# Context-aware robotic arm using fast embedded model predictive control

Shane Trimble*, Wasif Naeem†, Seán McLoone‡ and Pantelis Sopasakis§
*School of EEECS and* I-AMS *Centre Queen's University Belfast*, Belfast, UK
Email: *strimble08@qub.ac.uk, †w.naeem@qub.ac.uk, ‡s.mcloone@qub.ac.uk, §p.sopasakis@qub.ac.uk

***Abstract*—The growing number of collaborative robotics in unstructured environments creates highly nonconvex nonlinear shared dynamical systems. For safety and speed, path planning and collision avoidance are of the utmost importance in these situations. We present a novel nonlinear MPC solution for use on a three-dimensional four-axis robotic manipulator. The system is the first of it's kind to take into account moving obstacles. Using the OpEn framework, optimisation is done by the PANOC and ALM techniques. Experimentation demonstrates extremely fast solver times on both PC and embedded platforms.**

***Index Terms*—Model Predictive Control, Collaborative robotics, Fast embedded optimisation**

## I. Introduction

The field of robotics has seen large growth in the area of collaborative robotics; robots working both with each other and with humans [1], [2]. While traditionally robots are operated in isolation or in harmony with other moving entities of explicit and unambiguous motion routines, this trend can create a highly unstructured shared workspace [3]. Humans can be unpredictable and it can be both complex and costly to integrate robotic systems in close proximity [4]. For safety, speed and reliability it is imperative that collisions are avoided at all cost. Well established techniques of path planning and collision avoidance include artificial potential fields [5] and graph based search methods, such as rapidly-exploring random trees (RRT) [6]. These methods are effective but do not always compute the most efficient paths and incorporating dynamic constraints can be challenging.

Deep learning is becoming popular in the area of robotics, [7]. By learning in a similar fashion to humans, an agent adapts its actions over time with respect to a cost function, to improve performance of a task, both in simulation and on hardware [8]. While providing impressive results [9], it is difficult to observe how the system makes decisions and know how it will respond to new (unseen) events — thus such systems lack safety guarantees in unstructured environments. The agent also requires training, which often can only be performed in simulation — therefore, its reliability depends on the validity of the available model — which can be difficult to update online so that the agent can adapt to new environments.

When dealing with constrained dynamical systems, model predictive control (MPC) is a highly capable control framework [10]. MPC can take into account nonlinear system dynamics, input and state constraints present in most mechanical and robotics systems, as well as obstacle/collision avoidance constraints [11], [12]. Another benefit of MPC is the ability to take into account the behaviour of other environmental sub-systems, such as human motion [13], [14]. This allows for reliable collision avoidance, making a well-designed MPC suitable for collaborative workspaces [15]. Attempts have also been made to implement MPC on embedded systems [16].

In robotics, MPC has been used on applications such as UAVs, swarm formation, walking, humanoid and mobile robots, [11], [12], [17]. MPC for obstacle avoidance for manipulation tasks has been explored in [18], [19].

Trajectory planning with obstacle avoidance can be cast as a nonconvex optimisation problem. Optimal solutions can be found by using interior-point (IP) [20] and sequential quadratic programming (SQP) methods [21]. These techniques are not always appropriate for high-speed or embedded implementations due to computational complexity and limited onboard resources.

The authors of [22] combine the proximal averaged Newton-type method for optimal control (PANOC) [23] with the penalty method to compute collision-free paths using MPC. PANOC has also been used to enable the solution of fast embedded nonlinear MPC problems in real time on embedded systems and is gaining momentum in applications [24]–[27].

Another approach to solving constrained optimisation problems is the augmented Lagrangian method (ALM) [28]. In [29], PANOC is combined with ALM and PM in a high-speed framework for the design of fast, efficient embedded optimisation-based MPC controllers using Optimization Engine (OpEn): an open-source tool for designing high performace MPC controllers. Code is generated in Rust, a high-performance, memory-safe modern programming language. The algorithm is especially suited to embedded applications as it involves only simple algebraic operations and has a low memory footprint.

This paper proposes a high speed solution to the manipulator obstacle avoidance problem. We demonstrate that PANOC can be coupled with the augmented Lagrangian method to enable robotic manipulators to avoid moving obstacles in their environment. In this study obstacles are described in terms of Lagrangian constraints. A smooth cost function allows the system to also take into account soft constraints. With this formulation, the OpEn framework is used to create a fast and efficient MPC solver.

In this paper, we propose an MPC scheme for moving obstacle collision avoidance on a three-dimensional RRRR

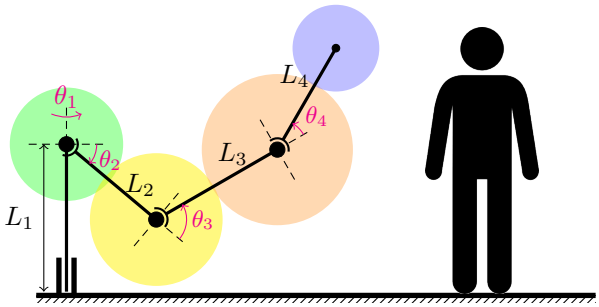

Fig. 1: Three-dimensional four-link robotic arm

elbow type robotic arm. While others have applied MPC to manipulator trajectory planning [30], [31], our solution improves on this as an external system is not required to compute collision-free paths. We also are the first to apply this approach to moving obstacles. This creates a highly nonconvex optimisation problem that needs to be solved in a limited time. We do this using the OpEn framework, combining PANOC with ALM, without the use of parallelisation. OpEn generates Rust code that is provably memory-safe and, therefore, perfectly suitable for embedded applications. We demonstrate our solution experimentally on both PC and embedded platforms, resulting in extremely fast solution times.

