This visual predictive control was performed using the local model with Runge Kutta approximation and the desired depth z^*. The trajectories in the images of visual features are displayed in figure 4 and show the achievement of the task for both cameras. Indeed, the proposed controller was able to drive both arms to make the cameras converge towards the desired features from the initial ones (respectively represented by the red and green crosses in figures 4(a) and 4(b)). To achieve the task in the image space, the arms had to reach the poses presented in figure 5. During the realization of this task, the image limits were never reached. Similarly there was no risk of collision during the servoing. Moreover, it can be seen in figures 6(a) and 6(b) that the constraints on the joint configurations were taken into account. Similarly, in figures 6(c) and 6(d), one can see that the constraints on the joint velocities were taken into account to achieve the task. Indeed, the applied commands stay within the given range $\kappa = [-0.2\ 0.2]$ all along the simulation.



(a) Left camera - Local model (b) Right camera - Local model

Figure 4: Visual features location using local Runge-Kutta model and real z - Blue dotted: Trajectories - Red crosses: Final locations - Green crosses: Initial locations

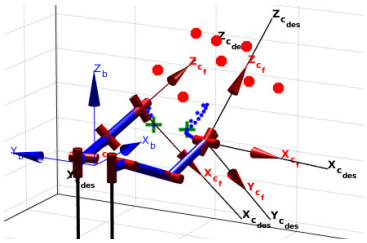
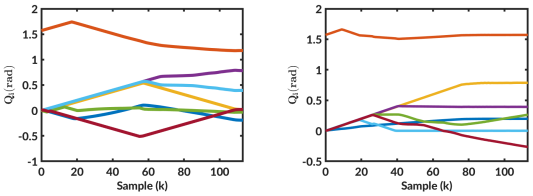
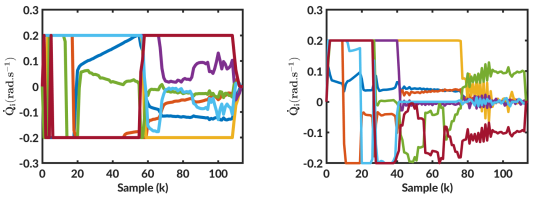


Figure 5: Final poses of the arms - Red dots represent the two targets - Green crosses are the initial poses of the end effectors



(a) Right arm joints angular values (b) Left arm joints angular values

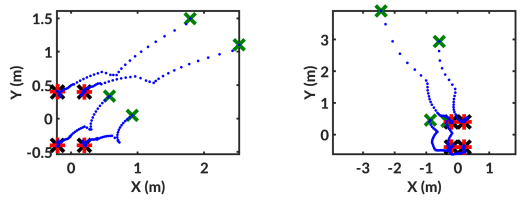


(c) Right arm joints angular velocities (d) Left arm joints angular velocities

Figure 6: Joint configurations and velocities

4.3 Collision avoidance

In this last simulation, the system has to reach the desired configuration $q^* = [\frac{\pi}{16}\ \frac{3\pi}{8}\ 0\ \frac{\pi}{4}\ 0\ \frac{\pi}{8}\ 0\ -\frac{\pi}{16}\ \frac{\pi}{2}\ \frac{\pi}{4}\ \frac{\pi}{8}\ 0\ 0\ 0]^T$. This set is run using the Runge Kutta local model with the desired depth values. One more time the desired visual features are reached from the initial poses (figure 7). However, as it can be seen in figure 8, the arms had to cross each other to reach the desired poses. Thanks to the shared environment constraint included in the control problem, the collision was avoided. Indeed, figure 9 shows that the distance between the different links was never smaller than the fixed value $D_{min} = 0.1$. In addition, the constraints on the joints angular values are respected. In figure 10, it can be seen that the right arm joints never reach their angular limits represented by red dashed lines (the joints represented in figures 10(e) and 10(g) do not have limits on PR2 robot). Similarly, the control inputs of the right arm present saturations due to the angular velocity constraints, represented by red dashed lines in figure 11, but never goes beyond.



