Sampling-Based MPC for Constrained Vision Based Control

Ihab S. Mohamed, Guillaume Allibert, Philippe Martinet

Sampling-Based MPC for Constrained Vision Based Control

Ihab S. Mohamed[1] and Guillaume Allibert[2] and Philippe Martinet[3]

***Abstract*— Visual servoing control schemes, such as Image-Based (*IBVS*), Pose Based (*PBVS*) or Hybrid-Based (*HBVS*) have been extensively developed over the last decades making possible their uses in a large number of applications. It is well-known that the main problems to be handled concern the presence of local minima or singularities, the visibility constraint, the joint limits, etc. Recently, Model Predictive Path Integral (*MPPI*) control algorithm has been developed for autonomous robot navigation tasks. In this paper, we propose a MPPI-VS framework applied for the control of a 6-DoF robot with 2D point, 3D point, and Pose Based Visual Servoing techniques. We performed intensive simulations under various operating conditions to show the potential advantages of the proposed control framework compared to the classical schemes. The effectiveness, the robustness and the capability in coping easily with the system constraints of the control framework are shown.**

I. INTRODUCTION

Vision Based Control, also named Visual Servoing (VS), has been developed using directly 2D information extracted from images (IBVS), indirectly by computing 3D information from it (PBVS), or by combining both 2D and 3D information in the same scheme (HBVS). Nice state of the art have been published in [1]–[4]. As referred in [5], it is still necessary to instantiate Visual servoing to dedicated applications, to develop new non linear control strategies (i.e multi sensor based control), and to explore new Direct Visual Servoing approaches by the use of a data-driven methodology. In addition, the study of stability, singularity locus, local minima of Visual Servoing scheme are still open problems to be addressed in deep. Many advanced control frameworks have been applied to Visual servoing, like General Predictive Control [6], Model Predictive Control (*MPC*) [7], Linear Quadratic Gaussian [8], and Computed Torque Control [9], to cite some.

A new algorithmic methodology based on Path Integral (PI) optimal control theory has been proposed by Kappen in [10], for solving the nonlinear Stochastic Optimal Control problem. Traditionally, based on dynamic programming, this problem is defined by a partial differential equation (PDE) known as the *Hamilton-Jacobi-Bellman* (HJB) equation which can not be solved analytically. In *PI control* theory, non-linear PDE can be transformed into an expectation over all possible trajectories using the *Feynman-Kac* (FK) lemma. This transformation allows to solve the problem by sampling methods, such as *forward-in-time* Monte-Carlo approximation, instead of solving the HJB equation *backward-in-time*. Inspired by the *PI control* theory, Williams et al. [11] proposed a sampling-based model predictive control algorithm known as Model Predictive Path Integral (*MPPI*) control framework, which has been successfully applied to autonomous driving task in 2D environments. More recently, a generic and elegant *MPPI* control framework has been presented in [12], which enables the robot to navigate autonomously in either 2D or 3D environments that are inherently uncertain and partially observable.

Although MPPI and MPC follow the same control strategy, we believe that the *MPPI* control framework significantly outperforms the conventional *MPC* strategy because it is a sampling-based and derivative-free optimization method, and it does neither require the computation of gradients (i.e., derivatives), and neither the first- or second-order approximation of the system dynamics and quadratic approximation of the cost functions. Additionally, discontinuous cost functions (i.e., indicator functions) can be easily handled and added to the running cost function. For instance, in the context of autonomous flying tasks, a *large-weighted* indicator function can be employed as part of the running cost, for penalizing the collision with ground or obstacles.

In this paper, the *MPPI-VS* framework based on *PI control* theory is presented. More precisely, we propose a *real-time* and *inversion-free* control method for all image-based, 3D point-based (*3DVS*), and position-based visual servo schemes, which has been validated on a 6-DoF Cartesian robot with an *eye-in-hand* camera. The paper is organized as follows. In section II, classical visual servoing concepts are summarized for 2D image point, 3D point, and Pose features. Section III described the proposed MPPI-VS framework. Section IV is dedicated to the results and discussion. The last section presents the conclusion and perspectives of the work.

II. CLASSICAL VISUAL SERVOING CONTROL SCHEMES

Classical Visual Servoing techniques are summarized in [2]. They are based on the establishment of an interaction matrix $L_{\mathbf{s}}$ that characterizes the evolution of a sensor feature $\mathbf{s}(t)$ regarding to the relative velocity between the sensor and the environment part from which the feature is extracted. The main objective of all vision-based control schemes is to minimize the error $\mathbf{e}(t)$ between the current visual features $\mathbf{s}(t)$ and the desired features $\mathbf{s}^*$, which is typically defined as $\mathbf{e}(t) = \mathbf{s}(t) - \mathbf{s}^*$. By imposing an exponential decrease of the error ($\dot{\mathbf{e}}(t) = -\lambda_s \cdot \mathbf{e}(t)$), and considering an *eye-in-hand*

[1]Ihab S. Mohamed carried out this work at the Université Côte d'Azur, Inria, CNRS, I3S, France. He is now with the Luddy School of Informatics, Computing, and Engineering, Indiana University, Bloomington, IN 47408 USA, mohamedi@iu.edu

[2]Guillaume Allibert is at the Université Côte d'Azur, CNRS, Inria, I3S, France, allibert@i3s.unice.fr

[3]Philippe Martinet is at the Université Côte d'Azur, Inria, CNRS, I3S, France, {philippe.martinet}@inria.fr

configuration, we obtain a simple proportional control law:

$$\mathbf{v}_c = -\lambda_s \cdot \widehat{L_s}^{+} \cdot (\mathbf{s}(t) - \mathbf{s}^*) \tag{1}$$