## II. MPC FORMULATION

Consider the four-link robotic arm shown in Fig. 1. The ends of its four links with respect to the Earth-fixed frame of reference have coordinates $p_1 = (0, 0, L_1)$, $p_2 = (x_2, y_2, z_2)$, $p_3 = (x_3, y_3, z_3)$ and $p_4 = (x_4, y_4, z_4)$. Define the function

$$g(x, y) = \begin{bmatrix} \sin x \cos y \\ \cos x \cos y \\ \sin y \end{bmatrix}. \tag{1}$$

The forward kinematics of this four-link robotic arm is

$$p_2 = p_1 + L_2 g(\theta_1, \theta_2), \tag{2a}$$
$$p_3 = p_2 + L_3 g(\theta_1, \theta_2 + \theta_3), \tag{2b}$$
$$p_4 = p_3 + L_4 g(\theta_1, \theta_2 + \theta_3 + \theta_4). \tag{2c}$$

The dynamics is described by $\dot{\theta}_i = \omega_i$, for $i = 1, \ldots, 4$, where $\omega_i$ is the angular velocity of motor $i$, which can be directly manipulated for robots equipped with stepper motors Stepper motors obviate the need for more involved dynamics such as the ones reported in [32].

The robotic arm is a system with inputs $u = (\omega_1, \omega_2, \omega_3, \omega_4)$, (measured) states $x = (\theta_1, \theta_2, \theta_3, \theta_4)$, and outputs $z = (p_2, p_3, p_4)$. We may write the system succinctly as

$$\dot{x}(t) = u(t), \tag{3a}$$
$$z(t) = G(x(t)), \tag{3b}$$

where $G : \mathbb{R}^4 \to \mathbb{R}^9$ is a mapping defined by Equation (2). The angular velocities, $\omega_i(t)$, are constrained in the interval $[-\omega_{\max}, \omega_{\max}]$.

The robotic arm needs to avoid collisions with a moving obstacle which is described by a set $O(t) \subseteq \mathbb{R}^3$. In order to guarantee that the robotic arm does not collide with the obstacle, we require that

$$p_i(t) \notin O(t) \oplus \mathcal{B}_{a_i}, \tag{4}$$

for $i \in \mathbb{N}_{[1,4]}$, where $\oplus$ denotes the Minkowski sum of two sets and $\mathcal{B}_{a_i}$ denotes a ball of radius $a_i > 0$ centred at $p_i(t)$, where the radii $a_i$ of each ball are chosen such that the entire robotic arm is covered as shown in Fig. 1. For example if $L_i = L$ for all $i$, we can choose $a_i = L/2$. Equation (4) defines a sufficient condition for collision avoidance between the moving obstacle and the robot. For the sake of brevity we will consider the case of a single moving obstacle; however, our methodology can be readily extended to cases with multiple obstacles.

### A. Obstacle tracking and motion prediction

Suppose that the obstacle is described by a set $O(t) \subseteq \mathbb{R}^3$ given by

$$O(\xi(t)) = \left\{ p \in \mathbb{R}^3 : h_i(p, \xi(t)) > 0, i \in \mathbb{N}_{[1,n_h]} \right\}, \tag{5}$$

where $\xi(t)$ is a vector of obstacle-specific parameters. In other words, an obstacle $O$ is descibed by a set of $n_h$ inequalities which are expressed in terms of (smooth) functions $h_i$. For example, for ball-shaped obstacles we can define $\xi(t)$ as the centre of the ball, $p_c^o(t)$, and its radius, $r^o(t)$, that is, $\xi(t) = (p_c^o(t), r^o(t))$. The prediction horizon length is denoted by $n_h$. Then, we can define $h(t; \xi(t)) = r^o(t)^2 - \|p - p_c(t)\|^2$.

The motion of the obstacle can be fully identified by $\xi(t)$. Modern vision-based solutions can be used to track the position, orientation and shape of moving obstacles at a very high sampling rate [33]. For the purpose of designing an MPC controller, it is expedient to be able to predict the future trajectories of $\xi(t)$ using past observations. While advanced methods of trajectory prediction make use of methods such as deep learning [34], and state estimators [35], one of the simplest modelling approaches is to assume that the object will retain a constant velocity and shape [36]. Following this approach, the position of the obstacle at a future time $t + t'$, given its velocity at time $t$, is estimated by $p(t) + v(t)t'$, where $v(t)$ is the obstacle's velocity at time $t$.

In general, at every time $t$, MPC requires a predictor, $\hat{\xi}(\cdot)$, that will predict the obstacle's position, orientation and shape at the future time instant $t'$, $\hat{\xi}(t')$, using observations that are available up to time $t$.

### B. Model predictive control

The robot's objective is for its end effector to reach a certain position and orientation and, in some cases, a certain (angular) velocity, [37].

