Decoupled Model Predictive Control for Path Following on Complex Surfaces

Joao Cavalcanti Santos, Lénaïc Cuau, Philippe Poignet, Nabil Zemiti

Decoupled Model Predictive Control for Path Following on Complex Surfaces

João Cavalcanti Santos[1], Lenaïc Cuau[1], Philippe Poignet[1] and Nabil Zemiti[1]

***Abstract*—The present letter proposes a predictive path following control (PPFC) that controls 5 degrees of freedom (DoFs) of the end-effector while the remaining (decoupled) translational DoF should be controlled by an external controller. This PPFC is particularly useful for the path following on surfaces with geometric uncertainties such that the external controller can be independently designed to manage the interaction between the tool and the surface. Therefore, the proposed strategy turns out to be a versatile control scheme that can be integrated with external controllers designed for applications such as robotic surface finishing, welding and 3D printing on complex surfaces. The corresponding optimal control problem (OCP) considers mainly the positioning and orientation errors, tangential velocity and control input amplitudes. The proposed PPFC is validated experimentally in the context of robotic 3D bio-printing. A 7-DoF redundant manipulator equipped with a distance sensor is used to handle a print head through the desired print path. The distance measurements are used by an external controller to correct the printed layer height. The obtained accuracy is consistent with the repeatability of the used manipulator and computation time is compatible with high frequency controllers.**

***Index Terms*—Optimization and Optimal Control; Medical Robots and Systems; Motion Control**

I. INTRODUCTION

THE motion control schemes taking as reference a curve within a robot workspace can be broadly divided into two groups: position tracking and path following. The former takes a time parameterized reference, *i.e.* a specific desired robot state is determined at each sampling time. This methodology is extensively used for the motion control of industrial manipulators [1], [2]. In contrast, path following control strategies do not follow a temporal law. Typically, the control scheme aims at converging to the desired path (which is a subset of the robot workspace) along with a desired velocity profile. This kind of strategy is largely used in the control of mobile robots [3], [4]. In applications in which time parametrization is not necessary, the advantages of path following are well-known. Since this approach drops the temporal law, path following controllers are, in general, less likely to lead to saturated control inputs and result in smoother convergence [5]–[7].

Model predictive control (MPC) is a suitable approach for the design of path following controllers [8]–[11]. Thanks to its predictive nature, a MPC strategy is able to find an optimal set of control inputs anticipating changes within its predictive horizon [10], [11]. Moreover, system constraints can be explicitly considered in the formulation of the optimal control problem (OCP). This characteristic is of particular interest since the best performance is often obtained in the limits of the system capabilities [12]. Path following strategies taking advantages of these characteristics are referred to as predictive path following control (PPFC), *e.g.* [9]–[11].

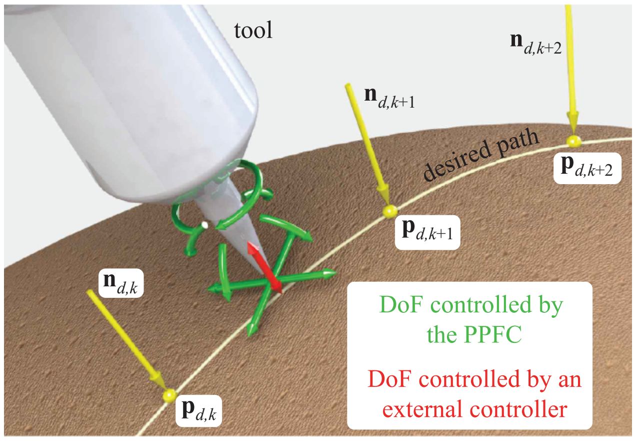

Fig. 1. Illustration of an application of the proposed strategy: 3D bio-printing in which the PPFC controls 5 DoFs of the tool while an external controller manages the distance between the extruder and the skin surface.

Among all the practical cases in which a robotic manipulator should follow a given trajectory, numerous applications involve a desired path projected on a specific surface, *e.g.* robotic welding, surface finishing, milling and 3D printing. Typically, the robot degrees of freedom (DoFs) related to the interaction between the tool and the surface should respond in a significantly different manner than the remaining DoFs. For instance, in the case of surface finishing, one may constrain the applied contact force perpendicular to the polished surface while the remaining DoFs should follow a given path and velocity profile. Solutions to deal with such an issue include the classic hybrid force/position control [1, Section 9.7], adaptive variable impedance [13] and neuro-adaptive control [14].

Hybrid force/position control, as in the early work [15], decouples the DoFs in which position tracking and force control should be applied. This approach has the advantage that the control scheme responsible for tracking the desired path can be designed independently from the controller managing the interaction with the surface (henceforth referred to as *external controller*), as illustrated in Figure 1 for the case studied in this letter. Nevertheless, most of the studies addressing the integration of different control schemes along different DoFs apply position tracking approaches [14], [16].

The design of *path following* control strategies for robotic manipulators capable of integrating different control schemes

Manuscript received: September, 16, 2022; Revised December, 15, 2022; Accepted January, 16, 2023. This paper was recommended for publication by Editor Lucia Pallottino upon evaluation of the Associate Editor and Reviewers' comments. This work was supported by the French project ANR-21-ASM2-0001 – Bloc-Print 2

[1]The authors are with the LIRMM, Univ Montpellier, CNRS, France {cavalcanti,cuau,poignet,zemiti}@lirmm.fr

Digital Object Identifier (DOI):

along different DoFs is scarcely addressed in the state of the art [17], [18]. Dahroug *et al.* propose in [17] a visual servoing controller able to follow a desired path keeping a constant remote center of motion. These different aspects are integrated thanks to the application of a task priority strategy. Similarly, Wen and Pagilla introduce in [18] a procedure in which an optimal path is determined based on a sequence of check points. The obtained path is used in an overall scheme integrating path following and force control. The introduced strategy is used for surface polishing. Both studies [17], [18] were focused in their corresponding applications. An extension of these schemes to different cases applying a general external controller is not addressed. Most importantly, they have in common that their path following controllers use Frenet-Serret frames [5] in order to compute tangential and transversal strategies but without using MPC. This leaves room for the aforementioned improvements obtained with predictive controllers. To the best of the authors' knowledge, the existing PPFC strategies do not integrate an external controller along a particular set of DoFs.

The main contribution of the present letter is the formulation of a PPFC strategy that can be integrated to an external controller managing independently the displacements along the direction normal to a given surface of interest. The proposed PPFC controls 5 tool DoFs while the remaining translational DoF normal to the surface (referred to as decoupled DoF) is controlled by the external controller. As a result, the obtained overall controller can perform path following on a surface with geometric uncertainties in which the behavior of the decoupled DoF controls the interaction (such as distance or force) between the tool and the surface. Since the external control can be designed independently, the interaction between the tool and the surface can be adapted to different applications, *e.g.* robotic surface finishing, welding and 3D printing. In order to perform the path following, the proposed OCP considers the tangential velocity along with the positioning and orientation errors with respect to the reference curve. The proposed OCP has the particularity that the positioning error with respect to a direction normal to the surface of interest is disregarded, since this DoF is controlled by the external controller.

A numerical optimization algorithm is proposed in order to meet the real-time constraints on the solution of the OCP. Applying such an algorithm, the strategy is validated experimentally in the context of 3D bio-printing. The printing path is generated on an initially unknown skin surface using a RGB-D camera. In this application, the surface geometric uncertainties are mainly caused by the errors related to the calibration between the camera coordinate frame and the robot base frame. As a consequence, the corresponding errors on the distance between the extruder and the skin surface for the theoretical printing path would hinder the appropriate deposition of the material. Similarly to conventional fused deposition modeling (FDM) 3D printing, the distance between the extruder and the surface of interest plays a particularly important role on the final result. Accordingly, the distance between the extruder and the skin surface are corrected with an external controller using a dedicated distance sensor. Satisfactory positioning error and computation time are obtained.

