2024 European Control Conference (ECC)
June 25-28, 2024. Stockholm, Sweden

Contactless Surface Following with Acceleration Limits: Enhancing Robot Manipulator Performance through Model Predictive Control

Johan Ubbink[1,2,3], Ruan Viljoen[1,2], Erwin Aertbeliën[1], Wilm Decré[1], Joris De Schutter[1]

***Abstract*— In robotics, automating surface following is desirable for applications such as spray painting, inspection, and surface finishing. However, current surface following approaches focus on in-contact applications, where the robot is restricted to operate within a slow dynamic range. We introduce an approach tailored for contactless surface following, which leverages Model Predictive Control (MPC) to better exploit the full dynamic range of the robot while taking its acceleration limits into account. Our formulation introduces a surface model based on Radial Basis Function Networks (RBFN), and we show that when it is combined with local sensing and a surface estimator, it achieves significantly better performance compared to state-of-the-art approaches which rely on a local quadratic model. The approach was validated through extensive simulations, and it was found to reliably compute feasible control inputs within real-time. Through this research we aim to enable less conservative surface-following behaviour and bring MPC closer to real-life industrial applications in robotics.**

I. INTRODUCTION

Surface following, as depicted in Fig. 1, involves a robot manipulator moving a tool over a surface while maintaining a desired distance and orientation. This task is key in various industries, ranging from industrial applications like spray painting [1], inspection [2], and surface finishing [3], to even medical applications [4, 5].

For low-accuracy applications, a global surface model obtained from a depth camera or CAD file can be utilised for open-loop trajectory planning and execution [2]. However, this is not robust against calibration errors, occlusions, or disturbances during execution. In contrast, this paper focuses on high-accuracy applications, where feedback controllers are combined with local sensing to estimate and correct for disturbances, ensuring accurate and robust surface following.

These applications can be categorised as either in-contact or contactless, depending on whether the robot tool makes physical contact with the surface. While conceptually similar, in-contact motion is often slower due to noisy force sensors, friction, and potentially stiff contact between the robot and the environment which may result in large forces. In contrast, contactless applications allow for more dynamic motion.

Although in-contact surface following has been extensively studied [3–9], this paper aims to address the notable gap in research concerning contactless surface following. While many of the in-contact control approaches can be applied to contactless applications with minimal modifications, these approaches are not tailored to exploit the dynamic motion possible in contactless scenarios, resulting in unnecessarily conservative behaviour.

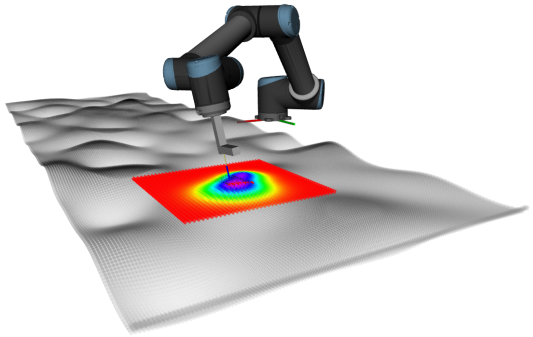


Fig. 1: **Surface following scenario.** The robot manipulator is controlled to maintain a desired distance and orientation with respect to the surface. For high-accuracy applications, a local sensor is used to measure and estimate the relative pose between the tool and the surface, with a feedback controller correcting any disturbances.

The dynamic motions possible with contactless surface following lead to high accelerations, which can cause the robot's internal motion controller to shut down, requiring operation stoppage and restart, with notable industrial repercussions. To address this, it is desirable to impose acceleration limits in the control law.

Research by Decré et al. [10] demonstrated that when controlling a robot subject to input constraints (such as velocity or acceleration limits), there is an advantage to incorporating a prediction horizon into the control law. However, the study assumed full knowledge of the environment; therefore, no online estimation was required. Moreover, the implementation was not fast enough for real-time applications.

More recent studies have investigated Model Predictive Control (MPC) for interacting with an environment [11–13]. Notably, Gold et al. [13] presented an approach where MPC is combined with a state estimator for estimating and controlling the force between the robot end-effector and the environment. Additionally, they demonstrated the real-time feasibility of the approach. However, these studies focused on in-contact applications, where the robot is restricted to a slower dynamic range, limiting the benefit of MPC. In fact, Gold et al. [13] recommended future research to explore more dynamic applications to fully leverage MPC. Hence, this paper investigates the use of MPC for the more dynamic application of contactless surface following.