At every sampling time instant, $t = kT_s$, we need to solve the following optimal control problem with prediction horizon $T_p > 0$, stage cost function $\ell$ and terminal cost function $\ell_f$

$$\mathbb{P}(x, \hat{\xi}(\,\cdot\,)) : \underset{\hat{u}:[0,T_p]\to U}{\text{Minimise}} \ \ell_f(\hat{z}(T_p), \hat{x}(T_p)) + \int_0^{T_p} \ell(\hat{z}(\tau), \hat{x}(\tau), \hat{u}(\tau)) \mathrm{d}\tau \tag{6a}$$

subject to the constraints

$$\dot{\hat{x}}(t) = \hat{u}(t), t \in [0, T_p], \hat{x}(0) = x, \tag{6b}$$

$$\hat{z}(t) = G(\hat{x}(t)), t \in [0, T_p], \tag{6c}$$

$$\hat{p}_\iota(t) \notin O(\hat{\xi}_t(t)) \oplus \mathcal{B}_{a_\iota}, \iota \in \mathbb{N}_{[1,4]}, t \in [0, T_p]. \tag{6d}$$

The control actions are constrained in set $U = \{u \in \mathbb{R}^4 : -\omega_{\max} \leq u_i \leq \omega_{\max}\}$, which is a $\|\cdot\|_\infty$-ball of radius $\omega_{\max}$. The stage cost function, $\ell$, and the terminal cost function, $\ell_f$, encode the task objectives. For example, for the end effector to reach a prescribed position, we may choose a stage cost function of the form

$$\ell(z, x, u) = q_p \|p_4 - p_4^{\text{ref}}\|^2 + q_w \left\| {}^1\!/_{L_4}(p_4 - p_3) - w^{\text{ref}} \right\|^2 + q_u \|u\|^2, \tag{7}$$

where $q_p, q_w, q_u \geq 0$ are weight parameters. The first term in the above cost function penalises the distance of the end effector from the desired position. The second term penalises the deviation of the orientation of the fourth link of the robot from the desired orientation which is prescribed by a unitary vector $w^{\text{ref}}$. The last term is used to penalise large angular velocities. An additional term of the form $\|\dot{u}\|$ can be included to in order to deliver smoother trajectories by penalising abrupt changes in the control actions.

The terminal cost function, $\ell_f$, can be taken to be a quadratic penalty function involving $p_4$ and $p_4 - p_3$, akin to the state cost function in Equation (7).

The above continuous-time optimal control problem can be discretised by assuming that control actions are delivered via a zero-order hold element with sampling time $T_s$ using an explicit discretisation and integration method, such as the Euler method or RK4 [25].

Assuming that $T_p = NT_s$, at every time instant $t = kT_s$ we solve the discretised optimisation problem, which is parametric in the current state of the system, $x$, and the sequence of predicted obstacle parameters, $\bar{\xi} = (\hat{\xi}_1, \ldots, \hat{\xi}_N)$,

$$\mathbb{P}_d(x, \bar{\xi}): \underset{\substack{\hat{u}_0, \hat{u}_1, \ldots, \\ \hat{u}_{N-1} \in U}}{\text{Min.}} \ \ell_{d,f}(\hat{z}_N, \hat{x}_N) + \sum_{k=0}^{N-1} \ell_d(\hat{z}_k, \hat{x}_k, \hat{u}_k), \tag{8a}$$

subject to the constraints

$$\hat{x}_{k+1} = \hat{x}_k + T_s \hat{u}_k, k \in \mathbb{N}_{[0,N-1]}, \hat{x}_0 = x, \tag{8b}$$

$$\hat{z}_k = G(\hat{x}_k), k \in \mathbb{N}_{[0,N-1]}, \tag{8c}$$

$$\hat{p}_{\iota,k} \notin O(\hat{\xi}_k) \oplus \mathcal{B}_{a_\iota}, \iota \in \mathbb{N}_{[1,4]}, k \in \mathbb{N}_{[1,N]}. \tag{8d}$$

At every discrete time instant we solve $\mathbb{P}_d$ and obtain an optimal sequence of control actions, $u_j^\star(x; \bar{\xi})$, $j \in \mathbb{N}_{[0,N-1]}$, which depends on the current state, $x$, and the predicted obstacle parameters, $\bar{\xi}$. The first element of this sequence, $u_0^\star(x; \bar{\xi})$, is applied to the system and $\mathbb{P}_d$ is solved again at the subsequent time instant in a receding horizon fashion.

## III. Solution approach

### A. Problem formulation

We follow the *single shooting formulation* where the cost function is modified so as to eliminate the sequence of states, $x_k$, and reference variables, $z_k$, so that the decision variable of Problem $\mathbb{P}_d$ in Equation (8) is the vector of control actions, $\bar{u} = (\hat{u}_0, \hat{u}_1, \ldots, \hat{u}_{N-1})$. To that end, we define $\bar{F}_0(\bar{u}, x) = x$ and $\bar{F}_{k+1}(\bar{u}, x) = F(\bar{F}_k(\bar{u}, x), u_k)$ for $k \in \mathbb{N}_{[0,N-1]}$ and the cost function becomes

$$f(\bar{u}; x) = \ell_{d,f}(G(\bar{F}_N(\bar{u}, x)), \bar{F}_N(\bar{u}, x)) + \sum_{k=0}^{N-1} \ell_d(G(\bar{F}_k(\bar{u}, x)), \bar{F}_k(\bar{u}, x), \hat{u}_k). \tag{9}$$

Now $\mathbb{P}_d$ can be written as

$$\mathbb{P}_d(x, \bar{\xi}) : \underset{\bar{u} \in U^N}{\text{Minimise}} \ f(\bar{u}; x) \tag{10a}$$

$$\text{subject to: } \hat{p}_{\iota,k} \notin O(\hat{\xi}_k) \oplus \mathcal{B}_{a_\iota}, \quad \iota \in \mathbb{N}_{[1,4]}, k \in \mathbb{N}_{[1,N]}. \tag{10b}$$