In this expression, $\mathbf{v}_c = [\boldsymbol{v}_c; \boldsymbol{\omega}_c]$ is the kinematic screw applied to the camera, with the translational velocity $\boldsymbol{v}_c = [v_x; v_y; v_z]$ and the rotational velocity $\boldsymbol{\omega}_c = [\omega_x; \omega_y; \omega_z]$. Let's define a $3D$ point $\boldsymbol{P} = [X; Y; Z]$ whose image projection is $\boldsymbol{P_{Im}} = [u; v]$. Let's also define the rotation matrix between the desired camera frame and the current camera frame ${}^{c^*}\mathbf{R}_c$ and the pose (position, orientation) between this two frames as $\boldsymbol{Pos} = [\mathbf{t}; \theta\mathbf{u}]$. The following Table gives the expressions of the corresponding interaction matrices $(L_{\boldsymbol{P}}, L_{\boldsymbol{P_{Im}}}, L_{\boldsymbol{Pos}})$. Γ represents the intrinsic parameters (f_u, f_v, u_0, v_0) of a pinhole camera, and $I_3 \in \mathbb{R}^{3\times3}$ is the identity matrix.

$\mathbf{s}(t)$	$L_{\mathbf{s}} \quad (L_{\boldsymbol{P}}, L_{\boldsymbol{P_{Im}}}, L_{\boldsymbol{Pos}})$
$[X; Y; Z]$	$\begin{pmatrix} -I_3 & \widetilde{\boldsymbol{P}} \end{pmatrix}$, $\widetilde{\boldsymbol{P}}$ is the skew matrix of $\boldsymbol{P}$
$[u; v]$	$\begin{pmatrix} -\frac{f_u}{Z} & 0 & \frac{u}{Z} & \frac{uv}{f_v} & -f_u - \frac{u^2}{f_u} & \frac{f_u}{f_v} v \\ 0 & -\frac{f_v}{Z} & \frac{v}{Z} & f_v + \frac{v^2}{f_v} & -\frac{uv}{f_u} & -\frac{f_v}{f_u} u \end{pmatrix}$
$[\mathbf{t}; \theta\mathbf{u}]$	$\begin{pmatrix} {}^{c^*}\mathbf{R}_c & \mathbf{0}_3 \\ \mathbf{0}_3 & \mathbf{L}_{\theta\mathbf{u}} \end{pmatrix}$, $\mathbf{L}_{\theta\mathbf{u}} = I_3 - \frac{\theta}{2}[\mathbf{u}]_\times + \left(1 - \frac{\operatorname{sinc}\theta}{\operatorname{sinc}^2\frac{\theta}{2}}\right)[\mathbf{u}]_\times^2$

For the estimation of the interaction matrix $\widehat{L_s}$, we can consider different cases of ${L_s}^{\#}$: at each iteration (#0), at each iteration with fixed depth (#1), at equilibrium (#2), and finally the half sum of cases #0 and #2 [13].

Case #0	Case #1	Case #2	Case #3
$[X; Y; \widehat{Z}]$	/	$[X^*; Y^*; \widehat{Z^*}]$	$0.5(L_P^0 + L_P^2)$
$[u; v; \widehat{Z}, \widehat{\Gamma}]$	$[u; v; \widehat{Z^*}, \widehat{\Gamma}]$	$[u^*; v^*; \widehat{Z^*}, \widehat{\Gamma}]$	$0.5(L_{P_{Im}}^0 + L_{P_{Im}}^2)$
$[\mathbf{t}; \theta\mathbf{u}]$	/	$[0_{3\times1}; 0_{3\times1}]$	$0.5(L_{Pos}^0 + L_{Pos}^2)$

III. MPPI Control Strategy for Visual Servoing

In this section, we present the control strategy of our proposed sampling-based *MPC* approach (*MPPI*) for visual servoing systems; then, we state the mathematical formulation of *MPPI* in the presence of constraints such as the visibility, three-dimensional, and control constraints.

A. Review of MPPI

The *MPPI* control strategy is a sampling-based and derivative-free optimization method to MPC that can be easily applied in *online* to the real system, without requiring the first- or second-order approximation of the system dynamics and quadratic approximation of the objective functions. The control cycle of *MPPI* is described in Fig. 1. At each time-step Δt, *MPPI* samples thousands of control trajectories from the system dynamics using a Graphics Processing Unit (GPU) to ensure a *real-time* implementation.

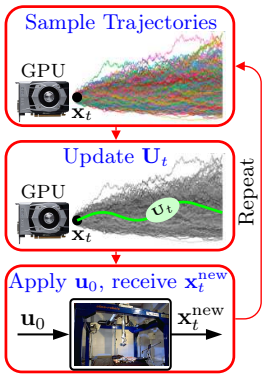

Fig. 1: *MPPI* control loop.

Afterward, based on the parallel nature of sampling, each of these trajectories is individually executed and then evaluated according to its expected cost. In sequential, the optimal control sequence $\mathbf{U}$, over a finite prediction time-horizon $t_p \in \{0, 1, 2, \ldots, T-1\}$, is updated based on a weighted average cost over these generated trajectories, where $\mathbf{U} = (\mathbf{u}_0, \mathbf{u}_1, \ldots, \mathbf{u}_{T-1}) \in \mathbb{R}^{m\times T}$, and $T \in \mathbb{R}^+$ refers to the number of timesteps. Finally, the first control $\mathbf{u}_0$ is applied to the system, while the remaining control sequence of length $T-1$ is slid-down to be used for providing a warm-starting to the optimization at the next time-step.