This research was supported by project G0D1119N of the Research Foundation - Flanders (FWO - Flanders).

[1] All authors are with the Department of Mechanical Engineering, KU Leuven, and Flanders Make@KU Leuven, Leuven, Belgium.
[2] The first two authors contributed equally to this work.
[3] Corresponding author: johan.ubbink@kuleuven.be

When using MPC for contactless surface following, the choice of surface model becomes an important aspect. Due to the slower nature of in-contact surface following, having a good local approximation of the surface is typically sufficient. Therefore, state-of-the-art research [8] has predominantly relied on quadratic models to locally describe the surface. However, it is well known that quadratic models (or any polynomial model) may extrapolate poorly, which can be problematic for the MPC controller which predicts (and therefore extrapolates) into the future. Consequently, we propose a surface model which is better suited for the contactless surface following task.

Concretely, the main contributions of this paper are:

1) an MPC controller and surface estimator that can be combined with local sensing to provide good performance for the contactless surface following task while ensuring that the acceleration limits of the robot are not exceeded;
2) a surface model, based on Radial Basis Function Networks (RBFN) [14], which can be used to locally approximate the shape of the surface, while still providing good extrapolation capabilities. We show that this model achieves significantly better performance compared to a quadratic model.

The approach is validated through simulations involving randomly generated surfaces. Notably, the computation time of both the estimator and the controller are sufficiently fast for online usage, making it promising for real-world applications.

The remainder of this paper is structured as follows: Section II presents the surface modelling and estimation approach, while Section III covers the task specification and control formulation. Section IV is dedicated to the experimental validation, and Section V provides a conclusion and suggestions for future work.

II. SURFACE MODELLING AND ESTIMATION

This section presents the considered surface models and the approach used to estimate their parameters online from laser measurements.

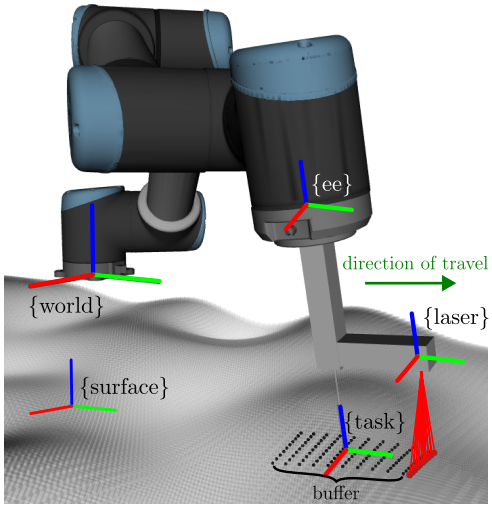
A. Surface model

In this paper, we assume the following surface model

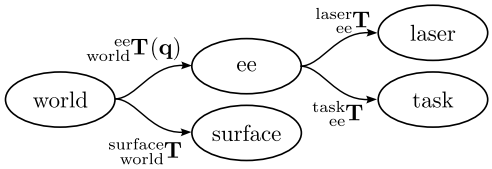
$$\mathrm{s}_{\boldsymbol{\theta}}(p_x, p_y) = p_z, \tag{1}$$

where $\mathrm{s}_{\boldsymbol{\theta}}$ represents the surface parameterised by $\boldsymbol{\theta}$, and (p_x, p_y, p_z) denotes a point on the surface. This point is defined in the *surface frame*, which is fixed to the *world frame*, as shown in Fig. 2. For the sake of simplicity, we consider an explicit as opposed to an implicit surface model. While this limits the type of surfaces that can be modelled adequately, it remains a valid approach for a wide variety of practical applications.

Two surface models are compared in the paper, as shown in Fig. 3. The first model is a quadratic model, selected because it has shown promising results for in-contact surface following [8]. Mathematically it can be expressed as



(a) Visualisation of surface following setup.



(b) Definition of reference frames.