The remainder of this letter is organized as follows. After formulating the addressed problem in Section II, Section III introduces the PPFC strategy, detailing the main aspects considered in the corresponding OCP. The numerical solution of this optimization problem is discussed in Section IV. The experimental validation is detailed in Section V and conclusions are drawn in Section VI.

II. PRELIMINARIES

As depicted in Figure 1, this letter addresses a motion control problem in which the desired path is projected on a complex surface. The desired path consists of a sequence of desired tool positions and a sequence of desired tool orientations. These sequences are denoted as $\mathcal{P}_d = \{\mathbf{p}_{d,1},\, \mathbf{p}_{d,2}, \dots, \mathbf{p}_{d,N}\}$ and $\mathcal{N}_d = \{\mathbf{n}_{d,1},\, \mathbf{n}_{d,2}, \dots, \mathbf{n}_{d,N}\}$. The strictly positive integer N represents the number of reference points while vectors $\mathbf{p}_{d,k},\, \mathbf{n}_{d,k} \in \mathbb{R}^3$ determine the desired tool pose and orientation, respectively.

Typically, $\mathbf{n}_{d,k}$ is a unit vector normal to the surface at point $\mathbf{p}_{d,k}$. As a result, one may note that a tuple $\{\mathbf{p}_{d,k},\, \mathbf{n}_{d,k}\}$ constrains 5 DoFs of the tool. The remaining sixth DoF (the tool rotation around each $\mathbf{n}_{d,k}$) is considered as a redundant DoF. This is the case, for instance, for robotic welding, sanding, surface finishing and 3D printing.

Consider a n-DoF manipulator with joint positions given by $\mathbf{q} \in \mathbb{R}^n$, joint velocities $\dot{\mathbf{q}} \in \mathbb{R}^n$ and $n \geqslant 6$. The proposed control strategy takes as state vector $\mathbf{x} = \begin{bmatrix}\mathbf{q}^T & \dot{\mathbf{q}}^T\end{bmatrix}^T$ and aims at managing these states in order to follow the desired path $\{\mathcal{P}_d,\, \mathcal{N}_d\}$.

III. DECOUPLED PPFC

After introducing the system model in Section III-A, the PPFC algorithm is discussed in Section III-B.

A. Model

In order to reduce the real-time computational burden, the control input is considered as the robot joint accelerations $\mathbf{u} = \ddot{\mathbf{q}}$. Similar approaches are commonly used in the state of the art, *e.g.* [11]. Accordingly, it is considered that an internal controller is able to track constant joint accelerations for sufficiently small sampling periods Δt. Therefore, the resulting discrete-time dynamic system determines the $(i+1)^{th}$ state $\mathbf{x}_{i+1}$ based on the i^{th} state $\mathbf{x}_i$ and control input $\mathbf{u}_{i+1} \in \mathbb{R}^n$ according to

$$\begin{aligned}\mathbf{x}_{i+1} = \begin{bmatrix}\mathbf{q}_{i+1}\\ \dot{\mathbf{q}}_{i+1}\end{bmatrix} &= \begin{bmatrix}\mathbf{q}_i + \dot{\mathbf{q}}_i\,\Delta t + \mathbf{u}_{i+1}\,\left({}^{\Delta t^2}\!/_2\right)\\ \dot{\mathbf{q}}_i + \mathbf{u}_{i+1}\,\Delta t\end{bmatrix}\\ &= \mathbf{A}\,\mathbf{x}_i + \mathbf{B}\,\mathbf{u}_{i+1}\end{aligned} \tag{1}$$

with constant matrices $\mathbf{A} \in \mathbb{R}^{2n\times 2n}$ and $\mathbf{B} \in \mathbb{R}^{2n\times n}$. The considered model is hence a linear time-invariant system. Accordingly, for a given initial state $\mathbf{x}_0 = \begin{bmatrix}\mathbf{q}_0^T & \dot{\mathbf{q}}_0^T\end{bmatrix}^T$, the i^{th} state vector after the application of a sequence of control inputs $\mathbf{u}_1, \dots, \mathbf{u}_i$ can be written as

$$\mathbf{x}_i = \mathbf{A}^i\,\mathbf{x}_0 + \begin{bmatrix}\mathbf{A}^{i-1}\mathbf{B} & \mathbf{A}^{i-2}\mathbf{B} & \dots & \mathbf{B}\end{bmatrix}\begin{bmatrix}\mathbf{u}_1\\ \vdots\\ \mathbf{u}_i\end{bmatrix} \tag{2}$$

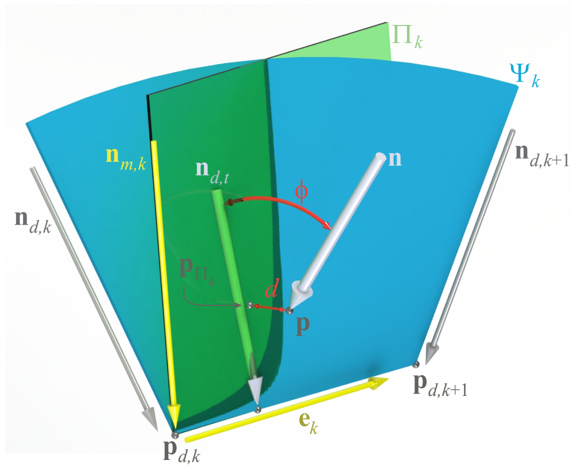

Fig. 2. Main notation used in the design of the PPFC.

Consider also that the kinematic model of the manipulator is known, such that the tool position $\mathbf{p} \in \mathbb{R}^3$ and orientation $\mathbf{n} \in \mathbb{R}^3$ can be computed applying the forward kinematics for a given $\mathbf{q}$. As discussed in Section II, note that the unit vector $\mathbf{n}$ represents uniquely the tool axis, *i.e.* the DoF corresponding to the rotation around $\mathbf{n}$ is considered as redundant in the present model. The tool linear $\dot{\mathbf{p}}$ and angular $\boldsymbol{\omega}$ velocities are computed based on the differential kinematics through $\begin{bmatrix}\dot{\mathbf{p}}^T & \boldsymbol{\omega}^T\end{bmatrix}^T = \mathbf{J}(\mathbf{q})\,\dot{\mathbf{q}}$, where $\mathbf{J} \in \mathbb{R}^{6\times n}$ is the manipulator jacobian matrix. This matrix is computed as a function of the joint positions $\mathbf{q}$ for each sampling period.

B. Control Scheme

The proposed path following control is formulated as an MPC in which, for each controller cycle, an OCP is solved. Considering an actual measured state $\mathbf{x}_0$, the next states $\mathbf{x}_1, \ldots, \mathbf{x}_{h_p}$ can be predicted using (2) for a given sequence of control inputs $\boldsymbol{\mu} = \begin{bmatrix}\mathbf{u}_1^T & \ldots & \mathbf{u}_{h_p}^T\end{bmatrix}^T$. The strictly positive integer h_p is called prediction horizon. The solution of the OCP leads to an optimal $\boldsymbol{\mu}^*$ that minimizes a cost function J_{h_p}. This function should summarize the desired behavior of the system. The design of J_{h_p} is discussed in the remainder of this section.