Let $\delta\mathbf{u}_t \in \mathbb{R}^m$ be a zero-mean Gaussian noise vector with a variance of $\Sigma_{\mathbf{u}}$, i.e., $\delta\mathbf{u}_t \sim \mathcal{N}(\mathbf{0}, \Sigma_{\mathbf{u}})$, where $\delta\mathbf{u}_t$ represents the random noise associated with the commanded control input $\mathbf{u}_t$ to the system. Then, the control input is defined by the exploring update $\mathbf{v}_t = \mathbf{u}_t + \delta\mathbf{u}_t$. Suppose that the number of the samples (namely, trajectories or rollouts) drawn from the discrete-time dynamics system, $\mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{v}_t)$, is K, where $\mathbf{x}_t \in \mathbb{R}^n$ denotes the state of the system at time t. Moreover, let $\tilde{S}(\tau_{t,k}) \in \mathbb{R}^+$ be the *cost-to-go* of the k^{th} trajectory from time t onward. Then, based on the detailed derivation given in [11] and well-summarized in [12], the optimal control sequence $\{\mathbf{u}_t\}_{t=0}^{T-1}$ can be readily updated using the following iterative update law:

$$\mathbf{u}_t \leftarrow \mathbf{u}_t + \frac{\sum_{k=1}^{K} \exp\left(-(1/\lambda)\tilde{S}(\tau_{t,k})\right) \delta\mathbf{u}_{t,k}}{\sum_{k=1}^{K} \exp\left(-(1/\lambda)\tilde{S}(\tau_{t,k})\right)} \tag{2}$$

where $\lambda \in \mathbb{R}^+$ is so-called the inverse temperature which determines the level of the selectiveness of the weighted average. We defined the *cost-to-go* of each trajectory τ over the predefined prediction time-horizon as:

$$\tilde{S}(\tau) = \phi(\mathbf{x}_T) + \sum_{t=0}^{T-1} \tilde{q}(\mathbf{x}_t, \mathbf{u}_t, \delta\mathbf{u}_t) \tag{3}$$

in which $\phi(\mathbf{x}_T)$ refers to the terminal cost. Whilst $\tilde{q}(\mathbf{x}_t, \mathbf{u}_t, \delta\mathbf{u}_t)$ denotes the instantaneous running cost, which composed of the sum of state-dependent running cost $q(\mathbf{x}_t)$ and quadratic control cost, and is defined as follows:

$$\tilde{q} = \underbrace{q(\mathbf{x}_t)}_{\textit{State-dep.}} + \underbrace{\frac{(1-\nu^{-1})}{2}\delta\mathbf{u}_t^T R\delta\mathbf{u}_t + \mathbf{u}_t^T R\delta\mathbf{u}_t + \frac{1}{2}\mathbf{u}_t^T R\mathbf{u}_t}_{\textit{Quadratic Control Cost}}$$

where $R \in \mathbb{R}^{m\times m}$ is a positive definite control weight matrix, and $\nu \in \mathbb{R}^+$ is so-called the exploration noise which determines how aggressively *MPPI* explores the state-space. The impact of changing ν has been studied in [12].

B. MPPI-VS Control Strategy

From the relation $\dot{s}(t) = L_{\mathbf{s}} \cdot \mathbf{v}_c$, a discrete-time model can be approximated using the Newton–Euler method as

$$\mathbf{s}(t+1) = \mathbf{s}(t) + \widehat{\mathbf{L}}_{\mathbf{s}}(t)\mathbf{v}_c(t)\Delta t \tag{4}$$

where $\mathbf{s}(t)$ defines the state vector of the model, composed of a set of n_p 2D image points in *MPPI-IBVS*, a set of n_p 3D-points in *MPPI-3DVS*, or by a pose vector in *MPPI-PBVS*. Similarly, the approximated interaction matrix is given either by $\widehat{\mathbf{L}}_{\mathbf{s}} = \widehat{\mathbf{L}}_{\boldsymbol{P_{Im}}} \in \mathbb{R}^{2n_p \times 6}$, $\widehat{\mathbf{L}}_{\boldsymbol{P}} \in \mathbb{R}^{3n_p \times 6}$ or $\widehat{\mathbf{L}}_{\boldsymbol{Pos}} \in \mathbb{R}^{6\times 6}$ for *MPPI-{IBVS,3DVS,PBVS}* control schemes. *MPPI-VS* control strategy directly uses the estimation of the interaction matrix. In *CASE #0* and #3 of the approximate $\widehat{\mathbf{L}}_{\boldsymbol{P_{Im}}}$ where the depth Z_i of each point-like feature needs to be estimated, we use the discrete-time model of the 3D-point features given in (4) for predicting the future evolution of the object depths $Z_i(t)$, assuming that, at each time-step Δt, the initial depth $Z_i(0)$ of a set of n_p point-like features is known.

*1) **Handling Visibility and 3D Constraints**:* One of the most attractive features of *MPPI*, compared to the classical *MPC*, is its capability of coping easily with the hard and soft constraints. More precisely, a *large-weighted* indicator function can be employed as part of the state-dependent running cost function $q(\mathbf{x})$ for handling the constraints. In *MPPI-VS* control scheme, the instantaneous state-dependent running cost function, $q(\mathbf{x}) \equiv q(\mathbf{s})$, is defined as