Fig. 2: **Surface following setup.** A visualisation of the surface following setup (a), with (b) showing the respective reference frames, defined in Section II-A (*world* and *surface*), Section II-B (*ee* and *laser*), and Section III-B (*task*).

$$\mathrm{s}^{\mathrm{Q}}_{\boldsymbol{\theta}}(p_x, p_y) = a_1 + a_2 p_x + a_3 p_y + a_4 p_x p_y + a_5 p_x^2 + a_6 p_y^2. \tag{2}$$

Here, the surface is parameterised in terms of the surface weights, i.e. $\boldsymbol{\theta} = [a_1, a_2, a_3, a_4, a_5, a_6]^{\mathrm{T}}$. This surface representation has the desirable property that it is linear with respect to the parameters of the model, simplifying estimation.

As described in Section I, quadratic models may extrapolate poorly, which can be problematic for the MPC controller which predicts (and therefore extrapolates) into the future. For this reason, a second surface model is introduced based on an RBFN. This surface model is expressed as the weighted sum of N basis functions

$$\mathrm{s}^{\mathrm{RBFN}}_{\boldsymbol{\theta}}(p_x, p_y) = \sum_{i=1}^{N} a_i \cdot \phi_i(||(p_x, p_y) - c_i||), \tag{3}$$

with $a_i \in \mathbb{R}$ the weight associated with each basis and $c_i \in \mathbb{R}^2$ the centre points. We selected a Gaussian radial basis function kernel $\phi_i(r)$ with $r = ||(p_x, p_y) - c_i||$ and shape parameters $\epsilon_i \in \mathbb{R}$, expressed as

$$\phi_i(r) = e^{-(\epsilon_i r)^2}. \tag{4}$$

In this case, the model is parameterised in terms of the weights, the shape parameters, and the centre points, i.e.

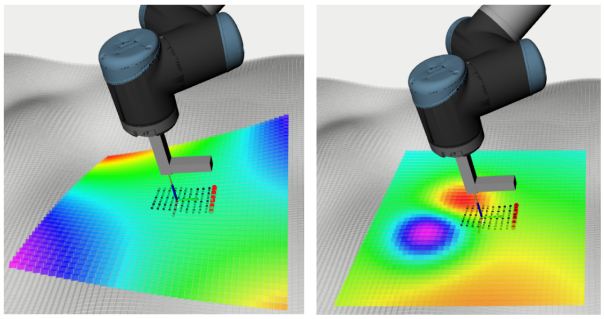


Fig. 3: **Two different surface models used.** The local approximation given by the surface model is shown (in colour) superimposed on the real surface (in gray). On the left is the quadratic model, while on the right the RBFN model. The RBFN provides a better extrapolation necessary for longer MPC prediction horizons.

$\boldsymbol{\theta} = [a_1 \dots a_N, \epsilon_1 \dots \epsilon_N, c_1 \dots c_N]^{\mathrm{T}}$. Therefore, this model is nonlinear with respect to the parameters of the basis functions, making the estimation more challenging. However, this surface model has the advantage that the output p_z smoothly tends to zero when evaluated away from the centre points (as opposed to the quadratic model which could go to infinity). This provides better extrapolation which is necessary for longer MPC prediction horizons.

B. Surface estimation

As shown in Fig. 2, the laser sensor is attached to the end-effector (*ee frame*) in front of the direction of travel. This placement aims to enhance the accuracy of surface measurements beneath the *task frame*, a frame fixed to the robot end-effector around which the motion of the robot will be controlled (to be described in Section III-B).

The laser sensor measures a line of surface points ${}_{\text{laser}}\mathbf{p}^{\text{s}}$ relative to its own *laser frame*. These points are transformed to the *surface frame*

$$ {}_{\text{surface}}\mathbf{p}^{s} = {}^{\text{laser}}_{\text{surface}}\mathbf{T}\, {}_{\text{laser}}\mathbf{p}^{s}, \tag{5} $$

where ${}^{\text{laser}}_{\text{surface}}\mathbf{T}$ represents the homogeneous transformation matrix between the laser frame and the surface frame, obtained using the robot's forward kinematics.