Let $\widetilde{O}(\xi, a)$ denote the enlarged obstacle $O(\xi) \oplus \mathcal{B}_a$ in (8d) and suppose that

$$\widetilde{O}(\xi, a) = \left\{ p \in \mathbb{R}^3 : \tilde{h}_i(p; \xi, a) > 0, \text{ for } i \in \mathbb{N}_{[1,n_h]} \right\} \tag{11}$$

In the special case where $n_h = 1$ — for example, when the obstacle is a Euclidean ball, or an ellipsoid — the obstacle avoidance constraints in (8d) for a point $p \in \mathbb{R}^3$ boils down to $\tilde{h}_1(p_\iota; \hat{\xi}_k, a_\iota) \leq 0$. In this case, we may define the mapping $\Phi : \mathbb{R}^{4N} \to \mathbb{R}^{4N}$ which is given by

$$\Phi(\bar{u}; x, \bar{\xi}) = \left[ \tilde{h}_1(p_\iota(\bar{u}; x); \hat{\xi}_k, \alpha_\iota) \right]_{k,\iota}. \tag{12}$$

Then, $\mathbb{P}_d$ is the following constrained optimisation problem

$$\mathbb{P}_d(x, \bar{\xi}) : \underset{\bar{u} \in U^N}{\text{Minimise}} \ f(\bar{u}; x) \tag{13a}$$

$$\text{subject to: } \Phi(\bar{u}; x, \bar{\xi}) \leq 0. \tag{13b}$$

Often — for example, when the obstacle is a Euclidean ball or an ellipsoid — function $\Phi$ is smooth in $\bar{u}$. This allows us to use the augmented Lagrangian method (see Section III-B).

In the more general case when the obstacle is described by a set of $n_h > 1$ inequalities, the obstacle avoidance constraints in (8d) are satisfied if and only if there is an $i_0 \in \mathbb{N}_{[1,n_h]}$ such that $\tilde{h}_i(p; \hat{\xi}_k, a_\iota) \leq 0$. This condition holds if and only if the following equality condition is satisfied

$$\psi_\iota(\bar{u}, x; \hat{\xi}_k) := \prod_{i=1}^{n_h} \left[ \tilde{h}_i(p_\iota(\bar{u}, x); \hat{\xi}_k, a_\iota) \right]_+ = 0. \tag{14}$$

Let us define $\Psi(\bar{u}, x; \bar{\xi}) = [\psi_\iota(\bar{u}, x; \hat{\xi}_k)]_{\iota,k}$. Then, $\mathbb{P}_d$ can be written as follow

$$\mathbb{P}_d(x, \bar{\xi}) : \underset{\bar{u} \in U^N}{\text{Minimise}} \ f(\bar{u}; x) \tag{15a}$$

$$\text{subject to: } \Psi(\bar{u}; x, \bar{\xi}) = 0. \tag{15b}$$

This problem can be solved using the quadratic penalty method as discussed in [38].

### B. Augmented Lagrangian method combined with PANOC

We will focus on Problem $\mathbb{P}_d$ in the form given in (13). We will combine the augmented Lagrangian method with PANOC leading to a fast numerical algorithm which is suitable for embedded real-time implementations and applications. To that end, we introduce the augmented Lagrangian function

$$L_c(\bar{u}, v, y; x, \bar{\xi}) \coloneqq f(\bar{u}; x) + y^\top(\Phi(\bar{u}; x, \bar{\xi}) - v) + \tfrac{c}{2}\|\Phi(\bar{u}; x, \bar{\xi}) - v\|^2, \quad (16)$$

for some $c > 0$, which is defined for $\bar{u} \in U^N$, $v \in \mathbb{R}^{4N}_{\geq 0}$, and $y \in \mathbb{R}^{4N}$. It can be verified that [29]

$$\min_{\substack{\bar{u}\in U^N \\ v\in\mathbb{R}^{4N}_{\geq 0}}} L_c(\bar{u}, v, y; x, \bar{\xi}) = -\tfrac{1}{2c}\|y\|^2 + \min_{\bar{u}\in U^N} \tilde{L}_c(\bar{u}; v, y, x, \bar{\xi}),$$

where $\tilde{L}_c(\bar{u}; y, x, \bar{\xi}) = f(\bar{u}; x) + \frac{c}{2}\|\Phi(\bar{u}; x, \bar{\xi}) + \frac{1}{c}y - \left[\Phi(\bar{u}; x, \bar{\xi}) + \frac{1}{c}y\right]_+\|^2$. Function $\tilde{L}_c$ is differentiable in $\bar{u}$ with gradient

$$\nabla_{\bar{u}}\tilde{L}_c(\bar{u}; v, y, x, \bar{\xi}) = \nabla_{\bar{u}} f(\bar{u}; x) + cJ\Phi(\bar{u}; x, \bar{\xi})^\top\left(\Phi(\bar{u}; x, \bar{\xi}) + \tfrac{1}{c}y - \left[\Phi(\bar{u}; x, \bar{\xi}) + \tfrac{1}{c}y\right]_+\right).$$

This can be computed symbolically using automatic differentiation software; e.g., CasADi [39]. We can formulate a numerical algorithm (see Alg. 1) based on the augmented Lagrangian method [28] where the inner problem has the form

$$\mathbb{P}_{\text{in}}(c, y, x, \bar{\xi}) : \underset{\bar{u}\in U^N}{\text{Minimise}}\ \tilde{L}_c(\bar{u}; v, y, x, \bar{\xi}). \quad (17)$$