$$q(\mathbf{s}_t) = q_1(\mathbf{s}_t) + q_2(\mathbf{s}_t) \tag{5}$$

where $q_1(\mathbf{s}_t) = (\mathbf{s}_t - \mathbf{s}^*)^{\mathrm{T}} Q(\mathbf{s}_t - \mathbf{s}^*)$ denotes the state-dependent cost which is a simple quadratic cost for enforcing the current state $\mathbf{s}_t$ to reach its desired value $\mathbf{s}^*$. $q_2(\mathbf{s}_t)$ refers to an indicator function used for handling the visibility constraints (expressed as $\mathbf{s}_{\min} \leq \mathbf{s}_t \leq \mathbf{s}_{\max}$, where $\mathbf{s}_{\min} = [u_{\min}; v_{\min}]$ and $\mathbf{s}_{\max} = [u_{\max}; v_{\max}]$ refer to the lower and upper ranges, in pixels, of the image point coordinates) and the three-dimensional constraints such as workspace or joint limits. The definition of $q_2(\mathbf{s})$ depends on the control scheme. In *MPPI-IBVS* control scheme, q_2 is formulated as $q_2(\mathbf{s}_k) = 10^7 C_1 + 10^5 C_2$, where C_1 and C_2 are Boolean variables that are used to heavily penalize each trajectory that violates the visibility and 3D constraints, respectively. For instance, C_1 is turned on (i.e., $C_1 = 1$) if, at any time, $\mathbf{s}$ exceeds its given bounds, i.e., $C_1 = 1 \Leftrightarrow \exists \mathbf{s} |\ \big((\mathbf{s} > \mathbf{s}_{\max}) \vee (\mathbf{s} < \mathbf{s}_{\min})\big)$. We can also add the second term, if needed, for constraining the future evolution of the 3D-point features, namely, $C_2 = 1 \Leftrightarrow \exists \boldsymbol{P}_i |\ \big((\boldsymbol{P}_i > \boldsymbol{P}_{\max}) \vee (\boldsymbol{P}_i < \boldsymbol{P}_{\min})\big)\, \forall i = 1, \ldots, n_p$. To name just a few, this term can be added to ensure that the camera remains in the workspace, simply by limiting its position in the workspace. The special case of camera retreat, present in large rotation around the optical axis, can easily be taken into account. C_2 will be active if at any time the Z position of the camera exceeds a threshold $Z_{\max}$, i.e., $C_2 = 1 \Leftrightarrow \exists Z_i |\ (Z_i > Z_{\max})$. In *MPPI-PBVS* control scheme, the controller might produce input that would ultimately lead to that some visual features may leave the camera's Field of View (FoV). This is mainly because the control law is explicitly expressed in Cartesian space and there is no direct control to the visual features on the image plane. Therefore, we propose two methods to guarantee that the object always remains within the camera's FoV. Concerning the first proposed method, the indicator function q_2 is defined as a *large-weighted* exponential penalty function, as follows:

$$q_2 = \beta \sum_{i=1}^{n_p} \left(e^{-\alpha\left(x_{\max} - |x_i|\right)} + e^{-\alpha\left(y_{\max} - |y_i|\right)} \right) \tag{6}$$

so that the visibility constraints are constantly satisfied which can be written as $|p_i| \leq p_{\max}\, \forall i = 1, \ldots, n_p$, where $p_i = [x_i; y_i] = [\frac{X_i}{Z_i}; \frac{Y_i}{Z_i}] \in \mathbb{R}^2$ denotes the normalized Cartesian coordinates of an image point, $p_{\max} = [x_{\max}; y_{\max}]$, $p_{\min} = -p_{\max}$ are the minimum and maximum bounds, in meters, of the point feature projected on the normalized image plane. β and α are positive scalar variables. In the second method, we proposed an alternative solution based on augmenting the 3D-point features within the nominal state vector. In further words, we combine both *MPPI-PBVS* and *MPPI-3DVS* control schemes. Thus, in this case, we have $\mathbf{s} = [\boldsymbol{Pos}; \{\boldsymbol{P_i}\}_i] = \left[{}^{c^*}\mathbf{t}_c; \theta\mathbf{u}; \{X_i; Y_i; Z_i\}_i\right] \in \mathbb{R}^{(6+3n_p)}$, whilst $\widehat{\mathbf{L}}_{\mathbf{s}} = \left[\widehat{\mathbf{L}}_{\boldsymbol{Pos}}; \widehat{\mathbf{L}}_{\boldsymbol{P}}\right] \in \mathbb{R}^{(6+3n_p)\times 6}$. In the present scheme, the penalty function $q_2(\mathbf{s})$ is replaced by a *large-weighted* quadratic state-dependent cost function which modulates how fast the current state vector of the 3D-point features $\boldsymbol{P}$ converges to its desired one $\boldsymbol{P}^*$. Therefore, the running cost function $q(\mathbf{s})$ is reformulated as

$$q(\mathbf{s}) = \underbrace{w_1 \sum_{i=0}^{5} (\mathbf{s}_i - \mathbf{s}_i^*)^2}_{:=q_1(\mathbf{s}),\, \mathbf{s}_i \in \boldsymbol{Pos},\, \mathbf{s}_i^* \in \boldsymbol{Pos}^*} + \underbrace{w_2 \sum_{i=6}^{n-1} (\mathbf{s}_i - \mathbf{s}_i^*)^2}_{:=q_2(\mathbf{s}),\, \mathbf{s}_i \in \boldsymbol{P},\, \mathbf{s}_i^* \in \boldsymbol{P}^*} \tag{7}$$

where $\mathbf{s}^* = [\boldsymbol{Pos}^*; \boldsymbol{P}^*]$, $n = 3n_p + 6$ is the length of the state vector $\mathbf{s}$, whereas $w_1, w_2 \in \mathbb{R}^+$ refer to the cost weighting of q_1 and q_2, respectively. It should be noted that assigning very high weights, particularly for w_2, is sufficient for enforcing the visual features to stay within the image plane, without imposing additional constraints.