These points are kept in a rolling buffer of length L, as shown in Fig. 2(a). The length of this buffer is selected such that it roughly covers the area from the laser sensor to underneath the task frame. At each control interval, a least-squares optimisation problem yields the best-fit parameters $\boldsymbol{\theta}^*$ given the buffer of points, defined as

$$ \boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \sum_{i=1}^{L} [\mathrm{s}_{\boldsymbol{\theta}}(p_{x,i}, p_{y,i}) - p_{z,i}]^2 + \mu\, \boldsymbol{\theta}^{\intercal}\, \mathbf{S}\, \boldsymbol{\theta}\,. \tag{6} $$

Here, the selection matrix $\mathbf{S}$ is used to determine which of the parameters $\boldsymbol{\theta}$ are regularised, while μ adjusts the extent of this regularisation. For the quadratic model, all the parameters are regularised, while for the RBFN model, only the weights of the basis functions are regularised, not the centre points or the shape parameters. For both surface models, this regularisation can be interpreted as a prior belief that the surface should be more or less planar.

For the quadratic model, solving (6) is a linear least-squares problem. However, for the RBFN model, it becomes a nonlinear optimisation problem, which is solved iteratively using PANTR [15], an efficient nonlinear matrix-free solver. To ensure real-time feasibility, the maximum number of iterations at every timestep is limited to ten.

III. TASK SPECIFICATION AND CONTROL

The surface-following controller is designed following a *constraint-based approach*, where the idea is to specify task constraints (often referred to as task functions) which describe the desired behaviour of the task [16]. Then, a feedback controller (in our case utilising MPC) is implemented to regulate the task constraints to a desired value. This section presents the robot model, the task specification, and the MPC controller.

A. Robot model

As mentioned in Section I, it is desirable to limit the acceleration of the robot. In reality, the actual limits of the robot are not acceleration limits but rather torque limits. While it is theoretically possible to incorporate the complete dynamic model, obtaining an accurate dynamic model becomes challenging, notably in the presence of friction. Moreover, doing so significantly escalates the computational complexity of the MPC problem.

Therefore, we choose to focus on accelerations, considering it a suitable proxy that simplifies the problem by disregarding the complexities associated with modelling the robot's torques and dynamics. This decision is also motivated by the fact that many robot manipulators are equipped with high-performance internal motion controllers, which, to a large extent, linearize and decouple the robot's dynamics.

Consequently, the system dynamics $\dot{\mathbf{x}} = f(\mathbf{x}(t), \mathbf{u}(t))$ are modelled as

$$ \frac{\mathrm{d}}{\mathrm{d}t}\begin{bmatrix}\mathbf{q}\\ \dot{\mathbf{q}}\end{bmatrix} = \begin{bmatrix}0 & 1\\ 0 & 0\end{bmatrix}\begin{bmatrix}\mathbf{q}\\ \dot{\mathbf{q}}\end{bmatrix} + \begin{bmatrix}0\\ 1\end{bmatrix}\ddot{\mathbf{q}}_{\text{ref}}, \tag{7} $$

where the state $\mathbf{x}$ consists of the joint positions $\mathbf{q}$ and velocities $\dot{\mathbf{q}}$. The reference acceleration $\ddot{\mathbf{q}}_{\text{ref}}$ serves as the control input $\mathbf{u}$. Furthermore, the states and control input are bounded by

$$ \mathbf{q} \in [\mathbf{q}^-, \mathbf{q}^+], \quad \dot{\mathbf{q}} \in [\dot{\mathbf{q}}^-, \dot{\mathbf{q}}^+], \quad \ddot{\mathbf{q}}_{\text{ref}} \in [\ddot{\mathbf{q}}^-, \ddot{\mathbf{q}}^+], \tag{8} $$

where $\square^-$ and $\square^+$ denote the lower and upper limits.

B. Task specification

As mentioned in Section II-B and depicted in Fig. 2, a *task frame* is defined with a fixed pose relative to the robot's end-effector (ee) frame. The general idea is that the origin of this task frame should be constrained to the surface, while the z-axis of this frame remains perpendicular to the surface. Furthermore, the aim is to move the task frame along the surface frame's y-axis while maintaining a fixed x-position.

To keep the laser sensor in front of the task frame, the y-axis of the task frame is further constrained to remain perpendicular to the x-axis of the surface frame. This is because the task frame is moving along the y-axis of the surface, and the laser sensor is installed along the (positive) y-axis of the task frame.

To aid in expressing the task constraints, the following shorthand notation is used:

$$g_i(\mathbf{x}) \xrightarrow{w_i} g_{\text{ref},i}, \tag{9}$$

where g_i is an output function with a corresponding reference value $g_{\text{ref},i}$. As will be discussed in Section III-C, each constraint has a corresponding weight w_i which describes the priority for conflicting tasks.