(a) Visual features in image left (b) Visual features in image right

Figure 7: Visual features location using local Runge-Kutta model and real z - Blue dotted: Trajectories - Red crosses: Final locations - Green crosses: Initial locations

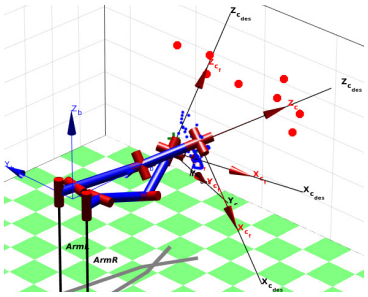


Figure 8: Final poses of the arms - Red dots represent the two targets - Green crosses are the initial poses of the end effectors

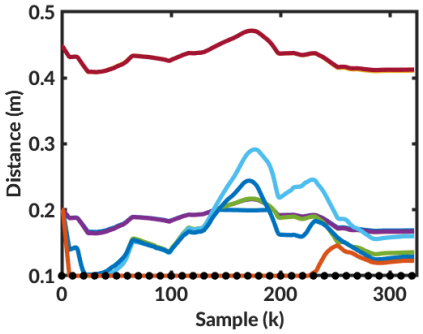
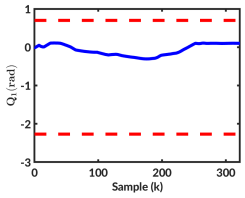
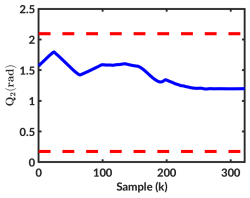


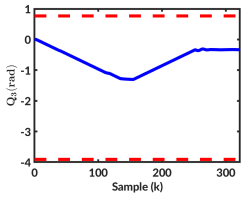
Figure 9: Collision distance d - Solid: Distances between links, Dotted: Minimal allowed distance



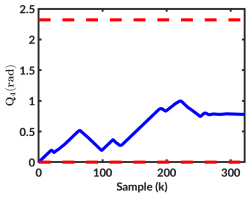
(a) q_{11} joint angular values



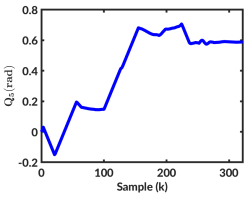
(b) q_{12} joint angular values



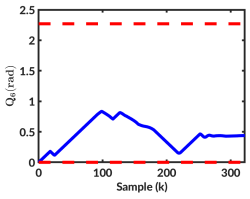
(c) q_{13} joint angular values



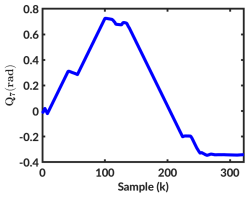
(d) q_{14} joint angular values



(e) q_{15} joint angular values



(f) q_{16} joint angular values

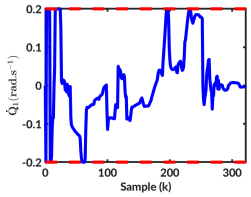


(g) q_{17} joint angular values

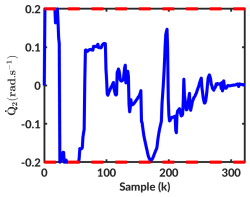
Figure 10: Configuration of the right arm over the simulation - Blue solid: Angular joint value q_{1j} - Red dashed: Maximum angles

5 Conclusion

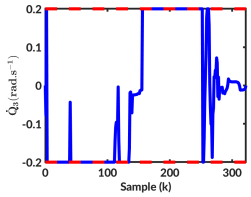
In this paper, we have addressed the problem of fruit picking in orchards performed by a multi-arms robot. The goal is to bring the different end-effectors close to their designated fruits while taking into account the dynamic environment as well as the constraints inherent to the system mechanics, the visual manipulation and the shared workspace. The proposed solution relies on a VPC strategy to fulfill these requirements. First, several prediction models were presented. Next, the visual servoing task and the constraints were described including them in a NMPC strategy. Finally, simulations based on PR2 manipulators were performed to assess the performance of each predictive model and to validate the completion of the servoing. The obtained results show the efficiency of this approach to perform various tasks in a shared workspace. However, multiple steps still need to be accomplished. First, this method must be integrated on a real robot. This step will require to optimize the computational time to run it in real time. In addition, a constraint avoiding the target occlusions in the image must be integrated. Finally, this manipulation controller will be coupled with our previous works about the autonomous navigation in orchards to perform a complete fruits picking task.



