2021 IEEE International Conference on
Autonomous Robot Systems and Competitions (ICARSC)
April 28-29, Santa Maria da Feira, Portugal

2021 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC) | 978-1-6654-3198-9/20/$31.00 ©2021 IEEE | DOI: 10.1109/ICARSC52212.2021.9429795

Payload transportation in microgravity with single and multiple cooperative free-flyer robots

Rui Correia
Institute for Systems and Robotics
Instituto Superior Técnico
Av. Rovisco Pais, Lisbon, Portugal
orcid.org/0000-0003-0247-1343

Rodrigo Ventura
Institute for Systems and Robotics
Instituto Superior Técnico
Av. Rovisco Pais, Lisbon, Portugal
orcid.org/0000-0002-5655-9562

***Abstract*—The use of free-flyer robots in space applications has been increasing in recent years. Several space agencies, such as NASA, DLR and JAXA, have at this moment, or intend to have, projects involving this type of vehicles. The increasing interest in these robots is largely motivated by the expansion of human presence beyond low earth orbit, to cislunar space and beyond. In particular, the envisioned Deep Space Gateway (DSG) will require autonomous robots performing logistics operations. This work presents a nonlinear model predictive control based method targeting a free-flyer robot in microgravity to perform 3 different tasks: waypoint navigation, single robot transportation, and multi-robot transportation. This controller is coupled with a trajectory generation method that provides feasible trajectory references to the controller, and is capable of handling confined spaces, composed by walls and obstacles, as well as control actuation limitations and system dynamics. All tests conducted in this work were obtained through realistic simulation based in ROS and Gazebo/RotorS.**

***Index Terms*—space robotics, multi-robot load transportation, mobile manipulation, nonlinear model predictive control, trajectory optimization**

I. Introduction

The Global Exploration Roadmap [1] of the International Space Exploration Coordination Group (ISECG) targets expanding human presence to Mars using the return of humans to the Moon as a stepping stone. In this process, the DSG is a space station being designed to orbit the Moon, primarily to support this return. One key requirement for this space station is the capability of performing logistics operations using autonomous robots, during the periods the station will not be inhabited. To this end, the International Space Station (ISS) presents an exceptional lab environment to the development and validation of such robots. National Aeronautics and Space Administration (NASA) Synchronized Position Hold Engage Reorient Experimental Satellites (SPHERES) were the first platform used for many experiments conducted in the ISS [2]. This research has allowed the development of NASA's Astrobee [3] that replaced SPHERES as a testbed onboard the ISS. Other examples of free-flyers are the Int-Ball [4] and the Crew Interactive MObile companioN (CIMON) [5]. Currently, robotic arms are already used to assist humans in maintenance tasks in the ISS, namely the *Canadarm2* [6]. However, in a remote space station there could be significant limitations to these systems. Free-flyer robots will allow more complex operations without human presence. In order to achieve this level of autonomy, several aspects have to be extensively studied, such as the vehicle interaction with the station, objects, and astronauts. Its navigation system must guarantee high awareness and risk-free maneuvers. Lastly, the control system must use the least amount of energy possible as access to energy might be severely restricted.

Nonlinear Model Predictive Control (NMPC) is an optimization based method for the feedback control of nonlinear systems, which consists in solving an Optimal Control Problem (OCP) and applying the first element of the solution set to the system at each time step and discarding the rest [7]. Generally, NMPC cannot rely on analytical control laws, alternatively, it relies on nonlinear numerical and optimization methods to solve the OCP. In general, a transcription process is necessary to obtain a parameter optimization problem with finite dimension, a Nonlinear Programming (NLP) problem, which can be solved by general NLP solvers [8].

NMPC is very versatile as it can take several roles within the control system, such as trajectory planning as in [9], where NMPC is proposed to adjust a trajectory in the presence of obstacles by incorporating new constraints during operation. NMPC is also being considered to perform low level control tasks, which require fast and stable rates of operation. The hierarchical controller implemented in [10] uses a NMPC controller for the attitude of Multi-rotor Aerial Vehicles (MRAV) and registered execution averages of 1 millisecond while operating at a 5 ms sampling time. In [11], a single non-hierarchical real-time NMPC controller implemented in a fixed wing aircraft is used to control both position and attitude for tracking a given reference trajectory while avoiding obstacles and respecting actuator bounds. The execution time required by the NMPC controller averages at 30 ms while operating at a sampling time of 500 ms. This can be an indication of the impact of system complexity on execution time. Also implemented in a MRAV, in [12], a hierarchical controller is used to track a feasible trajectory generated offline. A position NMPC controller computes attitude references feeding a second attitude NMPC controller, operating with sampling times of 200 ms and 2 ms, respectively. The application of NMPC

This work was supported by the LARSyS - FCT Project UIDB/50009/2020.

in a MRAV with Multi-Directional Thrust (MDT) capability is studied in [13] where a NMPC controller is used to compute force inputs for each rotor to track both position and attitude references.