The transformation between the surface frame and the task frame can be computed using the kinematics of the robot:

$$\prescript{\text{task}}{\text{surface}}{\mathbf{T}}(\mathbf{q}) = \prescript{\text{surface}}{\text{world}}{\mathbf{T}}^{-1} \prescript{\text{ee}}{\text{world}}{\mathbf{T}}(\mathbf{q}) \prescript{\text{task}}{\text{ee}}{\mathbf{T}} \tag{10}$$

This transformation consists of an origin $\prescript{\text{task}}{\text{surface}}{\mathbf{p}}(\mathbf{q})$ and a rotation $\prescript{\text{task}}{\text{surface}}{\mathbf{R}}(\mathbf{q})$, denoted by

$$\prescript{\text{task}}{\text{surface}}{\mathbf{p}}(\mathbf{q}) = \begin{bmatrix} p_x^{\text{task}} & p_y^{\text{task}} & p_z^{\text{task}} \end{bmatrix}^{\mathsf{T}}, \tag{11}$$

$$\prescript{\text{task}}{\text{surface}}{\mathbf{R}}(\mathbf{q}) = \begin{bmatrix} \mathbf{r}_x & \mathbf{r}_y & \mathbf{r}_z \end{bmatrix}. \tag{12}$$

Here, $\prescript{\text{task}}{\text{surface}}{\mathbf{R}}(\mathbf{q})$ is a rotation matrix consisting of three unit vectors $\mathbf{r}_x$, $\mathbf{r}_y$, and $\mathbf{r}_z$, representing the x, y, and z axes.

The first task constraint ensures that the origin of the task frame lies on the surface:

$$\mathrm{s}_{\boldsymbol{\theta}}(p_x^{\text{task}}, p_y^{\text{task}}) \xrightarrow{w_1} p_z^{\text{task}}. \tag{13}$$

A second task constraint keeps the z-component of the task frame aligned with the surface normal $\mathbf{n}$:

$$\frac{\mathbf{n}^{\mathsf{T}}}{||\,\mathbf{n}\,||}\mathbf{r}_z \xrightarrow{w_2} 1, \tag{14}$$

where $\mathbf{n}$ is defined as

$$\mathbf{n} = \begin{bmatrix} \frac{\partial\, \mathrm{s}_{\boldsymbol{\theta}}(p_x,p_y)}{\partial p_x} \\ \frac{\partial\, \mathrm{s}_{\boldsymbol{\theta}}(p_x,p_y)}{\partial p_y} \\ -1 \end{bmatrix}. \tag{15}$$

A third task constraint ensures the y-axis of the task frame remains perpendicular to the x-axis of the surface frame:

$$\begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \mathbf{r}_y \xrightarrow{w_3} 0. \tag{16}$$

A fourth task constraint keeps the origin of the task frame at a fixed x-position of the surface frame:

$$p_x^{\text{task}} \xrightarrow{w_4} p_{x,\text{ref}}, \tag{17}$$

where $p_{x,\text{ref}}$ is the desired x-position.

Finally, a fifth task constraint guides the task frame along the y-direction of the surface frame:

$$\prescript{\text{task}}{}{v}_y^{\text{task}} \xrightarrow{w_5} v_{y,\text{ref}}, \tag{18}$$

where $\prescript{\text{task}}{}{v}_y^{\text{task}}$ represents the y-component of the task frame's velocity in the task frame itself, and $v_{y,\text{ref}}$ is the desired velocity. This velocity is tangential to the surface.

C. Model Predictive Control

Once the task constraints are specified, a feedback controller is implemented to realise the desired behaviour. To achieve high performance with acceleration limits, this paper utilises MPC. With MPC, an Optimal Control Problem (OCP) is repeatedly solved for a horizon T into the future, with the following structure

$$\min_{\mathbf{u}(t)} \int_{t_k}^{t_k+T} l(\mathbf{x}(t), \mathbf{u}(t))\ \mathrm{d}t \tag{19a}$$

$$\text{s.t.} \quad \dot{\mathbf{x}}(t) = f(\mathbf{x}(t), \mathbf{u}(t)), \tag{19b}$$

$$\mathbf{x}(t_k) = \mathbf{x}_k, \tag{19c}$$

$$\mathbf{x}(t) \in \mathcal{X},\ \mathbf{u}(t) \in \mathcal{U}. \tag{19d}$$

At every timestep t_k, the OCP is solved given the latest state measurement $\mathbf{x}_k$, and the first control input $\mathbf{u}(t_k)$ is applied.