*2) **Handling Control Constraints**:* Most robotic systems have constraints on the actuators (range or velocity limitations). These constraints are known as control constraints which are considered to be hard constraints that should not be violated. In *MPPI* control strategy, there exists two ways for handling the control constraints. First, they can be implemented as a natural part of the running cost by adding an appropriate term to penalize all the trajectories that violate the constraints. However, the common issue associated with this way is that the control constraints acting as soft constraints, and, accordingly, it is difficult to ensure that the control input always remains within its allowed bounds. Therefore, we propose an alternative method based on pushing the control constraints into the dynamics system [14], meaning that $\mathbf{x}_{t+1} = f\big(\mathbf{x}_t, g(\mathbf{v}_t)\big)$, where $g(\mathbf{v}_t)$ is an element-wise clamping function that is used to restrict the control input $\mathbf{v}_t \sim \mathcal{N}(\mathbf{u}_t, \Sigma_{\mathbf{u}})$ to remain within a given range, for all

samples drawn from the dynamics system. Thus, $g(\mathbf{v})$ can be defined as $g(\mathbf{v}) = \max\left(\mathbf{v}_{\min}, \min(\mathbf{v}, \mathbf{v}_{\max})\right)$, where $\mathbf{v}_{\min}$ and $\mathbf{v}_{\max}$ are the lower and upper bounds of the control input. The major advantage is that the problem formulation of *MPPI*, which takes into account control constraints, is converted into an unconstrained one, without violating the constraints and affecting the convergence of the *MPPI* algorithm as $g(\mathbf{v})$ has an impact only on the dynamics system.

IV. Simulation-Based Evaluation

We conduct extensive simulations considering a 6-DoF Cartesian robot with an *eye-in-hand* camera configuration to evaluate and demonstrate the potential advantages of our proposed control strategy with a comparison to the classical schemes (*C-IBVS*, *C-3DVS*, and *C-PBVS*).

A. Simulation Settings and Performance Metrics

We validated the *MPPI-VS* control strategy as well as the classical schemes using four points (i.e., $n_p = 4$) in the image plane. These four dots, forming a square in the *xy*-plane, have the following coordinates in the reference frame $\mathcal{F}_o$: $P_1 = [-0.1; -0.1; 0]$, $P_2 = [0.1; -0.1; 0]$, $P_3 = [0.1; 0.1; 0]$, and $P_4 = [-0.1; 0.1; 0]$ [m]. During all the simulations, it is assumed that the *MPPI* algorithm runs with a time horizon t_p of 3.5 s, a control frequency of 50 Hz (i.e., $T = 175$), and generates 2000 samples each time-step Δt with an exploration variance ν of 1000. Moreover, the control weighting matrix R is set to $\frac{1}{2}\lambda\Sigma_{\mathbf{u}}^{-1}$, assuming that the random noise associated with the control input has a variance of $\Sigma_{\mathbf{u}} = \mathrm{Diag}\,(0.02, 0.01, 0.01, 0.02, 0.02, 0.01)$. The remaining parameters are defined in Table I.

TABLE I: *MPPI-VS* Parameters

Scheme	Parameter	Value	Parameter	Value
MPPI-IBVS	λ	100	Q	$2.5\mathbf{I}_8$
MPPI-3DVS	λ	10^{-2}	Q	$35\mathbf{I}_{12}$
MPPI-PBVS	λ	10^{-3}	Q	$35\mathbf{I}_6$
	β, α (6)	150, 10^3	w_1, w_2 (7)	35, 150

The inverse temperature λ is set to a high value in *MPPI-IBVS* scheme, and low values in both *MPPI-3DVS* and *MPPI-PBVS* control schemes. It is important to bear in mind that fine-tuning λ is mainly based on the state-dependent running cost function $q(\mathbf{s})$. The real-time *MPPI-VS* algorithm is executed on an NVIDIA GeForce GTX 1080 Ti desktop GPU. All the control schemes were developed using Visual Servoing Platform (ViSP) [15] integrated with the Robot Operating System framework.

*1) **Initial Camera Configurations**:* 120 initial camera configurations were randomly extracted from a uniform distribution within the robot's workspace, with a guarantee that the robot kinematics has initially the capability of reaching these generated poses, and the visual features are initially located within the camera's FoV.

*2) **Desired Camera Configurations**:* We considered two different desired poses of the camera, all expressed in the reference frame $\mathcal{F}_o$ attached to the object with z-axis pointing downward. The first desired camera pose is given by ${}^o\boldsymbol{Pos}^*_{c_1} = [{}^o\boldsymbol{t}_c; \theta\mathbf{u}] = [0, 0, -0.75, 0, 0, 0]^T$ in ([m], [deg]). We have also defined the second camera desired configuration ${}^o\boldsymbol{Pos}^*_{c_2} = [0.076, 0.202, -0.727, 10, -10, -15]^T$ in ([m], [deg]) in such a way that the object lies in the top-right corner of the image, to demonstrate how robust the proposed control strategy is against violating the system constraints particularly visibility constraints.

*3) **Performance Metrics**:* The visual servoing task is considered to be successful if the following metrics are satisfied:

$\mathcal{R}_{\text{LM}}$: The camera does not reach a local minimum (i.e., $\mathcal{R}_{\text{LM}} = 0$) during the simulations. Such configuration corresponding to a local minimum occurs when $\mathbf{v}_c = 0$ and ${}^o\boldsymbol{Pos}_c \neq {}^o\boldsymbol{Pos}^*_c$. Mathematically, we used the mean-squared error (*MSE*) as a metric that measures the positioning error ϵ, which can be formulated as $MSE = \frac{1}{n}\mathbf{e}^T\mathbf{e}$ in which $\mathbf{e} = {}^o\boldsymbol{Pos}_c - {}^o\boldsymbol{Pos}^*_c$ and $n = 6$. Thence, we considered that the local minimum is completely avoided if and only if $MSE_1 = \frac{2}{n}\mathbf{e}_1^T\mathbf{e}_1 < \epsilon_1$ and $MSE_2 = \frac{2}{n}\mathbf{e}_2^T\mathbf{e}_2 < \epsilon_2$, where $\mathbf{e}_1$ (in [m]) and $\mathbf{e}_2$ (in [rad]) refer to the translational and rotational errors between ${}^o\boldsymbol{Pos}_c$ and ${}^o\boldsymbol{Pos}^*_c$, respectively. During the simulations, we set the two thresholds as $\epsilon_1 = 10^{-5}$ and $\epsilon_2 = 10^{-4}$.