This paper contributes with a control architecture for single robot payload transportation and multi-robot cooperative transportation, in the presence of obstacles, for free-flyer robots in microgravity. This architecture is based on an optimal trajectory generator, coupled with a position and attitude NMPC controller, and is validated and evaluated in a realistic simulator using as model the Space CoBot[1] [14], which is a hexarotor holonomic free-flyer platform.

The remainder of this paper is organized as follows: the free-flyer and transport configuration models are described in Section II. The proposed system architecture is presented in Section III. In Section IV simulation results are presented and discussed. Lastly, conclusions are presented in Section V.

II. Models

Throughout this work, three operational scenarios are considered. A single free-flyer robot, a single-robot transport system composed by a free-flyer robot and payload, and a multi-robot transport system comprising two or more free-flyer robots and a shared payload. The possible scenarios are presented in Fig. 1. To observe the impact of operational scenarios in the NMPC controller performance, a similar goal will be considered for each scenario consisting in navigating between two given points in a constrained environment composed by walls and possibly one or more obstacles.

A. Free-flyer robot model

This model describes a single multi-rotor free-flyer robot, which is modelled through the Newton-Euler equations of motion following [14]. In this work, quaternions are used to represent attitude. Let us denote the position and linear velocity of the body frame $\mathcal{B}$, centered in the Center of Mass (CoM) of the robot, with respect to (w.r.t.) the inertial frame $\mathcal{I}$ as $\mathbf{p} = [x\,, y\,, z]^T$ and $\mathbf{v} = [u\,, v\,, w]^T$, the orientation quaternion, rotation matrix, and angular velocity of frame $\mathcal{B}$ w.r.t. $\mathcal{I}$ as $\mathbf{q} = [q_w\,, q_x\,, q_y\,, q_z]^T$, $\mathbf{R}$ and $\boldsymbol{\omega} = [\omega_x,\, \omega_y,\, \omega_z]^T$. These elements constitute the state vector

$$\mathbf{x} = [\mathbf{p}^T, \mathbf{v}^T, \mathbf{q}^T, \boldsymbol{\omega}^T]^T \tag{1}$$

The resulting model is described by

$$\begin{cases} \dot{\mathbf{p}} = \mathbf{v} \\ \dot{\mathbf{v}} = (1/m)\,\mathbf{R}\,\mathbf{F}^{\mathcal{B}} \\ \dot{\mathbf{q}} = (1/2)\,\boldsymbol{\Omega}(\mathbf{q})\,\boldsymbol{\omega} \\ \dot{\boldsymbol{\omega}} = \mathbf{J}^{-1}(\mathbf{M}^{\mathcal{B}} - \boldsymbol{\omega} \times \mathbf{J}\boldsymbol{\omega}) \end{cases} \tag{2}$$

where $\times$ is the cross product operator, the scalar m is the system's mass, $\mathbf{J}$ is the moment of inertia matrix about the CoM, and $\mathbf{F}^{\mathcal{B}}$ and $\mathbf{M}^{\mathcal{B}}$ are vectors representing the thrust force and torque generated by the rotors expressed in the body frame $\mathcal{B}$. $\boldsymbol{\Omega}$ is a 4×3 matrix defined by

$$\boldsymbol{\Omega}(\mathbf{q}) = \begin{bmatrix} q_w & -q_z & q_y \\ q_z & q_w & -q_x \\ -q_y & q_x & q_w \\ -q_x & -q_y & -q_z \end{bmatrix} \tag{3}$$

The propulsive system model follows the approach in [14] where each rotor i generates a scalar thrust f_i and a torque τ_i described by

$$f_i = K_1 u_i, \qquad \tau_i = w_i K_2 u_i \tag{4}$$

where K_1 and K_2 are constants used to describe the propeller aerodynamic factors, w_i is either -1 or 1 depending on whether the propeller rotates clockwise or anti-clockwise for a positive forward thrust $f_i > 0$. The actuation signal u_i is defined by $u_i = \text{sgn}(n_i)\, n_i^2$, to achieve a linear relation between the actuation and forces/moments, where sgn() is a sign function and n_i is expressed in revolutions per second. Each rotor is represented by a position vector relative to the CoM, $\mathbf{r}_i$, orthogonal to the body's z axis, and the unit vector $\hat{u}_i$, aligned with the propeller orientation axis. The resulting thrust $\mathbf{F}_i^{\mathcal{B}}$ and torque $\mathbf{M}_i^{\mathcal{B}}$ are obtained

$$\begin{pmatrix} \mathbf{F}_i^{\mathcal{B}} \\ \mathbf{M}_i^{\mathcal{B}} \end{pmatrix} = \mathbf{a}_i u_i \tag{5}$$

where $\mathbf{a}_i$ represents the components of force and torque of each rotor i and is defined by

$$\mathbf{a}_i = \begin{pmatrix} K_1 \hat{u}_i \\ K_1 \mathbf{r}_i \times \hat{u}_i - w_i K_2 \hat{u}_i \end{pmatrix} \tag{6}$$