Equation (19b) constrains the OCP to satisfy the system dynamics, previously defined in (7), whereas (19c) ensures that the system's state starts at the latest measured state $\mathbf{x}_k$. In (19d), the states and inputs are bounded to the sets $\mathcal{X}$ and $\mathcal{U}$, respectively. These sets are used to impose the state and input constraints expressed in (8).

The objective function (19a) describes the desired behaviour of the controller and has the following structure:

$$l(\mathbf{x}(t), \mathbf{u}(t)) = \mathbf{e}(\mathbf{x}(t))^{\intercal}\, \mathbf{W}\, \mathbf{e}(\mathbf{x}(t)) + \lambda\, \mathbf{u}(t)^{\intercal}\, \mathbf{u}(t), \tag{20}$$

where $\mathbf{e}(\mathbf{x}) = \mathbf{g}(\mathbf{x}) - \mathbf{g}_{\text{ref}}$ is the task error.

The task errors are weighted in the objective function with $\mathbf{W} = \text{diag}(w_1, w_2, \ldots)$, a diagonal matrix containing the weights associated with each task. Furthermore, the control inputs are regularised with a weight λ.

D. Implementation

The task constraints are modelled using CasADi [17] and Pinocchio [18] and the OCP is specified within Rockit [19], a software framework to quickly prototype OCPs. The OCP is transcribed to a nonlinear program using a multiple shooting approach and is solved online using the FAst TRajectory OPtimizer (FATROP) [20], an efficient OCP solver. Further details are available in the accompanying git repository[1].

IV. VALIDATION

Simulations were performed using ten randomly generated surfaces, illustrated in Fig. 4. These surfaces were generated by fitting a 2D B-spline over a grid of randomised points. The robot was simulated using (7), while the laser sensor was simulated by projecting laser beams, and computing their intersection with the B-spline surface.

The controller was tested with three surface models: a quadratic model, an RBFN with five basis functions, and the ground truth model (to assess controller performance without estimation error). The parameters of the first two surface models were estimated online from simulated laser measurements, as detailed in Section II.

[1] https://gitlab.kuleuven.be/rob/projects/etasl_mpc/surface-following/surface_following_simulation

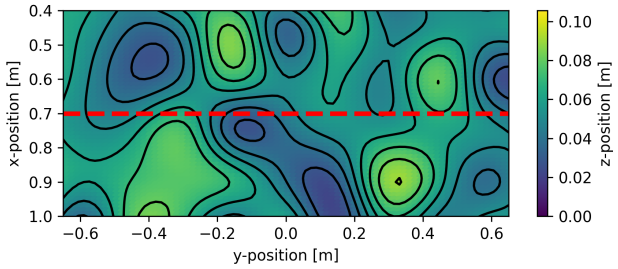


Fig. 4: **One of the ten ground truth surfaces.** Ten distinct surfaces were randomly generated for simulation. The red dashed line shows the desired tool path over the surface.

Each surface model was tested for 9 different prediction horizon lengths. Consequently, we conducted a total of 270 simulations (10 distinct surfaces x 3 surface models x 9 horizon lengths). Notably, during all simulations the MPC converged to a feasible solution, indicating that all the state constraints and input constraints (the acceleration limits) were satisfied for all the experiments.

The parameters used during the simulations are summarised in Table I. The control and estimation were performed at 100Hz on a laptop PC with an Intel® Core™ i7-10810U CPU and 32 GB of RAM.

A. Performance analysis

To analyse the approach's performance, the resulting surface following trajectories were assessed using the objective function in (20), however with a significant difference: for all three the surface models, the objective function was measured against the ground truth surface model and not the estimated surface model. This method ensures a more direct comparison with the true objective. Furthermore, the objective function was integrated over the ten-second duration of each trajectory.