**Algorithm 1** Augmented Lagrangian method for embedded nonlinear MPC

**Input:** $\bar{u}^0 \in \mathbb{R}^{4N}$ (initial guess), $x \in \mathbb{R}^{n_p}$ (current state), $\bar{\xi}$ (predicted obstacle parameters), $y^0 \in \mathbb{R}^{4N}$ (initial guess for the Lagrange multipliers), $\epsilon, \delta > 0$ (tolerances), $\beta$ (tolerance decrease coefficient), $\rho$ (penalty update coefficient), $\theta$ (sufficient decrease coefficient), $M > 0$

**Output:** $\delta$-infeasible, $\epsilon$-approximate stationary point $(\bar{u}^\star, y^\star)$

1: $\bar{\epsilon}_0 = \epsilon_0$
2: **for** $\nu = 0, \ldots, \nu_{\max}$ **do**
3: $\quad \bar{y}^\nu = \text{proj}_{[-M,M]^{4N}}(y^\nu)$
4: $\quad \bar{u}^{\nu+1}$ is a solution of $\mathbb{P}_{\text{in}}(c_\nu, \bar{y}^\nu)$ with tolerance $\bar{\epsilon}$ and initial guess $u^\nu$ (obtained using PANOC)
5: $\quad y^{\nu+1} = \bar{y}^\nu + c_\nu(\Phi(\bar{u}^{\nu+1}; x, \bar{\xi}) - [\Phi(\bar{u}^{\nu+1}; x, \bar{\xi}) + c_\nu^{-1}\bar{y}^\nu]_+)$
6: $\quad z_{\nu+1} = \|y^{\nu+1} - \bar{y}^\nu\|_\infty$
7: $\quad$ **if** $z_{\nu+1} \leq c_\nu\delta$ **and** $\bar{\epsilon}_\nu \leq \epsilon$ **then**
8: $\quad\quad$ **return** $(\bar{u}^\star, y^\star) = (\bar{u}^{\nu+1}, y^{\nu+1})$
9: $\quad$ **else if** $\nu > 0$, $z_{\nu+1} > \theta z_\nu$ **then**
10: $\quad\quad c_{\nu+1} = \rho c_\nu$
11: $\quad \bar{\epsilon}_{\nu+1} = \beta\bar{\epsilon}_\nu$

The most computationally demanding step of Alg. 1 is the solution of the inner problem, $\mathbb{P}_{\text{in}}$, in line 4. In the common case where $f$ is continuously differentiable in $\bar{u}$ with Lipschitz-continuous gradient and $\Phi$ is differentiable in $\bar{u}$ with Lipschitz-continuous Jacobian, $\tilde{L}_c$ is continuously differentiable in $\bar{u}$ with Lipschitz-continuous gradient. It is also easy to project on the set of constraints of $\mathbb{P}_{\text{in}}$, $U^N$. It is, therefore, possible to solve $\mathbb{P}_{\text{in}}$ with PANOC.

PANOC aims at determining a zero of the *fixed-point residual* (FPR) operator

$$R_\gamma(\bar{u}; c, y, x, \bar{\xi}) = {}^1\!/\!_\gamma\left(\bar{u} - \text{proj}_{U^N}(\bar{u} - \gamma\nabla_{\bar{u}}\tilde{L}_c(\bar{u}, y, x, \bar{\xi}))\right),$$

which defines a first-order optimality condition for $\mathbb{P}_{\text{in}}$. PANOC combines fast quasi-Newtonian updates with safe projected gradient type steps and uses a line search on the forward backward envelope — an exact, continuous, real-valued merit function — to guarantee global convergence. The algorithm terminates successfully once $\|R_\gamma(\bar{u}; c, y, x, \bar{\xi})\|_\infty \leq \epsilon$, for a specified tolerance $\epsilon > 0$. The algorithm is initialised with an initial guess, $\bar{u}^0$, for the sequence of control actions, and an initial guess, $y^0$, for the vector of Lagrange multipliers.

## IV. Experimental verification

In order to validate the proposed MPC solution, the controller algorithm was implemented on a modified benchtop six-axis manipulator system by ST Robotics: an ***RRRR Elbow*** type manipulator robot. The stepper motor drivers were connected to an Arduino, which was in turn connected to a PC housing the controller. The MPC solver was also compiled for and tested on a Raspberry Pi 2B.

The system model contained in the MPC controller undergoes discretization using the Euler method. The controller inputs are limited to $\pm 0.5\,{}^{\text{rad}}\!/\!_{\text{s}}$ for each joint. The end effector reference position is set to $[1.05, 0, 0.35]$ and the reference orientation is described by the unit vector $[0.9987, 0, -0.05175]$. Weights of $q_p = 20$, $q_\omega = 10$, $q_w = 1$ and $q_p = 0.1$ are applied to $\ell$ and $\ell_f$. A virtual obstacle, a ball of radius $r = 0.3\,\text{m}$, is added to the environment and a straight line trajectory is created to intersect the end effector pathway. The obstacle velocity with relation to the Earth-fixed frame of reference is $[-4, 4, 0]{}^{\text{m}}\!/\!_{\text{s}}$. The sampling time is set to $50\,\text{ms}$, creating a $20\,\text{Hz}$ frequency for the control system. A solution tolerances $\epsilon = 10^{-4}$ and $\delta = 10^{-3}$ are set for the fixed-point residual and the infeasibility respectively. A prediction horizon window of $N = 20$ is chosen. The system is operated in real time on the robot using the i7 PC and using a simulated dynamics model on the Raspberry Pi 2 with identical parameters. Performance metrics are recorded from the controller, encoders and dynamic model output and presented in Fig. 2.