The resulting net force and torque will be the sum of the contributions off all rotors. A free-flyer with 6 rotors results in

$$\begin{pmatrix} \mathbf{F}^{\mathcal{B}} \\ \mathbf{M}^{\mathcal{B}} \end{pmatrix} = \mathbf{A}\mathbf{u} \tag{7}$$

where $\mathbf{A} = [\mathbf{a}_1 \,...\, \mathbf{a}_6]$ is a square matrix hereby called actuation matrix and $\mathbf{u} = [u_1 \,...\, u_6]^T$ is the actuation input.

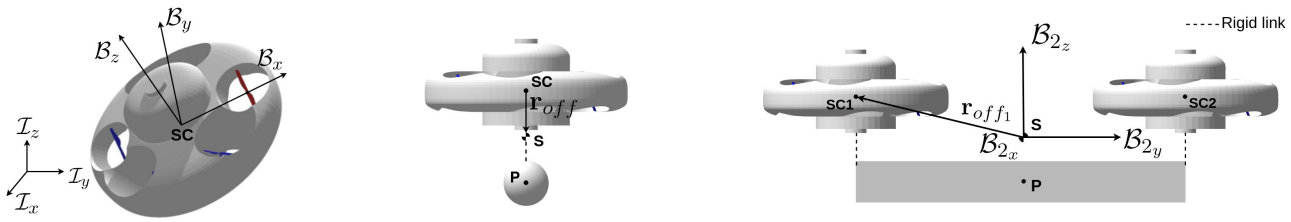
Even though a multi-rotor propulsion system with 6 rotors based in [14] is considered in this section, our approach does not strictly depend on it, since (7) is applicable to a broad range of propulsion systems, provided that the robot is fully actuated.

B. Single-robot transport model

The addition of a payload will cause some changes in the resulting model due to the displacement in the CoM of the resulting system. In order to model the payload, rigid body conditions are considered for the resulting system, observed in Fig. 1b. To describe the velocity and acceleration of the CoM of the robot w.r.t. to $\mathcal{I}$ and the rotation of the system in $\mathcal{B}$ some adjustments are required. The offset between the CoM of the free-flyer and the CoM of the system is denoted by $\mathbf{r}_{off}$, expressed in $\mathcal{B}$. The position of each rotor w.r.t. the CoM of the system is now $\mathbf{r}_{p_i}$ and is defined by

$$\mathbf{r}_{p_i} = \mathbf{r}_i - \mathbf{r}_{off} \tag{8}$$

[1] http://space-cobot.isr.tecnico.ulisboa.pt/



(a) Single free-flyer robot (b) Single-robot transport (c) Multi-robot transport

Fig. 1. Illustration of the considered operational scenarios based in the Space CoBot free-flyer with the attached inertial frame $\mathcal{I}$ and body-fixed frames $\mathcal{B}$ and $\mathcal{B}_2$. Points **SC**, **SC1** and **SC2** represent the center of mass of free-flyers and points **S** and **P** represent the centers of mass of the system and payload, respectively. Vector $\mathbf{r}_{off}$ represents the offset between the centers of mass of a free-flyer and the system expressed in (b) $\mathcal{B}$ and (c) $\mathcal{B}_2$.

The torque of each rotor, $\mathbf{M}_i^{\mathcal{B}}$, in (5) is modified and defined by

$$\mathbf{M}_i^{\mathcal{B}} = K_1 \mathbf{r}_{p_i} \times \hat{u}_i - w_i K_2 \hat{u}_i \tag{9}$$

As the frame $\mathcal{B}$ is not coincident with the CoM of the system, the resulting model is then

$$\begin{cases} \dot{\mathbf{p}} = \mathbf{v} \\ \dot{\mathbf{v}} = \mathbf{R}\left[\mathbf{F}^{\mathcal{B}}/m - \dot{\boldsymbol{\omega}} \times \mathbf{r}_{off} - \boldsymbol{\omega} \times (\boldsymbol{\omega} \times \mathbf{r}_{off})\right] \\ \dot{\mathbf{q}} = (1/2)\,\boldsymbol{\Omega}(\mathbf{q})\,\boldsymbol{\omega} \\ \dot{\boldsymbol{\omega}} = \mathbf{J}_S^{-1}(\mathbf{M}^{\mathcal{B}} - \boldsymbol{\omega} \times \mathbf{J}_S \boldsymbol{\omega}) \end{cases} \tag{10}$$

where $\mathbf{J}_S$ is the inertia matrix of the system about its CoM.