The OCP cost function $J_{h_p}(\boldsymbol{\mu}, \mathbf{x}_0, \mathbf{u}_0)$ is a sum of stage costs $\ell(i, \boldsymbol{\mu}, \mathbf{x}_0, \mathbf{u}_0)$ considering the control inputs and the i^{th} predicted state throughout the prediction horizon:

$$J_{h_p}(\boldsymbol{\mu}, \mathbf{x}_0, \mathbf{u}_0) = \sum_{i=1}^{h_p} \ell(i, \boldsymbol{\mu}, \mathbf{x}_0, \mathbf{u}_0), \tag{3}$$

where the stage cost $\ell(i, \boldsymbol{\mu}, \mathbf{x}_0, \mathbf{u}_0)$ is given by

$$\begin{aligned}\ell(i, \boldsymbol{\mu}, \mathbf{x}_0, \mathbf{u}_0) = \ell_g(\mathbf{q}_i) + \ell_v(\mathbf{q}_i, \dot{\mathbf{q}}_i) + \\ k_u \|\mathbf{u}_i\|^2 + k_{du}\|\mathbf{u}_i - \mathbf{u}_{i-1}\|^2 + k_{\dot{\mathbf{q}}}\|\dot{\mathbf{q}}_i\|^2,\end{aligned} \tag{4}$$

with positive scalars k_u, $k_{d,u}$ and $k_{\dot{\mathbf{q}}}$. The last three terms in (4) aim at smoothing the trajectory by minimizing the joint velocities and their derivatives. Vectors $\mathbf{q}_i$ and $\dot{\mathbf{q}}_i$ in (4) are obtained from (2) for given $\mathbf{x}_0$ and $\boldsymbol{\mu}$.

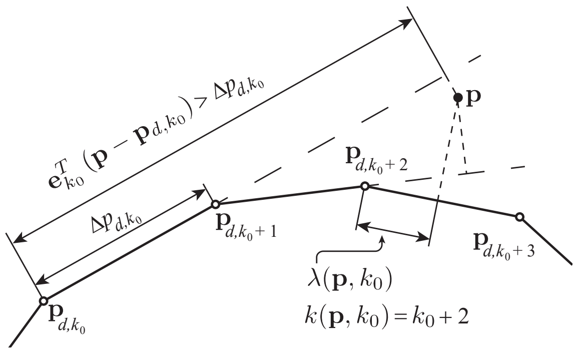
$\mathbf{e}_{k_0}^T(\mathbf{p} - \mathbf{p}_{d,k_0}) > \Delta p_{d,k_0}$
$\mathbf{p}$
$\mathbf{p}_{d,k_0+2}$
$\Delta p_{d,k_0}$
$\mathbf{p}_{d,k_0+1}$
$\mathbf{p}_{d,k_0+3}$
$\lambda(\mathbf{p}, k_0)$
$k(\mathbf{p}, k_0) = k_0 + 2$
$\mathbf{p}_{d,k_0}$

Fig. 3. Illustration of the steps involved in Algorithm 1.

The terms $\ell_g(\mathbf{q}_i)$ and $\ell_v(\mathbf{q}_i, \dot{\mathbf{q}}_i)$ in (4) deal with the positioning error and the tool velocity, respectively. As illustrated in Figure 1, the PPFC does not control the translation along the tool axis $\mathbf{n}$. Therefore, regarding the tool translation, the OCP should consider the positioning error of $\mathbf{p}$ with respect to the two axes perpendicular to $\mathbf{n}$. Additionally, the alignment error ϕ related to the orientation of $\mathbf{n}$ itself also should be considered. These two aspects are considered in $\ell_g(\mathbf{q}_i)$ and illustrated in Figure 2.

The desired path is considered as a linear interpolation between consecutive $\mathbf{p}_{d,k}$ and $\mathbf{p}_{d,k+1}$. For sufficiently dense $\mathcal{P}_d$ and $\mathcal{N}_d$ (as defined in Section II), the minimization of the joint velocities and their derivatives in (4) leads to a smooth trajectory taking this linearly interpolated path.

In order to compute the error on the tool position $\mathbf{p}$, one should first determine which segment $\{\mathbf{p}_{d,\hat{k}}, \mathbf{p}_{d,\hat{k}+1}\}$ should be considered. To this end, the closest segment to $\mathbf{p}$ is obtained using the Algorithm 1, *cf.* Figure 3.

Algorithm 1 Computation of $\hat{k}(\mathbf{p}, k_0)$ and $\lambda(\mathbf{p}, k_0)$

Input: k_0, $\mathbf{p}$, $\mathbf{p}_{d,j}$ for $j \in \{k_0, k_0+1, \ldots\}$
Output: $\hat{k}$, λ

```
1:  $k \leftarrow k_0$
2:  $\lambda \leftarrow \mathbf{e}_k^T(\mathbf{p} - \mathbf{p}_{d,k})$
3:  while $\lambda \geqslant \Delta p_{d,k}$ do
4:      $k \leftarrow k + 1$
5:      $\lambda \leftarrow \mathbf{e}_k^T(\mathbf{p} - \mathbf{p}_{d,k})$
6:      if $\lambda < 0$ then
7:          $\lambda \leftarrow 0$
8:          break
9:      end if
10: end while
11: $\hat{k} \leftarrow k$
```

For a given positive integer k_0, consider that the tracking of $\mathbf{p}_{d,k_0}$ has already been performed. Thus, the previous segments $\{\mathbf{p}_{d,k}, \mathbf{p}_{d,k+1}\} : k < k_0$ should not be considered in the algorithm. Let $\mathbf{e}_k$ be the unit vector

$$\mathbf{e}_k = \frac{\mathbf{p}_{d,k+1} - \mathbf{p}_{d,k}}{\|\mathbf{p}_{d,k+1} - \mathbf{p}_{d,k}\|} = \frac{\mathbf{p}_{d,k+1} + \mathbf{p}_{d,k}}{\Delta p_{d,k}}, \tag{5}$$

and $\Delta p_{d,k} = \|\mathbf{p}_{d,k+1} - \mathbf{p}_{d,k}\|$. The distance between $\mathbf{p}_{d,k_0}$ and the projection of $\mathbf{p}$ on the straight line defined by $\mathbf{p}_{d,k_0}$

and $\mathbf{p}_{d,k_0+1}$ can be written as $\mathbf{e}_{k_0}^T(\mathbf{p}-\mathbf{p}_{d,k_0})$. If the length $\mathbf{e}_{k_0}^T(\mathbf{p}-\mathbf{p}_{d,k_0})$ is greater than $\Delta p_{d,k_0}$, the next segment is considered. This procedure is repeated until $\mathbf{e}_k^T(\mathbf{p}-\mathbf{p}_{d,k}) < \Delta p_{d,k}$. It is possible to prove that the Algorithm 1 necessarily converges to a finite $\hat{k}$ under the assumption that there is a positive scalar ε such that every $\Delta p_{d,k} > \varepsilon$. Additionally, since the sequence $k = \{k_0,\, k_0+1, \dots\}$ is considered with an increasing order, the proposed algorithm is able to track self-intersecting curves. This ability is illustrated in the attached video. The convergence proof of Algorithm 1 and its capability to track self-intersecting curves are detailed in [19].

Similarly to the desired path, the desired orientations are interpolated between two consecutive $\mathbf{n}_{d,k}$ and $\mathbf{n}_{d,k+1}$. For this purpose, let the unit vector $\mathbf{r}_k \in \mathbb{R}^3$ and the positive scalar θ_k satisfy

$$\mathbf{n}_{d,k+1} = \mathbf{R}(\mathbf{r}_k, \theta_k)\,\mathbf{n}_{d,k}, \tag{6}$$

where $\mathbf{R}(\mathbf{r}_k, \theta_k)$ represents the rotation matrix defined by the axis of rotation $\mathbf{r}_k$ and angle θ_k. The desired orientation $\mathbf{n}_{d,t}$ for a given tool position $\mathbf{p}$, is defined as

$$\mathbf{n}_{d,t}(\mathbf{p}) = \mathbf{R}\left(\mathbf{r}_{\hat{k}}, c_{\hat{k}}\,\lambda\right)\,\mathbf{n}_{d,\hat{k}}, \tag{7}$$

with $\hat{k}$, λ computed using Algorithm 1 and $c_k = \theta_k/\Delta p_{d,k}$. Note that $\mathbf{n}_{d,t} = \mathbf{n}_{d,\hat{k}}$ if $\lambda = 0$, and $\mathbf{n}_{d,t} = \mathbf{n}_{d,\hat{k}+1}$ if $\lambda = \Delta p_{d,\hat{k}}$.