Fig. 5 presents the results for the three surface models across the nine different horizon lengths. Firstly, using the RBFN resulted in a significantly lower objective than when using the quadratic model. Secondly, the objective decreases as the control horizon increases for all three surface models. However, the rate of improvement diminishes with increasing horizon length.

B. Computation time analysis

To maintain practical relevance, we investigate the computation times and target the same control sample time used by Gold et al. [13], which was 0.01 seconds. Fig. 6 presents the computation time of the MPC for the three different surface models across the nine different horizon lengths. Two key observations can be made. Firstly, except for a few outliers with the quadratic model, all the computation times remained far below the target of 0.01 seconds for all horizon lengths, confirming the real-time feasibility of the approach. The exact cause of the outliers with the quadratic model is not entirely clear, and will be further investigated in the future. The second key observation from Fig. 6 is that the computation time increased as the horizon length increased. Therefore, in

TABLE I: Experiment parameters

Parameter	Eq. #	Value
Controller		
$[\mathbf{q}^-, \mathbf{q}^+]$	8	based on UR10 robot
$[\dot{\mathbf{q}}^-, \dot{\mathbf{q}}^+]$	8	[-1,1] rad.s^{-1}
$[\ddot{\mathbf{q}}^-, \ddot{\mathbf{q}}^+]$	8	[-1,1] rad.s^{-2}
w_1	13	25
w_2	14	5
w_3	16	5
w_4	17	25
w_5	18	1
λ	20	1×10^{-9}
$p_{x,\text{ref}}$	17	0.7 m
$v_{y,\text{ref}}$	18	0.25 m.s^{-1}
Estimator		
N	3	5
L	6	100
μ^{Q}	6	1×10^{-5}
μ^{RBFN}	6	1×10^{-1}

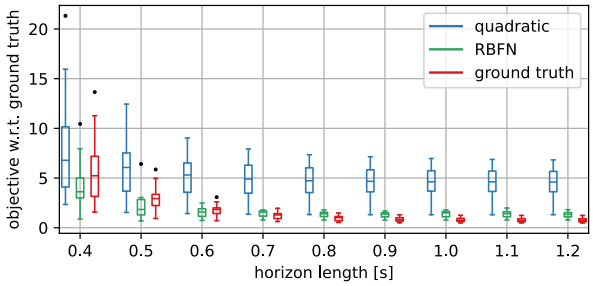


Fig. 5: **Performance of controller with different surface models.** The MPC controller's performance was assessed in combination with three different surface models, each with various horizon lengths. Evaluation involved measuring the objective function against the ground truth surface at each time step, with results integrated over a ten-second interval.

combination with Fig. 5, it shows that there exists a trade-off between performance and computational demand.

With regards to the estimator, the computation time during all the simulations was less than 0.005 seconds, confirming its real-time feasibility.

V. CONCLUSION AND FUTURE WORK

In this paper, we introduced an approach for contactless surface following with acceleration limits using MPC. Our method, combined with an RBFN surface model, demonstrates superior performance compared to other techniques, particularly outperforming local quadratic models.

However, even though the RBFN outperformed the quadratic model, the ground truth yielded even better results. This indicates that combining local sensing with a global surface model obtained from a depth camera or CAD could further enhance overall system performance. We consider this to be an area of future research.

Additionally, we observed that performance increased with the horizon length, but beyond a certain point, diminishing

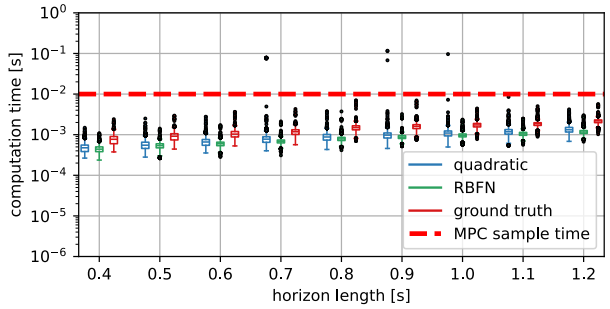


Fig. 6: **Computation time.** The computation time of the MPC for different surface models, with different horizon lengths.

returns set in, accompanied by increased computation times. Therefore, careful selection of the horizon length is imperative. Further research could explore heuristics for choosing a good horizon length and surface model for a given application based on surface characteristics, acceleration limits, and the desired accuracy.