C. Multi-robot transport model

The multi-robot configuration will also be considered as a single system where all elements are rigidly linked, comprising a rigid body. In this case, the model will be defined around the CoM of the resulting system, Fig. 1c. Let us then denote the body-fixed frame $\mathcal{B}_2$ centered in the CoM of the formation, and $\mathbf{p}_f$ and $\mathbf{v}_f$ as the position and velocity of frame $\mathcal{B}_2$ w.r.t. the inertial frame $\mathcal{I}$. The offset between the CoM of the j^{th} robot in the formation and the CoM of the system is denoted by $\mathbf{r}_{off_j}$, expressed in frame $\mathcal{B}_2$. The positions of the rotors of a given free-flyer will be described by

$$\mathbf{r}_{f_i} = \mathbf{r}_i + \mathbf{r}_{off_j} \tag{11}$$

The resulting thrust and torque are defined by

$$\begin{pmatrix} \mathbf{F}_i^{\mathcal{B}_2} \\ \mathbf{M}_i^{\mathcal{B}_2} \end{pmatrix} = \mathbf{a}_i u_i, \qquad \mathbf{a}_i = \begin{pmatrix} K_1 \hat{u}_i \\ K_1 \mathbf{r}_{f_i} \times \hat{u}_i - w_i K_2 \hat{u}_i \end{pmatrix} \tag{12}$$

The resulting net force and torque will once again be the sum of the contributions of all rotors in the formation. In matrix form

$$\begin{pmatrix} \mathbf{F}^{\mathcal{B}_2} \\ \mathbf{M}^{\mathcal{B}_2} \end{pmatrix} = \sum_{k=1}^{K} \mathbf{A}_k \mathbf{u}_k \tag{13}$$

where $\mathbf{A}_k$ and $\mathbf{u}_k$ are the actuation matrix and control inputs of each free-flyer, respectively, and $K > 1$ is the number of free-flyers in the formation. The resulting actuation signal for this configuration is $\mathbf{u} = \left[\mathbf{u}_1^T \ldots \mathbf{u}_K^T\right]^T$.

The corresponding model can then be described through

$$\begin{cases} \dot{\mathbf{p}}_f = \mathbf{v} \\ \dot{\mathbf{v}}_f = (1/m)\,\mathbf{R}\,\mathbf{F}^{\mathcal{B}_2} \\ \dot{\mathbf{q}} = (1/2)\,\boldsymbol{\Omega}(\mathbf{q})\,\boldsymbol{\omega} \\ \dot{\boldsymbol{\omega}} = \mathbf{J}_S^{-1}(\mathbf{M}^{\mathcal{B}_2} - \boldsymbol{\omega} \times \mathbf{J}_S \boldsymbol{\omega}) \end{cases} \tag{14}$$

Note that the states of the model mentioned above in (14) are described in the CoM of the formation while each free-flyer robot has sensors positioned onboard. Therefore, this information is transformed via the following equations

$$\mathbf{p}_f = \mathbf{p}_{SC_j} - \mathbf{R}\,\mathbf{r}_{off_j} \tag{15a}$$
$$\mathbf{v}_f = \mathbf{v}_{SC_j} - \mathbf{R}\,(\boldsymbol{\omega} \times \mathbf{r}_{off_j}) \tag{15b}$$
$$\mathbf{a}_f = \mathbf{a}_{SC_j} - \mathbf{R}\left[\dot{\boldsymbol{\omega}} \times \mathbf{r}_{off_j} + \boldsymbol{\omega} \times (\boldsymbol{\omega} \times \mathbf{r}_{off_j})\right] \tag{15c}$$

III. System Architecture

In this section, the proposed architecture is presented. The desired trajectories will be generated with the Augmented Lagrangian TRajectory Optimizer (ALTRO) [15] for all the states in the system. The controller is based on NMPC and computes the individual motor thrusts required to track the reference trajectory. Note that in the multi-robot configuration, each robot computes the full control input $\mathbf{u}$ but only uses the input that corresponds to itself. At this point, state measurements with small errors are provided directly to the controller without an estimation step.