Ideally, since the PPFC does not control the translation along the tool axis $\mathbf{n}$, the tool position $\mathbf{p}$ should follow the blue surface Ψ_k depicted in Figure 2. This surface contains the straight line segment defined by $\mathbf{p}_{d,k},\, \mathbf{p}_{d,k+1}$ with interpolated orientation according to (7) for $0 \leqslant \lambda \leqslant \Delta p_{d,k}$. This surface can be written as

$$\begin{aligned}\Psi_k = \big\{&\mathbf{z}\in\mathbb{R}^3 \,|\, \exists\, \lambda,\, \rho \in \mathbb{R} : \\ &\mathbf{z} = \mathbf{p}_{d,k} + \lambda\,\mathbf{e}_k + \rho\,\mathbf{R}(\mathbf{r}_k, c_k\,\lambda)\,\mathbf{n}_{d,k}\big\}.\end{aligned} \tag{8}$$

Nevertheless, the distance between $\mathbf{p}$ and this complex surface cannot be obtained analytically and its computation (based on numerical optimization) would significantly increase the computational burden related to the evaluation of J_{h_p}. Accordingly, Ψ_k is approximated by the plane Π_k illustrated in Figure 2. This plane contains the segment $\mathbf{p}_{d,k},\, \mathbf{p}_{d,k+1}$ and is oriented according to an intermediary vector $\mathbf{n}_{m,k}$ defined as

$$\mathbf{n}_{m,k} = \frac{(\mathbf{I}-\mathbf{e}_k\,\mathbf{e}_k^T)\,\mathbf{R}(\mathbf{r}_k, \theta_k/2)\,\mathbf{n}_{d,k}}{\left\|(\mathbf{I}-\mathbf{e}_k\,\mathbf{e}_k^T)\,\mathbf{R}(\mathbf{r}_k, \theta_k/2)\,\mathbf{n}_{d,k}\right\|}, \tag{9}$$

where the term $(\mathbf{I}-\mathbf{e}_k\,\mathbf{e}_k^T)$ is used to insure that $\mathbf{n}_{m,k}$ is perpendicular to $\mathbf{e}_k$.

Therefore, denoting $\mathbf{E}_k = \begin{bmatrix}\mathbf{e}_k & \mathbf{n}_{m,k}\end{bmatrix}$, the projection of $\mathbf{p}$ in the plane Π_k can be computed as

$$\mathbf{p}_{\Pi,k} = \mathbf{p}_{d,k} + \mathbf{E}_k\,\mathbf{E}_k^T\,(\mathbf{p}-\mathbf{p}_{d,k}) \tag{10}$$

and the distance $d_{\hat{k}}$ in Figure 2 is written as

$$d_{\hat{k}} = \left\|(\mathbf{I}-\mathbf{E}_{\hat{k}}\,\mathbf{E}_{\hat{k}}^T)\,(\mathbf{p}-\mathbf{p}_{d,\hat{k}})\right\|. \tag{11}$$

Reminding that $\ell_g(\mathbf{q}_i)$ should quantify the positioning error of $\mathbf{p}$ and the alignment error of $\mathbf{n}$, the distance $d_{\hat{k}}$ and the scalar product $\mathbf{n}^T\mathbf{n}_{d,t}$ are used for these respective goals so that

$$\begin{aligned}\ell_g(\mathbf{q}) = k_d\, &\left\|(\mathbf{I}-\mathbf{E}_{\hat{k}}\,\mathbf{E}_{\hat{k}}^T)\,(\mathbf{p}-\mathbf{p}_{d,\hat{k}})\right\|^2 - \\ &k_a\,(\mathbf{n}^T\,\mathbf{n}_{d,t})^2,\end{aligned} \tag{12}$$

with positive scalar gains k_d and k_a.

Finally, the partial stage cost $\ell_v(\mathbf{q}, \dot{\mathbf{q}})$ should penalize the difference between the velocities along $\mathbf{e}_{\hat{k}}$ and the desired velocity $\dot{p}_d$. This is performed with

$$\ell_v(\mathbf{q}, \dot{\mathbf{q}}) = k_v\left(\mathbf{e}_{\hat{k}}^T\,\dot{\mathbf{p}} - \dot{p}_d\right)^2, \tag{13}$$

with a positive scalar k_v.

In summary, the proposed PPFC takes the following OCP:

$$\boldsymbol{\mu}^* = \arg\min_{\boldsymbol{\mu}}\; J_{h_p}(\boldsymbol{\mu}, \mathbf{x}_0, \mathbf{u}_0) \tag{14a}$$

$$\text{s.t. } \mathbf{q}_{min} \leqslant \mathbf{q}_i \leqslant \mathbf{q}_{max}, \quad i = 1,\dots,h_p \tag{14b}$$

$$-\dot{\mathbf{q}}_{max} \leqslant \dot{\mathbf{q}}_i \leqslant \dot{\mathbf{q}}_{max}, i = 1,\dots,h_p \tag{14c}$$

with J_{hp} defined in (3) and $\mathbf{q}_{min}$, $\mathbf{q}_{max}$, $\dot{\mathbf{q}}_{max}$ the minimum joint position, maximum joint position and maximum joint velocity, respectively. The feedback output from the control scheme is $\mathbf{u}_{fb}(\mathbf{x}_0, \mathbf{u}_0) = \mathbf{u}_{1,\boldsymbol{\mu}^*}$, *i.e.* the first n elements of $\boldsymbol{\mu}^*$. Matrices $\mathbf{J}_p$, $\mathbf{J}_r \in \mathbb{R}^{3\times n}$ are defined as submatrices of $\mathbf{J}$ such as $\mathbf{J} = \begin{bmatrix}\mathbf{J}_p^T & \mathbf{J}_r^T\end{bmatrix}^T$ and $\dot{\mathbf{p}} = \mathbf{J}_p\,\dot{\mathbf{q}}$, $\boldsymbol{\omega} = \mathbf{J}_r\,\dot{\mathbf{q}}$.

For a given actual state $\begin{bmatrix}\mathbf{q}_0^T & \dot{\mathbf{q}}_0^T\end{bmatrix}^T$, the desired joint accelerations $\mathbf{u}_{fb}$ computed through (14) can be used within a tracking control at joint level. Typically, desired joint velocities can be sent to the internal robot driver.

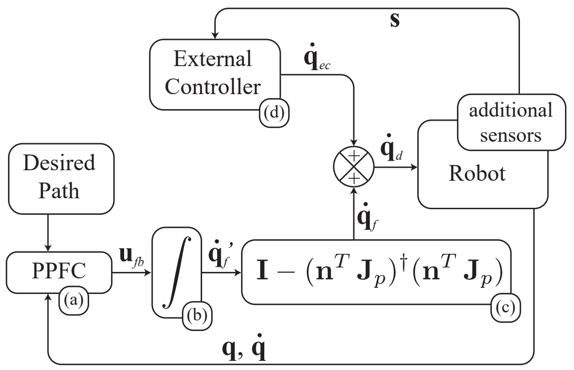

Fig. 4. Overall control scheme.

Figure 4 depicts the application of the proposed PPFC in an overall control scheme. The PPFC output of (a) is integrated over time in (b), leading to $\dot{\mathbf{q}}_f'$. One may note that partial stage costs ℓ_g and ℓ_v present constant values for positions and velocities varying along $\mathbf{n}_{m,\hat{k}}$. Therefore, the minimization of ℓ_g and ℓ_v should not lead to displacements along the direction normal to the surface. Nevertheless, the minimization of $\dot{\mathbf{q}}$ and its derivatives in (4) may induce non-null velocities along $\mathbf{n}$, *i.e.* $\mathbf{n}^T\mathbf{J}_p\,\dot{\mathbf{q}}_f' \neq 0$. For this reason, block (c) projects $\dot{\mathbf{q}}_f'$ within a subspace of $\mathbb{R}^n$, such that $\mathbf{n}^T\mathbf{J}_p\,\dot{\mathbf{q}}_f' = 0$, insuring that the tool velocity along $\mathbf{n}$ generated by $\dot{\mathbf{q}}_f$ is null. This procedure is equivalent to the application of a selection matrix (as in [18], for instance).