$\mathcal{R}_{\text{JL}}$: The robot joint limits are avoided (i.e., $\mathcal{R}_{\text{JL}} = 0$). If the robot reaches one of the joint limits, the velocity of each joint will be set to zero by the low level robot controller.

$\mathcal{P}_{\text{out}}$: The considered visual features always remain within the camera's FoV (i.e., $\mathcal{P}_{\text{out}} = 0$).

Finally, the number of successful servoing tasks, with respect to the total initial camera configurations, is denoted as $\mathcal{N}_{\text{success}}$, while $\mathcal{S}_{\text{rate}}$ indicates the success rate.

B. Intensive Simulation Results

Six different tables (II:VII) summarize the overall performance of each control scheme individually, given the 120 initial camera configurations. In total, we conduct a set of 24 different tests to evaluate and compare the performance of *MPPI-VS* control scheme with classical visual servoing. The robustness, the parameters influence and the capacity to handle different constraints are also studied.

TABLE II: Prediction Influence and Comparison

Test No.	Control Scheme	$\mathcal{R}_{\text{LM}}$	$\mathcal{P}_{\text{out}}$	$\mathcal{R}_{\text{JL}}$	$\mathcal{N}_{\text{success}}$	$\mathcal{S}_{\text{rate}}$
Test 1	*MPPI (CASE #0)*	0	0	1	119	99.16%
Test 2	*MPPI (CASE #1)*	0	0	29	91	75.8%
Test 3	*MPPI (CASE #2)*	0	0	54	66	55%
Test 4	*MPPI (CASE #3)*	0	0	6	114	95%
Test 5	*MPPI (CASE #0)*	0	0	0	120	100%
Test 6	*Classic (CASE #0)*	0	0	37	83	69.17%
Test 7	*Classic (CASE #0)*	0	0	36	84	70%

*1) **Prediction Process Influence and Comparison with C-IBVS**:* In the four first tests (see Table II), we analyzed the

impact of the image prediction process on the performance of the *MPPI-IBVS* control strategy by taking into account the four cases of the approximate interaction matrix $\widehat{\mathbf{L}}_{\boldsymbol{P_{Im}}}$. It can be clearly seen that the prediction process exerts a high influence as *MPPI* provides better performance in *CASE* #0 and #3, in which the 2D visual features and their depth information are assumed to be estimated each iteration. As expected, the worst performance is achieved when $\widehat{\mathbf{L}}_{\boldsymbol{P_{Im}}}$ is assumed to be constant, as depicted in Test 3.

More accurately, for Test 1, we conducted 3 trials. For all the trials, our proposed control scheme succeeded in completing 119 out of 120 tasks/configurations, while there existed only one initial camera configuration led the robot to reach one of its joint limits. This failure case can be easily tackled by increasing the prediction horizon as illustrated in Test 5 where we adopted $t_p = 5\,\mathrm{s}$ and $K = 1000$ instead of $t_p = 3.5\,\mathrm{s}$ and $K = 2000$.

In order to illustrate the superiority of *MPPI-IBVS*, intensive simulations of the C-IBVS control scheme are carried out in Test 6 and 7 in which we set λ_s to 0.5 and 0.2, respectively. As anticipated, the success rate of visual servoing tasks was reduced to 70%, with 36-$\mathcal{R}_{\text{JL}}$ failure cases. In fact, these failure cases occurred due to the fact that their corresponding initial camera configurations have large rotations around the camera optical axis as well as their positions along the z-axis are much closer to the minimum allowable limit of the third joint.

*2) **MPPI Parameters Influence and Robustness**:* In Table III is presented the influence of *MPPI* parameters as well as the robustness to modeling errors. From Test 8 to 10, the influence of both control input updates $\Sigma_{\mathbf{u}}$, prediction horizon t_p, and number of sampled trajectories K is individually studied. In Test 8, we adopted $\Sigma_{\mathbf{u}} = \mathrm{Diag}\,(0.009, 0.009, 0.009, 0.03, 0.03, 0.02)$, while in Test 9 t_p is set to 1 s. Finally, we set K to 100 in Test 10. In general, the intensive simulations demonstrate that the impact of either having short-time horizons t_p or changing the control input updates $\Sigma_{\mathbf{u}}$ is appreciably higher than the influence of decreasing the number of samples K, as the success rate $\mathcal{S}_{\text{rate}}$ in Tests 8 and 9 is significantly lower than that in Test 10. Concerning the quality of successful servoing tasks, it is interesting to notice that assigning very low values to t_p and K affects the convergence rate[1] and quality of the trajectories in both the image and 3D space. The robustness of the *MPPI-IBVS* is then evaluated with respect to the camera modeling errors and measurement noise. Due to the lack of space, only the test combining all the errors is given (see Test 11). First, a white noise generated from a uniform distribution is added to the visual features and their 3D information (i.e., depth estimation) with a maximum error of ± 1 pixel for 2D features and $\pm 0.5\,\mathrm{cm}$ for 3D points. In addition, errors on the intrinsic parameters of the camera are also added: $+30\,\%$ in f, $\pm 20\,\%$ in ρ_u and ρ_v (with $f_u = \frac{f}{\rho_u}$ and $f_v = \frac{f}{\rho_v}$), $\pm 15\,\%$ in u_0 and v_0. As can be seen, our proposed control scheme behaves perfectly as there exists a maximum of 4-$\mathcal{R}_{\text{JL}}$ failure cases that occurred, demonstrating its robustness.