The motion of the robot and obstacle at three time instants is shown in Fig. 3. The target point is represented by a red cross. The predicted pathways of the ball and end effector are denoted by green and red arrows respectively, for horizon $N$.

Fig. 2 reveals that the PC maximum runtime is approximately ten times faster than the embedded system ($3.6\,\text{ms}$ vs $39\,\text{ms}$, respectively), although both consistently converge and stay within the $50\,\text{ms}$ sample time window. The infeasibility plot in Fig. 2(b) shows that the safety radius of the object is penetrated within the configured tolerance. The fixed-point

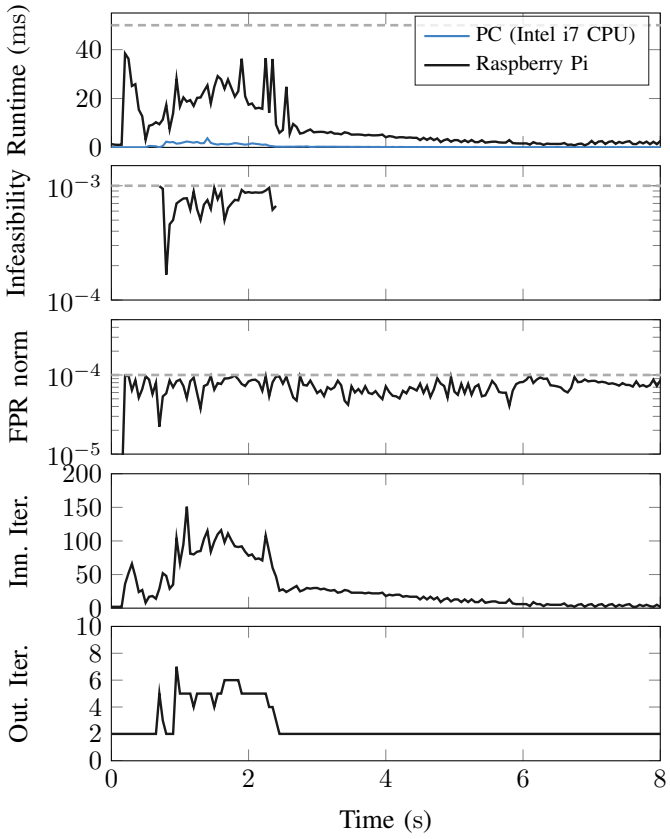


Fig. 2: (First) Solver times on a PC and a Raspberry Pi 2, Model B. The dashed grey line is the sampling time (50 ms). (Second) Infeasibility ($\|[\Phi(\bar{u}; x, \bar{\xi})]_+\|_\infty$), (Third) Norm of the fixed point residual, $\|R_\gamma(\bar{u})\|_\infty$, (Fourth) inner iterations performed by PANOC, (Fifth) outer (ALM) iterations.

residual plot (Fig. 2(c)) shows that the problem converges up to the specificed tolerance for each time sample. The numbers of inner and outer iterations reflect the complexity of convergence at each time instance. It can be seen that the number of each is substantially higher when the ball is near the manipulator.

The distances from the end effector, joint 4 and joint 3 to the obstacle surface for each time-sample are presented in Fig. 4. From this is can be seen that the end effector comes very close, but does not touch the obstacle. The obstacle also does not touch the other moving joints. Controller inputs and states are presented in Fig. 5 and 6 respectively.

## V. Conclusions and Future Work

This paper proposes a three-dimensional collision avoidance and trajectory planning nonlinear MPC solution for a RRRR elbow type manipulator, based on the OpEn framework. Collision avoidance is applied to obstacles of arbitrary shape, with position prediction incorporated into the optimisation scheme. Our solution employs the PANOC and ALM optimisation techniques, making implementation on low resource embedded hardware highly feasible. Experimentation is performed on a real robotic arm. Results show that solver time is as low as

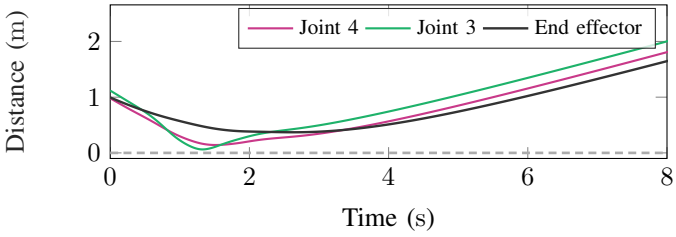


Fig. 4: Distance from end effector to the surface of the object against time.

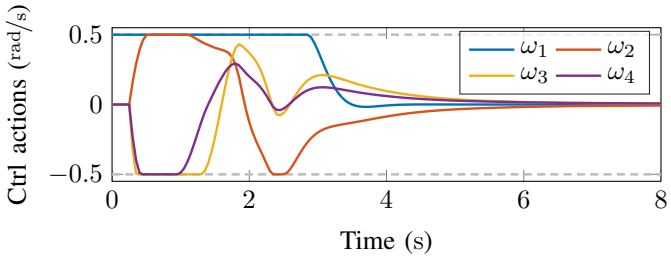


Fig. 5: Control actions computed by the MPC controller ($\omega_i$, $i = 1, \ldots, 4$). Note that the control actions satisfy the prescribed bounds (dashed lines).

3.5 ms, which is extremely fast for a nonlinear, nonconvex system. On both the PC and embedded platform constraint violation is consistently within set limits, allowing for low scan cycles. Future work will incorporate obstacle motion of higher complexity.