Regarding the control of the motion along $\mathbf{n}$, an external controller (d) computes $\dot{\mathbf{q}}_{ec}$ based on a given feedback signal $\mathbf{s}$. The measurements $\mathbf{s}$ may be obtained, for instance, with force or distance sensors. Note that $\dot{\mathbf{q}}_{ec}$ should generate uniquely tool velocities along $\mathbf{n}$, *i.e.* $\mathbf{n}\,\mathbf{n}^T\,\mathbf{J}_p\,\dot{\mathbf{q}}_{ec} = \mathbf{J}_p\,\dot{\mathbf{q}}_{ec}$ and $\mathbf{J}_r\,\dot{\mathbf{q}}_{ec} = \mathbf{0}$.

IV. NUMERICAL OPTIMIZATION

The applicability of a MPC scheme in a real-time robotic system depends on the computation time necessary to solve the corresponding OCP. One may note that the OCP (14) is nonlinear and may have a significant amount of arguments for a typical prediction horizon, *e.g.* $56 = n \times h_p$ for $n = 7$, $h_p = 8$ as in Section V. Therefore, a detailed discussion on the numerical solution of (14) is necessary. The present section proposes an optimization algorithm specifically designed for this goal.

The proposed algorithm described in Section IV-A is derived from the one developed in [20], in which the position tracking of cable-driven parallel robots is addressed. Since a gradient-based solver is used, the computation of the derivatives of J_{h_p} is necessary. The numerical estimation of these derivatives requires the computation of the overall cost function (3) at least $n \times h_p + 1$ times. This may significantly impair the efficiency of the controller. As a solution to this issue, Section IV-B details the analytical expressions for the derivatives of J_{h_p}.

A. Overall Algorithm

Consider given $\mathbf{x}_0$ and $\mathbf{u}_0$, such that the sought solution is $\boldsymbol{\mu}^*$ minimizing $J_{h_p}(\boldsymbol{\mu}, \mathbf{x}_0, \mathbf{u}_0)$, as in (14). For a given $\boldsymbol{\mu}$, (2) leads to predicted tool positions $\mathbf{p}_i$ for $i = 1, \dots, h_p$ and corresponding sequence of integers $\hat{\mathcal{K}} = \{\hat{k}_1, \dots, \hat{k}_{h_p}\}$ computed according to Algorithm 1. A particular numerical issue related to the computation of $\boldsymbol{\mu}^*$ is that variations of $\boldsymbol{\mu}$ would lead to variations of the integers $\hat{k}_i$, which results in discontinuous J_{h_p}. This would impair the efficiency of purely gradient based solvers. For that reason, the algorithm described in Algorithm 2 is used.

Algorithm 2 Overall numerical algorithm to solve (14)

```
Input: $\mathbf{x}_0$, $\mathbf{u}_0$ and initial guess $\boldsymbol{\mu}_p$
Output: $\boldsymbol{\mu}^*$
1:  $\boldsymbol{\mu}^* \leftarrow \boldsymbol{\mu}_p$
2:  loop
3:      Update $\hat{\mathcal{K}}$ for $\boldsymbol{\mu}^*$ using (2) and Algorithm 1
4:      Compute $\boldsymbol{\mu}^*$ as the solution of (14) for constant $\hat{\mathcal{K}}$
5:      if $\|\boldsymbol{\mu}^* - \boldsymbol{\mu}_p\| < \epsilon$ then
6:          break
7:      else
8:          $\boldsymbol{\mu}_p \leftarrow \boldsymbol{\mu}^*$
9:      end if
10: end loop
```

In short, the Algorithm 2 iteratively solves simplified versions of (14) in which the sequence $\hat{\mathcal{K}}$ is taken as constant. More precisely, the steps 3-10 of Algorithm 1 are bypassed in order to prevent the step $k \leftarrow k+1$, avoiding the variations on $\hat{\mathcal{K}}$. Accordingly, the optimization problem in Algorithm 2 (step 4) can be solved efficiently with, for instance, sequential quadratic programming (SQP) algorithms. It is worth noting that the convergence of Algorithm 2 is more reliable for reference points $\mathcal{P}_d$ that are sufficiently close to each other, reducing the variation of the cost function (14a) for successive iterations of the algorithm. The results presented in Section V were obtained with a SQP adapted from the solver used in [20]. This numerical algorithm applies an altered version of the nonlinear gradient project method described in [21, Section 18.6], *cf.* [22, Chapter 4]. Basic matrix operations were performed using tools provided by the Eigen C++ library [23].

B. Derivatives of J_{h_p}

This section presents the analytical derivatives of J_{h_p} for constant $\hat{\mathcal{K}}$. The analytical derivatives are obtained straightforwardly based on the stage cost described in Section III-B. The complete deduction of these expressions are omitted for the sake of conciseness.

Consider the functions $f : \mathbb{R}^{m_1} \to \mathbb{R}$ and $\mathbf{g} : \mathbb{R}^{m_1} \to \mathbb{R}^{m_2}$ for any strictly positive integers m_1 and m_2. For given input variable $\mathbf{y} \in \mathbb{R}^{m_1}$, the derivatives of such functions are denoted as

$$\frac{\partial}{\partial\,\mathbf{y}} f(\mathbf{y}) = \begin{bmatrix} \frac{\partial}{\partial\,y_1} f(\mathbf{y}) & \cdots & \frac{\partial}{\partial\,y_{m_1}} f(\mathbf{y}) \end{bmatrix} \text{ and}$$
$$\frac{\partial}{\partial\,\mathbf{y}} \mathbf{g}(\mathbf{y}) = \begin{bmatrix} \frac{\partial}{\partial\,y_1} g_1(\mathbf{y}) & \cdots & \frac{\partial}{\partial\,y_{m_1}} g_1(\mathbf{y}) \\ \vdots & \ddots & \vdots \\ \frac{\partial}{\partial\,y_1} g_{m_2}(\mathbf{y}) & \cdots & \frac{\partial}{\partial\,y_{m_1}} g_{m_2}(\mathbf{y}) \end{bmatrix}. \tag{15}$$

The gradient of the cost function in (14) is computed through

$$\frac{\partial}{\partial\,\boldsymbol{\mu}} J_{h_p}(\boldsymbol{\mu}, \mathbf{x}_0, \mathbf{u}_0) = \sum_{i=1}^{h_p} \frac{\partial}{\partial\,\boldsymbol{\mu}} \ell(i, \boldsymbol{\mu}, \mathbf{x}_0, \mathbf{u}_0). \tag{16}$$

Therefore, the gradient (16) can be computed by means of the derivatives of each $\ell(i, \boldsymbol{\mu}, \mathbf{x}_0, \mathbf{u}_0)$ with respect to $\boldsymbol{\mu}$. Denoting

$$\ell_{u,i} = \ell_u(i, \boldsymbol{\mu}, \mathbf{u}_0) = k_u\,\|\mathbf{u}_i\|^2 + k_{du}\|\mathbf{u}_i - \mathbf{u}_{i-1}\|^2,$$

the following expression can be deduced for $i > 1$:

$$\frac{\partial\,\ell_{u,i}}{\partial\,\boldsymbol{\mu}} = 2\Big[\mathbf{0}^T_{(i-2)\times n} \quad -k_{du}\,(\mathbf{u}_i - \mathbf{u}_{i-1})^T \quad \dots$$
$$k_u\,\mathbf{u}_i^T + k_{du}\,(\mathbf{u}_i - \mathbf{u}_{i-1})^T \quad \mathbf{0}^T_{(h_p-i-1)\times n}\Big]$$

with $\mathbf{0}_m = \begin{bmatrix} 0 & \dots & 0 \end{bmatrix}^T \in \mathbb{R}^m$. Similarly, for $i = 1$,

$$\frac{\partial\,\ell_{u,i}}{\partial\,\boldsymbol{\mu}} = \begin{bmatrix} 2\,k_u\,\mathbf{u}_1^T + k_{du}(\mathbf{u}_1 - \mathbf{u}_0)^T & \mathbf{0}^T_{(h_p-1)\times n} \end{bmatrix}.$$