Our next step involves the practical validation of the approach using a real-world setup. We are confident in the feasibility of this transition, as our computation times are within the sample time of the robot. Even though this research focused on the application of contactless surface following, we believe it contributes to the overarching goal of bringing MPC closer to real-life industrial applications in robotics.

REFERENCES

[1] A. Verduyn, J. De Schutter, W. Decré, and M. Vochten, "Shape-based path adaptation and simulation-based velocity optimization of initial tool trajectories for robotic spray painting," *IEEE International Conference on Automation Science and Engineering*, 2023.

[2] D. Nakhaeinia, R. Fareh, P. Payeur, and R. Laganiere, "Trajectory planning for surface following with a manipulator under RGB-D visual guidance," in *IEEE International Symposium on Safety, Security, and Rescue Robotics*, 2013.

[3] Y. Wen, J. Hu, and P. R. Pagilla, "A Novel Robotic System for Finishing of Freeform Surfaces," en, *IEEE International Conference on Robotics and Automation*, 2019.

[4] M. Dyck, A. Sachtler, J. Klodmann, and A. Albu-Schaffer, "Impedance Control on Arbitrary Surfaces for Ultrasound Scanning Using Discrete Differential Geometry," *IEEE Robotics and Automation Letters*, 2022.

[5] I. Huang, D. Chow, and R. Bajcsy, "Soft Tactile Contour Following for Robot-Assisted Wiping and Bathing," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2022.

[6] S. S. M. Salehian and A. Billard, "A Dynamical-System-Based Approach for Controlling Robotic Manipulators During Noncontact/Contact Transitions," *IEEE Robotics and Automation Letters*, 2018.

[7] S. Demey and J. De Schutter, "Enhancing surface following with invariant differential part models," *IEEE International Conference on Robotics and Automation*, 1994.

[8] M. Amersdorfer, J. Kappey, and T. Meurer, "Real-time freeform surface and path tracking for force controlled robotic tooling applications," *Robotics and Computer-Integrated Manufacturing*, 2020.

[9] M. Iskandar, C. Ott, A. Albu-Schäffer, B. Siciliano, and A. Dietrich, "Hybrid force-impedance control for fast end-effector motions," *IEEE Robotics and Automation Letters*, 2023.

[10] W. Decré, H. Bruyninckx, and J. De Schutter, "Extending the itasc constraint-based robot task specification framework to time-independent trajectories and user-configurable task horizons," in *IEEE International Conference on Robotics and Automation*, 2013.

[11] A. Wahrburg and K. Listmann, "Mpc-based admittance control for robotic manipulators," in *IEEE Conference on Decision and Control*, 2016.

[12] J. Matschek, J. Bethge, P. Zometa, and R. Findeisen, "Force feedback and path following using predictive control: Concept and application to a lightweight robot," *IFAC-PapersOnLine*, 2017.

[13] T. Gold, A. Völz, and K. Graichen, "Model predictive interaction control for industrial robots," *IFAC-PapersOnLine*, 2020, 21st IFAC World Congress.

[14] D. Broomhead and D. Lowe, "Radial basis functions, multi-variable functional interpolation and adaptive networks," *Royal signals and radar establishment Malvern (United Kingdom)*, 1988.

[15] A. Bodard, P. Pas, and P. Patrinos, "PANTR: A proximal algorithm with trust-region updates for nonconvex constrained optimization," *IEEE Control Systems Letters*, 2023.

[16] C. Samson, M. Le Borgne, and B. Espiau, *Robot Control, the Task Function Approach* (Combinatorial Scientific Computing). Clarendon Press, 1991.

[17] J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, "CasADi – A software framework for nonlinear optimization and optimal control," *Mathematical Programming Computation*, 2019.

[18] J. Carpentier et al., "The pinocchio c++ library – a fast and flexible implementation of rigid body dynamics algorithms and their analytical derivatives," in *IEEE International Symposium on System Integrations*, 2019.

[19] J. Gillis, B. Vandewal, G. Pipeleers, and J. Swevers, "Effortless modeling of optimal control problems with rockit," *Benelux Meeting on Systems and Control*, 2020.

[20] L. Vanroye, A. Sathya, J. De Schutter, and W. Decré, "Fatrop: A fast constrained optimal control problem solver for robot trajectory optimization and control," *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2023.