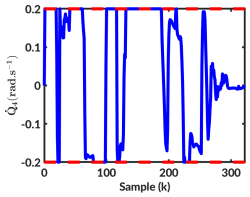
(a) $\dot{q}_{11}$ joints angular velocity



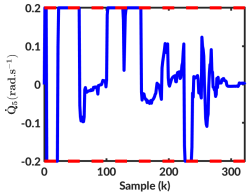
(b) $\dot{q}_{12}$ joints angular velocity



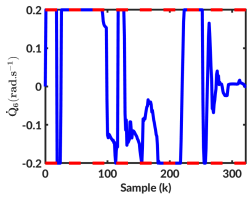
(c) $\dot{q}_{13}$ joints angular velocity



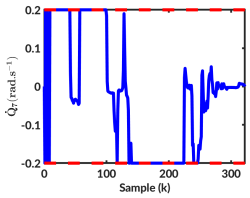
(d) $\dot{q}_{14}$ joints angular velocity



(e) $\dot{q}_{15}$ joints angular velocity



(f) $\dot{q}_{16}$ joints angular velocity



(g) $\dot{q}_{17}$ joints angular velocity

Figure 11: Right arm joints velocities - Blue solid: Joint velocity value - Red dashed: Maximum velocities

REFERENCES

Allibert, G., Courtial, E., and Chaumette, F. (2010). Predictive Control for Constrained Image-Based Visual Servoing. *IEEE Transactions on Robotics*, 26(5):933–939.

Chaumette, F. and Hutchinson, S. (2006). Visual servo control. i. basic approaches. *IEEE Robotics & Automation Magazine*, 13(4):82–90.

Diehl, M. and Mombaur, K., editors (2006). *Fast motions in biomechanics and robotics: optimization and feedback control*. Number 340 in Lecture notes in control and information sciences. Springer, Berlin ; New York. OCLC: ocm71814232.

Durand-Petiteville, A., Le Flecher, E., Cadenat, V., Sentenac, T., and Vougioukas, S. (2017). Design of a sensor-based controller performing u-turn to navigate in orchards. In *Proc. Int. Conf. Inform. Control, Automat. Robot.*, volume 2, pages 172–181.

Foley, J. A., Ramankutty, N., Brauman, K. A., Cassidy, E. S., Gerber, J. S., Johnston, M., Mueller, N. D., O'Connell, C., Ray, D. K., West, P. C., Balzer, C., Bennett, E. M., Carpenter, S. R., Hill, J., Monfreda, C., Polasky, S., Rockstrm, J., Sheehan, J., Siebert, S., Tilman, D., and Zaks, D. P. M. (2011). Solutions for a cultivated planet. *Nature*, 478:337–342.

Garage, W. (2012). Pr2 user manual.

Grift, T., Zhang, Q., Kondo, N., and Ting, K. C. (2008). A review of automation and robotics for the bio- industry. *Journal of Biomechatronics Engineering*, 1(1):19.

Hajiloo, A., Keshmiri, M., Xie, W.-F., and Wang, T.-T. (2015). Robust On-Line Model Predictive Control for a Constrained Image Based Visual Servoing. *IEEE Transactions on Industrial Electronics*, pages 1–1.

Lars, G. and Pannek, J. (2016). *Nonlinear model predictive control*. Springer Berlin Heidelberg, New York, NY.

Le Flecher, E., Durand-Petiteville, A., Cadenat, V., Sentenac, T., and Vougioukas, S. (2017). Implementation on a harvesting robot of a sensor-based controller performing a u-turn. In *2017 IEEE International Workshop of Electronics, Control, Measurement, Signals and their Application to Mechatronics (ECMSM)*, pages 1–6. IEEE.

Sauvee, M., Poignet, P., Dombre, E., and Courtial, E. (2006). Image Based Visual Servoing through Nonlinear Model Predictive Control. In *Proceedings of the 45th IEEE Conference on Decision and Control*, pages 1776–1781, San Diego, CA, USA. IEEE.

Van Henten, E., Hemming, J., Van Tuijl, B., Kornet, J., and Bontsema, J. (2003). Collision-free Motion Planning for a Cucumber Picking Robot. *Biosystems Engineering*, 86(2):135–144.

Vougioukas, S. G., Arikapudi, R., and Munic, J. (2016). A Study of Fruit Reachability in Orchard Trees by Linear-Only Motion. *IFAC-PapersOnLine*, 49(16):277–280.

Zhao, Y., Gong, L., Huang, Y., and Liu, C. (2016a). A review of key techniques of vision-based control for harvesting robot. *Computers and Electronics in Agriculture*, 127:311–323.

Zhao, Y., Gong, L., Liu, C., and Huang, Y. (2016b). Dual-arm Robot Design and Testing for Harvesting Tomato in Greenhouse. *IFAC-PapersOnLine*, 49(16):161–165.