The elements of $\ell(i, \boldsymbol{\mu}, \mathbf{x}_0, \mathbf{u}_0)$ that depend on the state variables are first derived with respect to $\mathbf{q}_i$ and $\dot{\mathbf{q}}_i$. Clearly,

$$\frac{\partial}{\partial\,\dot{\mathbf{q}}} k_{\dot{\mathbf{q}}}\|\dot{\mathbf{q}}\|^2 = 2\,k_{\dot{\mathbf{q}}}\,\dot{\mathbf{q}}^T. \tag{17}$$

Likewise, the derivatives of $\ell_{g,i} = \ell_g(\mathbf{q}_i)$ are given by

$$\begin{aligned}\frac{\partial \ell_{g,i}}{\partial \mathbf{q}_i} =& 2\, k_d\, d_{\hat{k}_i}\, \mathbf{w}_{\hat{k}_i}^T\, \mathbf{J}_p(\mathbf{q}_i) - \\ & 2\, k_a\, (\mathbf{n}^T\, \mathbf{n}_{d,t}) \left(\mathbf{n}^T \frac{\partial \mathbf{n}_{d,t}}{\partial \mathbf{q}_i} + \mathbf{n}_{d,t}^T \frac{\partial \mathbf{n}}{\partial \mathbf{q}_i} \right),\end{aligned} \tag{18}$$

with $\mathbf{w}_{\hat{k}_i} = \mathbf{p}_i - \mathbf{p}_{\Pi,\hat{k}_i} / \|\mathbf{p} - \mathbf{p}_{\Pi,\hat{k}_i}\|$ a unitary vector perpendicular to the plane $\Pi_{\hat{k}_i}$. Integer $\hat{k}_i$ is computed as a function of $\mathbf{p}_i$ using Algorithm 1 and $\mathbf{n}_{d,t}$ is obtained with (7). The derivatives of $\mathbf{n}_{d,t}$ and $\mathbf{n}$ can be written as

$$\begin{aligned}\frac{\partial}{\partial \mathbf{q}} \mathbf{n}_{d,t} &= \frac{\theta}{\Delta p_d} (\mathbf{r} \times \mathbf{n}_{d,t})\, \mathbf{e}^T\, \mathbf{J}_p(\mathbf{q}) \\ \frac{\partial}{\partial \mathbf{q}} \mathbf{n} &= -\mathbf{n} \times \mathbf{J}_r(\mathbf{q}).\end{aligned}$$

The derivatives of $\ell_{v,i} = \ell_v(\mathbf{q}_i, \dot{\mathbf{q}}_i)$ are given by

$$\frac{\partial \ell_{v,i}}{\partial \dot{\mathbf{q}}} = 2\, k_v\, (\mathbf{e}^T \dot{\mathbf{p}} - \dot{p}_d)\, \mathbf{e}^T \mathbf{J}_p \tag{19a}$$

$$\frac{\partial \ell_{v,i}}{\partial \mathbf{q}} = 2\, k_v\, (\mathbf{e}^T \dot{\mathbf{p}} - \dot{p}_d)\, \mathbf{e}^T \sum_{j=1}^{n} \frac{\partial \mathbf{j}_{p,j}}{\partial \mathbf{q}}\, \dot{q}_j \tag{19b}$$

where $\mathbf{j}_{p,j} \in \mathbb{R}^3$ is the j^{th} column vector of $\mathbf{J}_p$. Its derivatives

$$\frac{\partial \mathbf{j}_{p,j}}{\partial \mathbf{q}} = \begin{bmatrix} \frac{\partial \mathbf{j}_{p,j}}{\partial q_1} & \dots & \frac{\partial \mathbf{j}_{p,j}}{\partial q_n} \end{bmatrix}$$

have vector columns computed as

$$\frac{\partial \mathbf{j}_{p,j}}{\partial q_l} = \begin{cases} \mathbf{j}_{r,l} \times \mathbf{j}_{p,j}, & \text{if } j \geqslant l \\ \mathbf{j}_{r,j} \times \mathbf{j}_{p,l}, & \text{if } j < l \end{cases}$$

where $\mathbf{j}_{r,j}$ is the j^{th} column vector of $\mathbf{J}_r$.

For a given $\mathbf{v} \in \mathbb{R}^n$ representing a gradient with respect to $\mathbf{q}_i$, the multiplication $\mathbf{v}^T\, \partial\, \mathbf{q}_i / \partial\, \boldsymbol{\mu}$ considering the prediction (2) can be written as

$$\begin{aligned}\mathbf{v}^T \frac{\partial \mathbf{q}_i}{\partial \boldsymbol{\mu}} = (\Delta t^2) \Bigg[& \left(\frac{1}{2} + (i-1) \right) \mathbf{v}^T \quad \dots \\ & \left(\frac{1}{2} + (i-2) \right) \mathbf{v}^T \quad \dots \quad \frac{1}{2} \mathbf{v}^T \quad \mathbf{0}_{(h_p - i) \times n}^T \Bigg]\end{aligned} \tag{20}$$

Likewise, the multiplication $\mathbf{v}^T\, \partial\, \dot{\mathbf{q}}_i / \partial\, \boldsymbol{\mu}$ is computed using

$$\mathbf{v}^T \frac{\partial \dot{\mathbf{q}}_i}{\partial \boldsymbol{\mu}} = \Delta t \begin{bmatrix} \mathbf{v}^T & \dots & \mathbf{v}^T & \mathbf{0}_{h_p - i}^T \end{bmatrix}. \tag{21}$$

Replacing $\mathbf{v}^T$ in (20)-(21) by the derivatives in (17)-(19), the gradient of the stage cost is given by

$$\begin{aligned}\frac{\partial}{\partial \boldsymbol{\mu}} \ell(i, \boldsymbol{\mu}, \mathbf{x}_0, \mathbf{u}_0) = \frac{\partial \ell_{u,i}}{\partial \boldsymbol{\mu}} + \left(\frac{\partial \ell_{g,i}}{\partial \mathbf{q}_i} + \frac{\partial \ell_{v,i}}{\partial \mathbf{q}_i} \right) \frac{\partial \mathbf{q}_i}{\partial \boldsymbol{\mu}} + \\ \left(2\, k_{\dot{\mathbf{q}}}\, \dot{\mathbf{q}}_i{}^T + \frac{\partial \ell_{v,i}}{\partial \dot{\mathbf{q}}_i} \right) \frac{\partial \dot{\mathbf{q}}_i}{\partial \boldsymbol{\mu}}.\end{aligned}$$

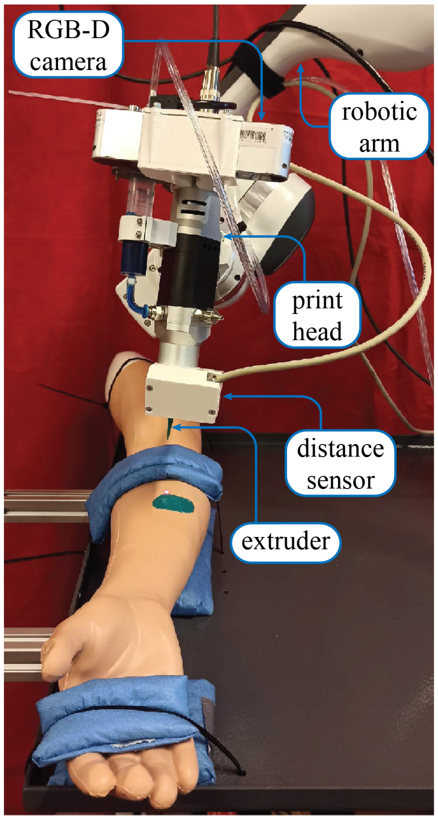

Fig. 5. Experimental set-up used to test robotic 3D bio-printing.