A. Trajectory generation

To provide a feasible reference trajectory to the controller, an OCP is formulated and solved with ALTRO [15] for the considered tasks. As different OCPs can have significant variations in computation time, this method will be performed offline before any movement execution, rendering its computation time not critical. The optimization horizon of this method, composed by N_{TO} points, is also considerably larger than the one considered in the NMPC controller to guarantee that the objective state is reached. The same models presented in section II are used to generate dynamically feasible trajectories for

each operational scenario. The objective function is composed by the cost function J and gives rise to the OCP

$$\min \quad J := (\mathbf{x}_{N_{TO}} - \mathbf{x}_f)^T \mathbf{Q}_f (\mathbf{x}_{N_{TO}} - \mathbf{x}_f) + \sum_{k=1}^{N_{TO}-1} \left[(\mathbf{x}_k - \mathbf{x}_f)^T \mathbf{Q} (\mathbf{x}_k - \mathbf{x}_f) + \mathbf{u}_k^T \mathbf{R}_u \mathbf{u}_k \right] \tag{16a}$$

$$\text{s.t.} \quad \mathbf{x}_k,\ \mathbf{x}_{N_{TO}} \in \mathcal{X} \quad \& \quad \mathbf{u}_k \in \mathcal{U} \tag{16b}$$

$$\mathbf{x}_{k+1} = f(\mathbf{x}_k, \mathbf{u}_k) \tag{16c}$$

where $\mathcal{X}$ represents the configuration space set and $\mathcal{U}$ the feasible actuation space set. $\mathbf{Q}_f$, $\mathbf{Q}$ and $\mathbf{R}_u$ are diagonal weighting matrices, associated to states and control, respectively. $\mathbf{x}_f$ is the desired terminal state. Several constraints can be considered in this OCP, which modify the sets $\mathcal{X}$ and $\mathcal{U}$, such as goal constraints (17a), state bound constraints (17b), control bound constraints (17c), and obstacle constraints (17d), that can vary both in number and shape.

$$\mathbf{x}_{N_{TO}} = a \tag{17a}$$

$$\mathbf{x}_{\min} \leq \mathbf{x}_k \leq \mathbf{x}_{\max} \tag{17b}$$

$$\mathbf{u}_{\min} \leq \mathbf{u}_k \leq \mathbf{u}_{\max} \tag{17c}$$

$$d_{obs}^2(p_k) \geq r_{clear}^2 \tag{17d}$$

where $\mathbf{x}_{\min}$, $\mathbf{x}_{\max}$, $\mathbf{u}_{\min}$ and $\mathbf{u}_{\max}$ are vectors comprising the individual states and control bounds, respectively. $d_{obs}(p_k)$ is the euclidean distance between the position of the free-flyer, $\mathbf{p}_k$, and the closest point of an obstacle, and r_{clear} is the clearance distance from the obstacle. The optimal trajectory $\mathbf{x}^{opt} \in \mathbb{R}^{13 \times N_{TO}}$ is obtained.

B. NMPC formulation

To formulate the NMPC controller, the state considered is $\mathbf{x}$, as in (1). To lower the magnitude of the control input, it is expressed in force (newtons), $\mathbf{u}_f$, defined by

$$\mathbf{u}_f = \lambda_{rpm} \mathbf{u}_{rpm} \tag{18}$$

where λ_{rpm} is a conversion factor defined by

$$\lambda_{rpm} = \frac{K_1}{60^2} \tag{19}$$

that converts the control input between force and angular velocity (rotations per minute), which is useful for implementation purposes. At each time step the following OCP will be solved

$$\min \int_t^{t+t_N} h \, dt + h_N \tag{20a}$$

$$\text{s.t.} \quad \dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u}_f, t) \tag{20b}$$

$$p(\mathbf{x}, \mathbf{u}_f, t) \leq 0, \tag{20c}$$

$$g(t_0, t_f, \mathbf{x}(t_0), \mathbf{x}(t_f)) \leq 0 \tag{20d}$$

where (20b) are the system dynamics, (20c) represents path constraints and (20d) represents boundary constraints,

$$h = [\Delta \mathbf{p}^T, \delta \mathbf{q}^T, \mathbf{u}_f^T] \mathbf{W} [\Delta \mathbf{p}^T, \delta \mathbf{q}^T, \mathbf{u}_f^T]^T \tag{21}$$

is the integral objective function for the position and attitude errors, and control input magnitude and

$$h_N = [\Delta \mathbf{p}_N^T, \delta \mathbf{q}_N^T] \mathbf{W}_N [\Delta \mathbf{p}_N^T, \delta \mathbf{q}_N^T]^T \tag{22}$$

is the boundary objective function for the position and attitude errors defined by

$$\Delta \mathbf{p} = \mathbf{p} - \mathbf{p}_{ref} \tag{23a}$$

$$\delta \mathbf{q} = \begin{bmatrix} q_w^{ref} q_x + q_z^{ref} q_y - q_y^{ref} q_z - q_x^{ref} q_w \\ -q_z^{ref} q_x + q_w^{ref} q_y + q_x^{ref} q_z - q_y^{ref} q_w \\ q_y^{ref} q_x - q_x^{ref} q_y + q_w^{ref} q_z - q_z^{ref} q_w \end{bmatrix} \tag{23b}$$

where $\delta \mathbf{q} = [\delta q_x, \delta q_y, \delta q_z]^T$ is a truncated version of the quaternion error described in [16], $\mathbf{W}$ and $\mathbf{W}_N$ are diagonal matrices comprising the individual cost weights for the members of the integral objective function and boundary objective function, respectively. The size of $\mathbf{W}$ can vary depending on the operational scenario.