TABLE III: ***MPPI*** Parameters Influence and Robustness

Test No.	Control Scheme	$\mathcal{R}_{\text{LM}}$	$\mathcal{P}_{\text{out}}$	$\mathcal{R}_{\text{JL}}$	$\mathcal{N}_{\text{success}}$	$\mathcal{S}_{\text{rate}}$
Test 8	*MPPI (CASE #0)*	0	0	19	101	84.17%
Test 9	*MPPI (CASE #0)*	0	0	35	85	70.8%
Test 10	*MPPI (CASE #0)*	0	0	7	113	94.17%
Test 11	*MPPI (CASE #0)*	0	0	4	116	96.67%

*3) **Handling Constraints**:* In Table IV are presented the results when handling constraints. In Test 12, we repeated the simulations of Test 1 under the assumption that the maximum control input of the camera velocity screw $\mathbf{v}_{\max}$ is limited to $0.5\,\mathrm{m/s}$ for the translational speed and $0.3\,\mathrm{rad/s}$ for the rotational speed, while the minimum control input $\mathbf{v}_{\min}$ equals to $-\mathbf{v}_{\max}$. The obtained results of the 105 successful tasks demonstrate that our proposed control scheme performs perfectly with a high capability of handling the control constraints. Note that the 15-$\mathcal{R}_{\text{JL}}$ failure cases can be easily avoided and tackled by involving the 3D constraints, especially joint limits constraints, as a part of the *MPPI-IBVS* optimization problem.

TABLE IV: Handling Control and Visual Constraints

Test No.	Control Scheme	$\mathcal{R}_{\text{LM}}$	$\mathcal{P}_{\text{out}}$	$\mathcal{R}_{\text{JL}}$	$\mathcal{N}_{\text{success}}$	$\mathcal{S}_{\text{rate}}$
Test 12	*MPPI (CASE #0)*	0	0	15	105	87.5%
Test 13	*MPPI (CASE #0)*	0	0	25	95	79.17%
Test 14	*Classic (CASE #0)*	0	0	42	78	65%

In all tests where *MPPI-IBVS* is utilized, the simulations are carried out taking into account the visibility constraints, which are defined by the following inequalities:

$$\begin{bmatrix} u_{\min} = 0 \\ v_{\min} = 0 \end{bmatrix} \leq \mathbf{s}_t \leq \begin{bmatrix} u_{\max} = 640 \\ v_{\max} = 480 \end{bmatrix}, \text{in [pixels]}. \quad (8)$$

Additionally, to better evaluate the capability of handling the constraints, the simulations of Test 1 are repeated in Test 13, considering ${}^{o}\boldsymbol{Pos}^{*}_{c_2}$ as the desired configuration instead of ${}^{o}\boldsymbol{Pos}^{*}_{c_1}$. Moreover, in Test 14, we repeated the simulations of *C-IBVS* given in Test 6 with respect to ${}^{o}\boldsymbol{Pos}^{*}_{c_2}$.

The intensive simulations presented in Table IV demonstrate the efficiency and capability of our proposed control strategy in coping easily with the visibility constraints, as the visual features always remain within the camera's FoV (i.e., $\mathcal{P}_{\text{out}} = 0$ for all tests).

*4) **3DVS Control Schemes**:* In Table V, the results of *MPPI-3DVS* are presented. Four intensive simulations are carried out in Tests 15 to 18 for assessing the performance of

TABLE V: *3DVS* Performance

Test No.	Control Scheme	$\mathcal{R}_{\text{LM}}$	$\mathcal{P}_{\text{out}}$	$\mathcal{R}_{\text{JL}}$	$\mathcal{N}_{\text{success}}$	$\mathcal{S}_{\text{rate}}$
Test 15	*MPPI* (${}^{o}\boldsymbol{Pos}^{*}_{c_1}$)	0	0	0	120	100%
Test 16	*Classic* (${}^{o}\boldsymbol{Pos}^{*}_{c_1}$)	0	0	4	116	96.67%
Test 17	*MPPI* (${}^{o}\boldsymbol{Pos}^{*}_{c_2}$)	0	0	5	115	95.83%
Test 18	*Classic* (${}^{o}\boldsymbol{Pos}^{*}_{c_2}$)	0	1	6	113	94.17%

[1]Note that the simulation time of Test 9 is adopted to be 180 s as the convergence rate of *MPPI*, in this case, is extremely slow.

both *MPPI-3DVS* and *C-3DVS* (with $\lambda_s = 0.5$), considering the two desired camera configurations and the *MPPI-3DVS* parameters listed in Table I. As excepted when 3D feature points are considered, the simulations show that both control strategies give very satisfactory results, where the maximum failure cases were 7 (6 for $\mathcal{R}_{\text{JL}}$ and only one for $\mathcal{P}_{\text{out}}$) occurred in Test 18.

*5) **Unconstrained MPPI-PBVS V.S. C-PBVS**:* Table VI shows the results of *MPPI-PBVS* without constraint. In the

TABLE VI: *PBVS* Performance in Unconstrained Case

Test No.	Control Scheme	$\mathcal{R}_{\text{LM}}$	$\mathcal{P}_{\text{out}}$	$\mathcal{R}_{\text{JL}}$	$\mathcal{N}_{\text{success}}$	$\mathcal{S}_{\text{rate}}$
Test 19	*MPPI* ($^o\boldsymbol{Pos}^*_{c_1}$)	0	45	0	75	62.50%
Test 20	*Classic* ($^o\boldsymbol{Pos}^*_{c_1}$)	0	25	0	95	79.17%
Test 21	*Classic* ($^o\boldsymbol{Pos}^*_{c_2}$)	0	21	0	99	82.50%

Tests 19 to 21, simulations are conducted to verify the performance of our proposed *PBVS* control scheme. We studied the behavior of *MPPI-PBVS* without applying the visibility constraints, together with the performance of the classical scheme. The obtained results illustrate that both control strategies produce control input ultimately leading to that some visual features leave the camera's FoV (i.e., $\mathcal{P}_{\text{out}} \neq 0$). This result was expected since no position-based controllers (classical or MPPI-based) provide any guarantee that the projections of the visual features remain within the camera's FoV. This is especially true in the case of MPPI due to the stochastic nature of the control sequence.