## References


[1] W. Wang, R. Li, Y. Chen, Z. M. Diekel, and Y. Jia, "Facilitating humanrobot collaborative tasks by teaching-learning-collaboration from human demonstrations," *IEEE Trans Aut Sci Eng*, vol. 16, pp. 640–653, April 2019.

[2] H. Oliff, Y. Liu, M. Kumar, and M. Williams, "Improving human-robot interaction utilizing learning and intelligence: A human factors-based approach," *IEEE Trans Aut Sci Eng*, pp. 1–14, 2020.

[3] G. Raiola, C. A. Cardenas, T. S. Tadele, T. de Vries, and S. Stramigioli, "Development of a safety- and energy-aware impedance controller for collaborative robots," *IEEE Rob Aut Lett*, vol. 3, pp. 1237–1244, April 2018.

[4] S. A. Green, M. Billinghurst, X. Chen, and J. G. Chase, "Human-robot collaboration: A literature review and augmented reality approach in design," *Int J Adv Rob Sys*, vol. 5, no. 1, p. 1, 2008.

[5] S. Byrne, W. Naeem, and S. Ferguson, "Improved APF strategies for dual-arm local motion planning," *Trans IMC*, vol. 37, no. 1, pp. 73–90, 2015.

[6] J. J. Kuffner and S. M. LaValle, "RRT-connect: An efficient approach to single-query path planning," in *ICRA*, vol. 2, pp. 995–1001, IEEE, 2000.

[7] J. Xie, Z. Shao, Y. Li, Y. Guan, and J. Tan, "Deep reinforcement learning with optimized reward functions for robotic trajectory planning," *IEEE Access*, vol. 7, pp. 105669–105679, 2019.

[8] B. Sangiovanni, A. Rendiniello, G. P. Incremona, A. Ferrara, and M. Piastra, "Deep reinforcement learning for collision avoidance of robotic manipulators," in *ECC*, pp. 2063–2068, June 2018.

[9] T. G. Thuruthel, E. Falotico, F. Renda, and C. Laschi, "Model-based reinforcement learning for closed-loop dynamic control of soft robotic manipulators," *IEEE Trans Rob*, vol. 35, pp. 124–134, Feb 2019.

[10] J. V. Frasch, A. Gray, M. Zanon, H. J. Ferreau, S. Sager, F. Borrelli, and M. Diehl, "An auto-generated nonlinear MPC algorithm for real-time obstacle avoidance of ground vehicles," in *ECC*, pp. 4136–4141, IEEE, 2013.

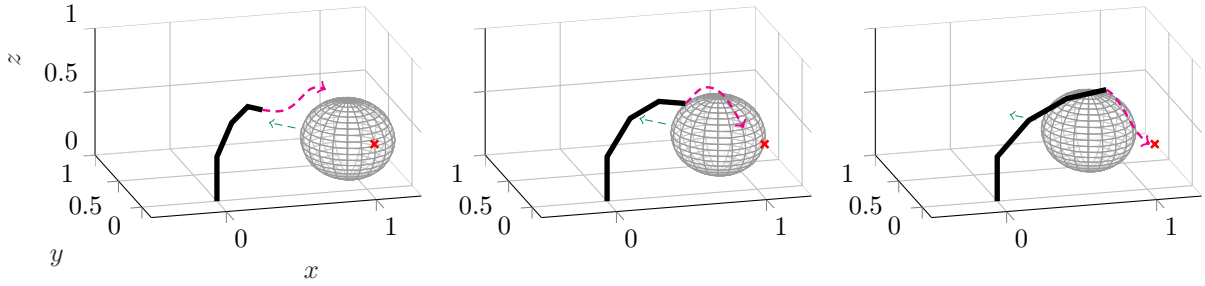


Fig. 3: Snapshots of experimental manipulator obstacle avoidance. The predicted paths of the end effector and ball are shown in red and green respectively.

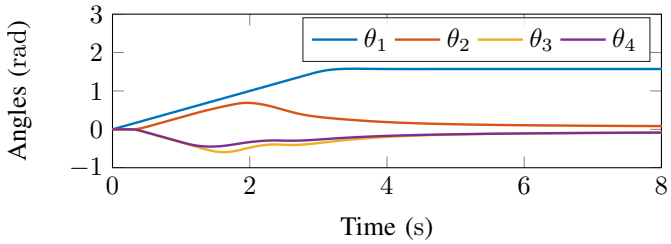


Fig. 6: Angles $\theta_1, \theta_2, \theta_3, \theta_4$ against time for the MPC-controlled system.


[11] R. Grandia, F. Farshidian, R. Ranftl, and M. Hutter, "Feedback MPC for torque-controlled legged robots," *arXiv:1905.06144*, 2019.

[12] Z. Chao, L. Ming, Z. Shaolei, and Z. Wenguang, "Collision-free UAV formation flight control based on nonlinear MPC," in *ICECC*, pp. 1951–1956, IEEE, 2011.

[13] N. Paperno, M. Rupp, E. M. Maboudou-Tchao, J. A. Smither, and A. Behal, "A predictive model for use of an assistive robotic manipulator: Human factors versus performance in pick-and-place/retrieval tasks," *IEEE Transf Human-Machine Sys*, vol. 46, pp. 846–858, Dec 2016.

[14] J. G. Storms and D. M. Tilbury, "Blending of human and obstacle avoidance control for a high speed mobile robot," in *ACC*, pp. 3488–3493, 2014.