V. EXPERIMENTAL RESULTS

The experimental validation presented in this section is inspired by the application of the proposed PPFC for 3D robotic bio-printing used in the treatment of serious skin wounds. The goal is to deposit a skin substitute over the wound surface using a print head embedded on a robotic arm end-effector. The used set-up is depicted in Figure 5.

A Revopoint Acusense RGB-D camera is used to determine the three-dimensional geometry of the surface on which the skin substitute should be applied. Based on this three-dimensional geometry, a print path covering the whole wound can be generated. The print path consists of a sequence of desired tool positions $\mathbf{p}_{d,k}$ and the corresponding unit vectors $\mathbf{n}_{d,k}$ normal to the surface at position $\mathbf{p}_{d,k}$. The printing is performed using a 7-DoF Franka Emika Panda robotic arm to position a ViscoTec vipro-HEAD 5 print head along the desired path. The substitute skin is simulated using Pluronic F-127 + HBSS solution mixture.

The definition of the vectors $\mathbf{p}_{d,k}$ and $\mathbf{n}_{d,k}$ relies on the calibration of the camera coordinate system with respect to the robot base coordinate system. Using the materials described in this letter, such a procedure is subject to errors of ≈ 2 mm, *cf.* [24]. Indeed, errors with this magnitude affecting the distance between the extruder and the skin surface would play a critical role.

Using the set-up presented in Figure 5, this distance should be equal to 0.8 mm. Similarly to fused deposition modeling (FDM) 3D printing, variations on this distance affect significantly the quality of the printed surface and errors of 2 mm would lead to unacceptable issues. Accordingly, as depicted in Figure 5, an Acuity AR-100 distance sensor is used to monitor the distance between the extruder and the skin. This measurement is used by an external controller to move the extruder along its axis $\mathbf{n}$ in order to correct the distance to the skin surface. More precisely, the applied external control strategy consists in a simple proportional correction, with a tool translational velocity given by

$$\dot{\mathbf{p}}_{ec} = \mathbf{n}\, k_{lh}(h_m - h_d), \tag{22}$$

where h_m, h_d and k_{lh} are the measured layer height, desired layer height and a scalar positive gain, respectively. The joint velocities $\dot{\mathbf{q}}_{ec}$ can be obtained with a classic redundancy resolution method [25]. One may note that the external controller (22) does not have any feed-forward term and is based uniquely on the tracking error $h_m - h_d$. In spite of the simplicity of this controller, h_m converges asymptotically to a constant h_d if the positioning errors related to $\mathbf{n}$ normal to the surface are negligible. A feed-forward term is not necessary to obtain stability because a precise positioning of $\mathbf{n}$ results in translational velocities computed by the PPFC scheme that are tangential to the surface. This matter is discussed in detail in [19].

In addition to the external controller (22), the remaining tool DoFs are controlled with the predictive control scheme proposed in Section III-B. As a result, the desired path is followed with the introduced PPFC, while the external controller corrects the distance between the extruder and the skin surface. Since the controlled robotic arm has $n = 7$ DoFs and the positioning of the extruder given by the tuple $\{\mathbf{p}, \mathbf{n}\}$ represents 5 DoFs, it is worth noting that the proposed control strategy implicitly performs the corresponding redundancy resolution. Table I summarizes the used controller parameters in accordance with the notation introduced in Section III-B. The algorithm programmed in C++ was ran with an Intel(R) Core(TM) i7-7820HQ CPU @ 2.90GHz.

The desired print path is determined based on the three dimensional geometry of the skin obtained with the RGB-D camera. The printed surface covers $\approx 800\ \mathrm{mm}^2$, and leads to normal vectors $\mathbf{n}_{d,k}$ with maximal variation greater than 30°. Since the desired path consisting of points $\mathcal{P}_d$ has significant errors due to the camera calibration, the comparison between the original desired path and the printed one $\mathcal{P}$ would be meaningless. A printing performed with null error with respect to $\mathcal{P}_d$ would fail to deposit the substitute skin due to the inappropriate distance between the extruder and the skin. Therefore, in order to compare the desired and the printed path, a rigid transformation ${}^{d}\mathbf{T}_r$ is applied to the point cloud determined by $\mathcal{P}_d$. The rigid transformation ${}^{d}\mathbf{T}_r$ is obtained applying iterative closest point (ICP, [26]) in order to register the point clouds $\mathcal{P}$ and $\mathcal{P}_d$. The registered desired path $\mathcal{P}_r$ and the measured printed path $\mathcal{P}$ are depicted in Figure 6. Regarding the orientation of the extruder, Figure 7 shows the desired $\mathcal{N}_d$ and measured $\mathcal{N}$ orientations along with the corresponding angular error. A summary of the data related to the errors is presented in Table II. The sets $\mathcal{P}$ and $\mathcal{N}$ are obtained based on the forward kinematics.

TABLE I
CONTROL PARAMETERS

parameter	value	parameter	value
h_p	8	$\dot{p}_d$	6.0×10^{-3} mm/s
Δt	10.0 ms	k_{lh}	4.0
k_u	6.0×10^{-2}	k_d	8.0×10^{4}
k_a	6.0×10^{3}	k_v	6.0×10^{4}
$k_{\dot{\mathbf{q}}}$	6.0	k_{du}	1.0

TABLE II
SUMMARY OF THE POSITIONING AND ANGULAR ERRORS

	Compared sets	RMS	Std. Dev.
Positioning error	$\{\mathcal{P}, \mathcal{P}_r\}$	0.266 mm	0.114 mm
Angular error	$\{\mathcal{N}, \mathcal{N}_d\}$	0.595°	0.308°

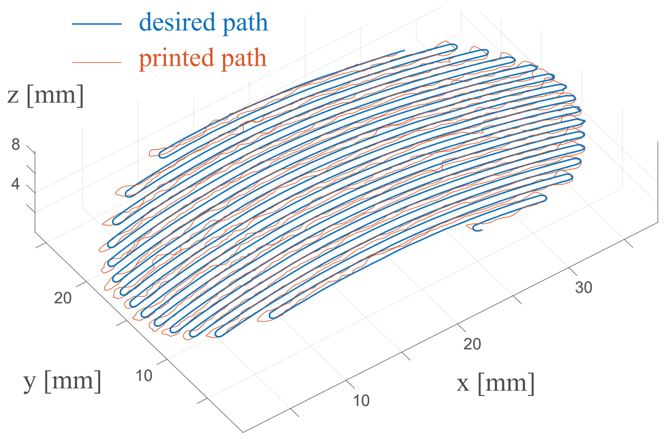

Fig. 6. Experimental results: comparison between desired and printed path.

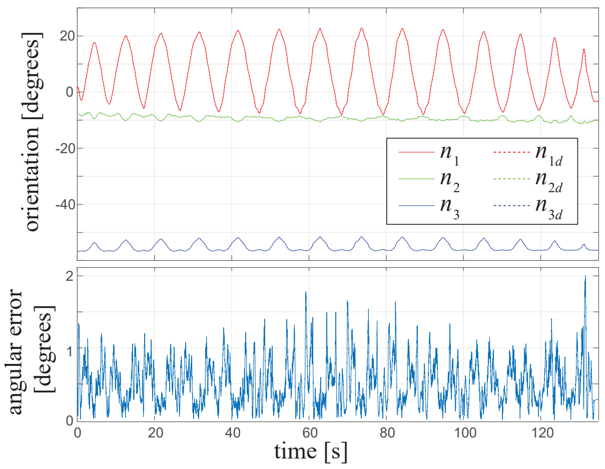

Fig. 7. Experimental results: comparison between desired and actual tool orientation.