*6) **Constrained MPPI-PBVS**:* In Table VII are presented the results of *MPPI-PBVS* with constraints. To take into

TABLE VII: *PBVS* Performance in Constrained Case

Test No.	Control Scheme	$\mathcal{R}_{\text{LM}}$	$\mathcal{P}_{\text{out}}$	$\mathcal{R}_{\text{JL}}$	$\mathcal{N}_{\text{success}}$	$\mathcal{S}_{\text{rate}}$
Test 22	*MPPI* ($^o\boldsymbol{Pos}^*_{c_1}$, (6))	0	0	0	120	100%
Test 23	*MPPI* ($^o\boldsymbol{Pos}^*_{c_2}$, (6))	0	0	0	120	100%
Test 24	*MPPI* ($^o\boldsymbol{Pos}^*_{c_1}$, (7))	0	0	0	120	100%

account the difficulties mentioned above, we propose to handle visual constraints in *MPPI-PBVS* schemes. The two methods proposed in (6) and (7) to cope with the visibility constraints are used for the desired configuration $^o\boldsymbol{Pos}^*_{c1}$ (see Tests 22 and 24). Results show the validity and effectiveness of our proposed methods in handling visibility constraints, regardless of which desired camera configuration is considered (Test 22 and 23), as there exists neither $\mathcal{R}_{\text{JL}}$ nor $\mathcal{P}_{\text{out}}$ failure cases. It is also observed that despite *MPPI-PBVS* on the basis of (7) provides better motion in the image space while the 3D camera trajectory obtained by applying (6) is considerably shorter. It is noteworthy that the trajectories obtained by (7), particularly in the image plane, and satisfying the visibility constraints are mainly based on the assigned value to w_2. The results show that higher this value is, the better is the performance.

V. Conclusions and Future Work

In this work, we have proposed a *real-time* sampling-based *MPC* approach for predicting the future behavior of the VS system, without solving the online optimization problem which usually exceeds the real system-sampling time and suffers from the computational burden. There is no need for estimating the interaction matrix inversion or performing the pseudo-inversion in real-time. The proposed approach directly uses the approximate interaction matrix, i.e., it is an *inversion-free* control method. The visibility, three-dimensional (i.e., 3D), and control constraints as well as parametric uncertainties can be easily handled. Our next step will be the validation on a real Cartesian robot.

References

[1] S. Hutchinson, G. D. Hager, and P. I. Corke, "A tutorial on visual servo control," *IEEE transactions on robotics and automation*, vol. 12, no. 5, pp. 651–670, 1996.

[2] F. Chaumette and S. Hutchinson, "Visual servo control, part I: Basic approaches," *IEEE Robotics & Automation Magazine*, vol. 13, no. 4, pp. 82–90, 2006.

[3] ——, "Visual servo control. II. advanced approaches [tutorial]," *IEEE Robotics & Automation Magazine*, vol. 14, no. 1, pp. 109–118, 2007.

[4] F. Chaumette, S. Hutchinson, and P. Corke, "Visual Servoing," in *Handbook of Robotics, 2nd edition*, B. Siciliano and O. Khatib, Eds. Springer, 2016, pp. 841–866.

[5] F. Chaumette, "Visual Servoing," in *Computer Vision*, K. Ikeuchi, Ed. Springer International Publishing, October 2020.

[6] J. Gangloff, M. Mathelin, and G. Abba, "6 dof high speed dynamic visual servoing using gpc controllers," *Proceedings. 1998 IEEE International Conference on Robotics and Automation (Cat. No.98CH36146)*, vol. 3, pp. 2008–2013 vol.3, 1998.

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

[8] K. Hashimoto and H. Hidenori Kimura, "LQ optimal and nonlinear approaches to visual servoing," in *Visual Servoing: Real-Time Control of Robot Manipulators Based on Visual Sensory Feedback*, K. Hashimoto, Ed. World Scientific, October 1993, pp. 165–198.

[9] R. Dahmouche, N. Andreff, Y. Mezouar, O. Ait Aider, and P. Martinet, "Dynamic visual servoing from sequential regions of interest acquisition." *The International Journal of Robotics Research*, vol. 31, no. 4, pp. 520–537, Feb. 2012.

[10] H. J. Kappen, "Path integrals and symmetry breaking for optimal control theory," *Journal of statistical mechanics: theory and experiment*, vol. 2005, no. 11, p. P11011, 2005.

[11] G. Williams, A. Aldrich, and E. A. Theodorou, "Model predictive path integral control: From theory to parallel computation," *Journal of Guidance, Control, and Dynamics*, vol. 40, no. 2, pp. 344–357, 2017.

[12] I. S. Mohamed, G. Allibert, and P. Martinet, "Model predictive path integral control framework for partially observable navigation: A quadrotor case study," in *International Conference on Control, Automation, Robotics and Vision*, Shenzhen, China, Dec. 2020, pp. 196–203.

[13] E. Malis, "Improving vision-based control using efficient second-order minimization techniques," in *IEEE International Conference on Robotics and Automation*, vol. 2, New Orleans, LA, USA, Apr. 2004, pp. 1843–1848.

[14] G. Williams, N. Wagener, B. Goldfain, P. Drews, J. M. Rehg, B. Boots, and E. A. Theodorou, "Information theoretic MPC for model-based reinforcement learning," in *IEEE International Conference on Robotics and Automation*, Singapore, May 2017, pp. 1714–1721.

[15] É. Marchand, F. Spindler, and F. Chaumette, "Visp for visual servoing: a generic software platform with a wide class of robot control skills," *IEEE Robotics & Automation Magazine*, vol. 12, no. 4, pp. 40–52, 2005.