[15] M. Faroni, M. Beschi, and N. Pedrocchi, "An MPC framework for online motion planning in human-robot collaborative tasks," in *IEEE ETFA*, pp. 1555–1558, Sep 2019.

[16] T. A. Johansen, "Toward dependable embedded model predictive control," *IEEE Systems Journal*, vol. 11, pp. 1208–1219, June 2017.

[17] S.-M. Lee and H. Myung, "Receding horizon particle swarm optimisation-based formation control with collision avoidance for non-holonomic mobile robots," *IET Contr Theory & Appl*, vol. 9, no. 14, pp. 2075–2083, 2015.

[18] A. Carron, E. Arcari, M. Wermelinger, L. Hewing, M. Hutter, and M. N. Zeilinger, "Data-driven model predictive control for trajectory tracking with a robotic arm," *IEEE Rob Aut Lett*, vol. 4, pp. 3758–65, Oct 2019.

[19] T. Hu, T. Wang, J. Li, and W. Qian, "Obstacle avoidance for redundant manipulators utilizing a backward quadratic search algorithm," *Int J Adv Rob Sys*, vol. 13, no. 3, p. 119, 2016.

[20] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Mathematical programming*, vol. 106, no. 1, pp. 25–57, 2006.

[21] P. Tangpattanakul, A. Meesomboon, and P. Artrit, "Optimal trajectory of robot manipulator using harmony search algorithms," in *Recent advances in harmony search algorithm*, pp. 23–36, Springer, 2010.

[22] B. Hermans, G. Pipeleers, and P. Patrinos, "A penalty method algorithm for obstacle avoidance using nonlinear model predictive control," in *Benelux Meeting Syst & Contr*, (Soesterberg, The Netherlands), 2018.

[23] L. Stella, A. Themelis, P. Sopasakis, and P. Patrinos, "A simple and efficient algorithm for nonlinear model predictive control," in *IEEE CDC*, pp. 1939–44, IEEE, 2017.

[24] S. S. Mansouri, C. Kanellakis, E. Fresk, B. Lindqvist, D. Kominiak, A. Koval, P. Sopasakis, and G. Nikolakopoulos, "Subterranean MAV navigation based on nonlinear MPC with collision avoidance constraints," in *IFAC World Congress*, (Berlin, Germany), 2020.

[25] E. Small, P. Sopasakis, E. Fresk, P. Patrinos, and G. Nikolakopoulos, "Aerial navigation in obstructed environments with embedded nonlinear model predictive control," in *IEEE European Control Conference (ECC)*, pp. 3556–3563, June 2019.

[26] A. Sathya, P. Sopasakis, R. Van Parys, A. Themelis, G. Pipeleers, and P. Patrinos, "Embedded nonlinear model predictive control for obstacle avoidance using PANOC," in *IEEE European Control Conference (ECC)*, pp. 1523–1528, June 2018.

[27] B. Lindqvist, S. S. Mansouri, P. Sopasakis, and G. Nikolakopoulos, "Collision avoidance for multiple MAVs using fast centralized NMPC," in *IFAC World Congress*, (Berlin, Germany), 2020.

[28] E. G. Birgin and J. M. Martínez, *Practical augmented Lagrangian methods for constrained optimization*, vol. 10. SIAM, 2014.

[29] P. Sopasakis, E. Fresk, and P. Patrinos, "OpEn: Code generation for embedded nonconvex optimization," in *IFAC World Congr*, (Berlin, Germany), 2020. Software at: `doc.optimization-engine.xyz/`, arXiv:2003.00292.

[30] A. M. Jasour and M. Farrokhi, "Path tracking and obstacle avoidance for redundant robotic arms using fuzzy NMPC," in *ACC*, pp. 1353–58, 2009.

[31] 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, pp. 1505–11, July 2017.

[32] B. K. Kim and K. G. Shin, "Minimum-time path planning for robot arms and their dynamics," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-15, pp. 213–223, March 1985.

[33] K. Yamato, H. Chiba, T. Yamashita, and H. Oku, "1-ms three-dimensional feedback microscope with 69-khz synchronous modulation of focal position and illumination," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1978–1984, 2018.

[34] A. Pokle, R. Martín-Martín, P. Goebel, V. Chow, H. M. Ewald, J. Yang, Z. Wang, A. Sadeghian, D. Sadigh, S. Savarese, *et al.*, "Deep local trajectory replanning and control for robot navigation," in *ICRA*, pp. 5815–5822, IEEE, 2019.

[35] C. G. Prevost, A. Desbiens, and E. Gagnon, "Extended Kalman filter for state estimation and trajectory prediction of a moving object detected by an unmanned aerial vehicle," in *ACC*, pp. 1805–10, July 2007.

[36] D. Aguilar-Lleyda, E. Tubau, and J. Lopez-Moliner, "An object-tracking model that combines position and speed explains spatial and temporal responses in a timing task," *J Vision*, vol. 18, no. 12, p. 12, 2018.

[37] T.-L. Yang, A. Liu, H. Shen, L. Hang, Y. Luo, and Q. Jin, "Position and orientation characteristics equation for serial mechanisms," in *Topology Design of Robot Mechanisms*, pp. 43–66, Springer, 2018.

[38] B. Hermans, P. Patrinos, and G. Pipeleers, "A penalty method based approach for autonomous navigation using nonlinear model predictive control," *IFAC-PapersOnLine*, vol. 51, no. 20, pp. 234 – 240, 2018.

[39] J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, "CasADi – A software framework for nonlinear optimization and optimal control," *Math Progr Comp*, vol. 11, pp. 1–36, July 2018.