C. NMPC real-time implementation

To achieve a NMPC formulation, an NLP must be formulated. A direct multiple shooting technique is used to transform the OCP into a NLP, where the control input is discretized piecewise on a given time grid, $\mathbf{u}(t) = \mathbf{c}_i$, along with the considered system dynamics and any given constraints giving way to a Boundary Value Problem (BVP) for each interval

$$\dot{\mathbf{x}}(t) = f(\mathbf{x}_i(t), \mathbf{c}_i), \quad t \in [t_i, t_{i+1}] \tag{24a}$$

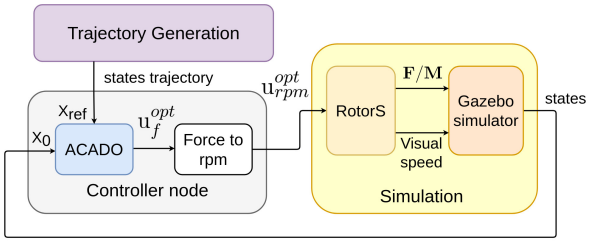
$$\mathbf{x}_i = \mathbf{s}_i \tag{24b}$$

where $\mathbf{s}_i$ is called a shooting node. Additional continuity constraints, $\mathbf{s}_{i+1} = \mathbf{x}_i(t_{i+1}, \mathbf{s}_i, \mathbf{c}_i)$, are also imposed to guarantee dynamic feasibility between intervals. The set of optimization variables is then $\mathbf{z} := \{\mathbf{s}_0, \mathbf{c}_0, ..., \mathbf{s}_{N+1}, \mathbf{c}_N\}$, where N is the number of intervals in the optimization horizon.

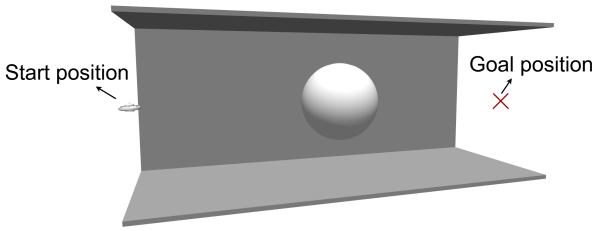
This NLP is solved using Sequential Quadratic Programming (SQP). In this work, the ACADO toolkit [17] is used to generate a real-time solver in efficient C code, including the Quadratic Programming (QP) solver *qpoases* [18], which is the one selected to solve the SQP subproblems. A Real Time Iteration (RTI) [19] scheme is used based on a Gauss-Newton approach [20], which avoids the computation of second order derivatives and, consequently, improves computation speed.

IV. Simulation Results

In this section, the simulation results for the proposed controller are presented. The solver generated in section III-C is included in a Robot Operating System (ROS) node, *controller node*, operating at 10 hertz with a prediction horizon set to $T_h = 2$ s. The weighting costs for control are set to 1 for single-robot scenarios and 0.1 for the multi-robot scenario. The weighting costs for the position and attitude elements are set equally to 20, 50 and 500 for the single free-flyer, single-robot transport and multi-robot transport scenarios, respectively. Trajectory generation considers a constrained environment, visible in Fig. 2b, composed by walls and a single spherical obstacle, with (17d) taking the form $\|\mathbf{p}_k - \mathbf{p}_c\|^2 \geq r_c^2$,



(a) Complete simulation scheme



(b) Single-robot simulation environment setup

Fig. 2. Visualization of the simulation scheme (a) and an experiment environment (b). The closest wall is omitted for clarity.

where $\mathbf{p}_c$ and r_c are the center and radius of the sphere, respectively. r_c is composed by the obstacle radius of 1 m plus a clearance radius $r_{clear} = 0.5$ m for single-robot scenarios and $r_{clear} = 1$ m for multi-robot scenarios. The time interval is selected to match the controller operation rate. The simulation environment is composed by the Gazebo simulator[2] and the MRAV simulator RotorS [21], depicted in Fig. 2a. All simulations were conducted in a laptop computer with an Intel Core i7-4710HQ @ 2.50 GHz.