It is important to highlight that the errors presented in Table II are consistent with the positioning precision obtained with a Panda Franka Emika robotic arm. This manipulator has a pose repeatability of 0.1 mm and path deviation of 1.25 mm (based on the ISO 9283 standard). Additionally, the final printed surface depicted in Figure 8 successfully covers the skin.

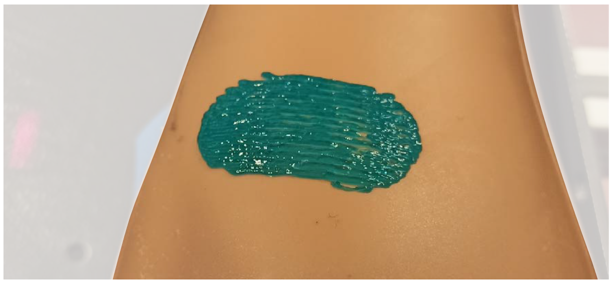

Fig. 8. Experimental results: final printed surface.

Regarding the computational burden, the average computing time involved in the solution of (14) is $1.05\ \mathrm{ms}$ with a standard deviation of $1.06\ \mathrm{ms}$. Recalling that this solution involves the numerical optimization of the nonlinear problem (14) taking as arguments 56 variables ($h_p \times n = 8 \times 7$), the computational efficiency of the proposed scheme is considered satisfactory. It is worth noting that the application of the analytical derivatives described in Section IV-B reduces $\approx 70\%$ of the computation time necessary to solve (14).

VI. Conclusions

The present letter introduced a PPFC scheme with a decoupled translational DoF aligned with the tool axis. The proposed strategy was conceived for the path following on surfaces with geometric uncertainties. Since the design of the external controller responsible for the management of the decoupled DoF is independent of the proposed PPFC, the behavior of the decoupled DoF can be adapted for different applications, *e.g.* robotic surface finishing and 3D printing on complex surfaces. A numerical algorithm was proposed for the solution of the corresponding optimal control problem. Experimental results in the context of robotic 3D bio-printing led to satisfying precision and computation time. Future works should analyze the stability of the obtained closed-loop system considering suboptimal control inputs and different optimization algorithms.

References

[1] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo, *Robotics: Modelling, Planning and Control*. Springer-Verlag London Limited, 2009, no. 4.

[2] F. L. Lewis, D. M. Dawson, and C. T. Abdallah, *Robot manipulator control: theory and practice*. CRC Press, 2003.

[3] P. B. Sujit and S. Saripalli, "Unmanned Aerial Vehicle Path Following," *IEEE Control Systems Magazine*, no. February, pp. 42–59, 2014.

[4] Y. Xu and S.-W. Au, "Stabilization and path following of a single wheel robot," *IEEE/ASME Transactions on Mechatronics*, vol. 9, no. 2, pp. 407–419, 2004.

[5] L. Lapierre and B. Jouvencel, "Robust nonlinear path-following control of an AUV," *IEEE Journal of Oceanic Engineering*, vol. 33, no. 2, pp. 89–102, 2008.

[6] R. J. Gill, D. Kulić, and C. Nielsen, "Spline path following for redundant mechanical systems," *IEEE Transactions on Robotics*, vol. 31, no. 6, pp. 1378–1392, 2015.

[7] J. Matschek, T. Bäthge, T. Faulwasser, and R. Findeisen, "Nonlinear Predictive Control for Trajectory Tracking and Path Following: An Introduction and Perspective," in *Handbook of Model Predictive Control*, 2019, pp. 169–198.

[8] R. Ritschel, F. Schrödel, J. Hädrich, and J. Jäkel, "Nonlinear model predictive path-following control for highly automated driving," *IFAC-PapersOnLine*, vol. 52, no. 8, pp. 350–355, 2019.

[9] T. Faulwasser and R. Findeisen, "Nonlinear Model Predictive Control for Constrained Output Path Following," *IEEE Transactions on Automatic Control*, vol. 61, no. 4, pp. 1026–1039, 2016.

[10] T. Faulwasser, T. Weber, P. Zometa, and R. Findeisen, "Implementation of Nonlinear Model Predictive Path-Following Control for an Industrial Robot," *IEEE Transactions on Control Systems Technology*, vol. 25, no. 4, pp. 1505–1511, 2017.

[11] R. Yang, L. Zheng, J. Pan, and H. Cheng, "Learning-based predictive path following control for nonlinear systems under uncertain disturbances," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 2854–2861, 2021.

[12] J. M. Maciejowski, *Predictive control: with constraints*. Pearson education, 2002.

[13] J. Duan, Y. Gan, M. Chen, and X. Dai, "Adaptive variable impedance control for dynamic contact force tracking in uncertain environment," *Robotics and Autonomous Systems*, vol. 102, pp. 54–65, 2018.

[14] Y. Karayiannidis, G. Rovithakis, and Z. Doulgeri, "Force/position tracking for a robotic manipulator in compliant contact with a surface using neuro-adaptive control," *Automatica*, vol. 43, no. 7, pp. 1281–1288, 2007.

[15] M. H. Raibert and J. J. Craig, "Hybrid Position/Force Control of Manipulators," *Journal of Dynamic Systems, Measurement, and Control*, vol. 103, no. 2, pp. 126–133, jun 1981.

[16] J. Pliego-Jiménez and M. A. Arteaga-Pérez, "Adaptive position/force control for robot manipulators in contact with a rigid surface with uncertain parameters," *European Journal of Control*, vol. 22, pp. 1–12, 2015.

[17] B. Dahroug, B. Tamadazte, and N. Andreff, "Visual servoing controller for time-invariant 3D path following with remote centre of motion constraint," *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 3612–3618, 2017.

[18] Y. Wen and P. R. Pagilla, "A novel 3D path following control framework for robots performing surface finishing tasks," *Mechatronics*, vol. 76, no. April, p. 102540, 2021.

[19] J. C. Santos, L. Cuau, P. Poignet, and N. Zemiti, "Supplementary Materials - Decoupled Model Predictive Control for Path Following on Complex Surfaces," jan 2023. [Online]. Available: https://hal-lirmm.ccsd.cnrs.fr/lirmm-03962764

[20] J. C. Santos, M. Gouttefarde, and A. Chemori, "A Nonlinear Model Predictive Control for the Position Tracking of Cable-Driven Parallel Robots," *IEEE Transactions on Robotics*, 2022.

[21] J. Nocedal and S. Wright, *Numerical optimization*. Springer Science and Business Media, 2006.

[22] J. C. Santos, "Model Predictive Tracking Control of Cable-Driven Parallel Robots : From Concept to Real-Time Validation," Thesis, Université Montpellier, oct 2020.

[23] G. Guennebaud, B. Jacob, and Others, "Eigen v3," http://eigen.tuxfamily.org, 2010.

[24] M. E. de Oliveira, H. G. Debarba, A. Lädermann, S. Chagué, and C. Charbonnier, "A hand-eye calibration method for augmented reality applied to computer-assisted orthopedic surgery," *International Journal of Medical Robotics and Computer Assisted Surgery*, vol. 15, no. 2, apr 2019.

[25] B. Siciliano, "Kinematic control of redundant robot manipulators: A tutorial," *Journal of Intelligent and Robotic Systems*, vol. 3, no. 3, pp. 201–212, 1990.

[26] P. J. Besl and N. D. McKay, "A method for registration of 3-D shapes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 2, pp. 239–256, 1992.