A. Trajectory execution

To evaluate the performance of the NMPC controller, the trajectory tracking errors based in (23) will be analyzed along with violations to the Karush-Kuhn-Tucker (KKT) optimality conditions and objective function values at each time step for a single trial of each operational scenario. The single-robot experiments are conducted in a $4 \times 10 \times 4$ m room whereas the multi-robot experiment takes place in a $8 \times 10 \times 4$ m room.

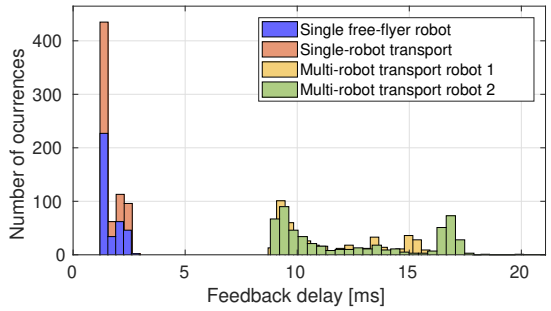


Fig. 3. NMPC controller feedback delay for the different operational scenarios.

1) Single free-flyer: The start and goal positions are selected to be 5 m away in opposite sides of the obstacle. Trajectory generation takes around 892 s to find a feasible optimal solution. Observing the results in Fig. 4a, the controller demonstrates a good tracking performance. Additionally, the obtained solutions present very small violations of the KKT conditions and the objective function converges to zero, which results in a smooth trajectory execution. The dispersion of the feedback delay in Fig. 3, shows that the computation time is not very sparse and causes an average feedback delay of 1.8 ms.

[2]http://gazebosim.org/

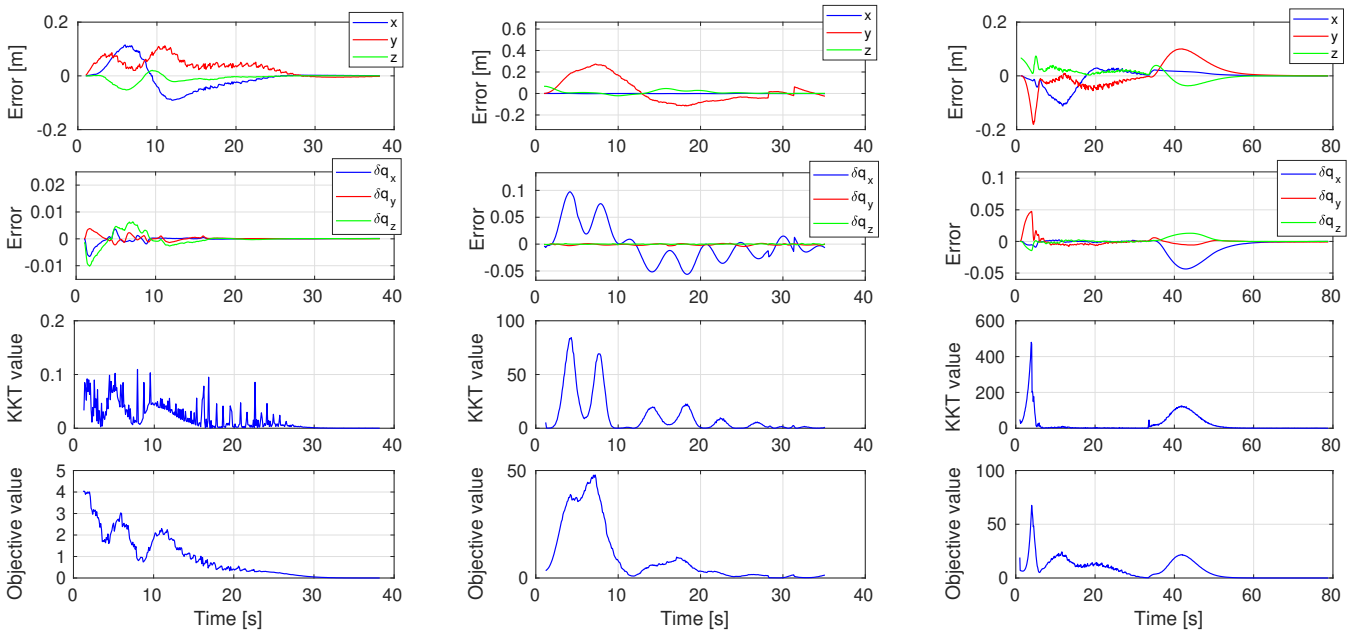
2) Single-robot transport: The same conditions are considered in this simulation. Trajectory generation takes 2903 s to achieve a solution. In Fig. 4b, the increase in model complexity clearly increases the tracking errors. Moreover, the increase in system mass and CoM displacement require higher actuation inputs to move and stabilize the system, which increases the objective value and could be making it hard to find good solutions. The significant increase in KKT violations indicate that more iterations may be required to obtain a better solution. In spite of this, the objective function still converges to zero. In Fig. 3, feedback delay is very similar to the single free-flyer case, which indicates that the increase in model complexity did not impact computation time.

3) Multi-robot transport: In this simulation, the start and goal positions are selected to be 8 m away. A trajectory solution is found in 2601 s. The results obtained via one of the free-flyers in the formation can be observed in Fig. 4c. The controller still demonstrates a good tracking performance. However, tracking errors are present not only in the beginning of the trajectory but also towards the end. In Fig. 3, computation times are more sparse and cause an average feedback delay of 12.5 ms, contributing to inconsistency and consequent tracking errors. This delay in execution also eventually leads to more aggressive control inputs to resume the desired trajectory. It is easy to identify inconsistent solutions as KKT violations tend to increase significantly. Nonetheless, the objective function still converges to zero.

V. Conclusions

The system architecture proposed in this work presents promising results for navigating under several operational scenarios, with and without load, single and multi-robot, where a generated optimal trajectory reference is tracked by a NMPC controller. Being this trajectory dynamically feasible, the role of the NMPC controller is to reject real-time disturbances. However, this trajectory generation method requires a high computation time and had to be performed at a significantly lower rate than the controller.

An experimental study of the NMPC controller limits was conducted finding that it is able to operate in real-time close to the microsecond range. Increases in computation time are expected if the OCP dimension increases, as it happens with the multi-robot transport configuration, where the increase in control input size greatly increases the computation time. In



(a) Single free-flyer (b) Single-robot transport (c) Multi-robot transport

Fig. 4. Trajectory execution simulation results for different operational scenarios. From top to bottom: Position reference tracking error in x, y and z axis; Attitude reference tracking error described in a truncated quaternion error, $\delta\mathbf{q}$ (23b); Optimality condition (KKT) violation values; Objective function values.

addition, the quality of the solution also decreases and the controller is not able to smoothly track the trajectory.

Future work includes coping with localization uncertainty, as well as seeking a decentralized version of this architecture, to improve scalability and decrease reliance on communication.

REFERENCES

[1] ISECG, *The Global Exploration Roadmap*, 3rd ed. ISECG, 2018.

[2] S. Mohan, A. Saenz-Otero, S. Nolet, D. W. Miller, and S. Sell, "SPHERES flight operations testing and execution," *Acta Astronautica*, vol. 65, no. 7-8, pp. 1121–1132, 2009.

[3] M. Bualat, J. Barlow, T. Fong, C. Provencher, T. Smith, and A. Zuniga, "Astrobee: Developing a free-flying robot for the international space station," *AIAA SPACE 2015 Conference and Exposition*, pp. 1–10, 2015.

[4] S. Mitani, M. Goto, R. Konomura, Y. Shoji, K. Hagiwara, S. Shigeto, and N. Tanishima, "Int-Ball: Crew-Supportive Autonomous Mobile Camera Robot on ISS/JEM," *IEEE Aerospace Conference Proceedings*, vol. 2019-March, pp. 1–15, 2019.

[5] DLR, "Experiment CIMON - astronaut assistance system," 2018. [Online]. Available: https://www.dlr.de/content/en/articles/missions-projects/horizons/experimente-horizons-cimon.html

[6] S. Sachdev, B. Marcotte, and G. Gibbs, "Canada and the international space station program: Overview and status," *International Astronautical Federation - 55th International Astronautical Congress 2004*, vol. 11, no. 1, pp. 7405–7415, 2004.

[7] L. Grüne and J. Pannek, *Nonlinear Model Predictive Control: Theory and Algorithms*. Springer London, 2011.

[8] D. Kouzoupis, G. Frison, A. Zanelli, and M. Diehl, "Recent Advances in Quadratic Programming Algorithms for Nonlinear Model Predictive Control," *Vietnam Journal of Mathematics*, vol. 46, no. 4, pp. 863–882, 2018.

[9] D. H. Shim, H. Chung, H. J. Kim, and S. Sastry, "Autonomous exploration in unknown urban environments for unmanned aerial vehicles," *Collection of Technical Papers - AIAA Guidance, Navigation, and Control Conference*, vol. 8, pp. 6381–6388, 2005.

[10] M. Kamel, K. Alexis, M. Achtelik, and R. Siegwart, "Fast nonlinear model predictive control for multicopter attitude tracking on SO(3)," *2015 IEEE Conference on Control and Applications, CCA 2015 - Proceedings*, no. 3, pp. 1160–1166, 2015.

[11] S. Gros, R. Quirynen, and M. Diehl, "Aircraft control based on fast non-linear MPC & multiple-shooting," *Proceedings of the IEEE Conference on Decision and Control*, 2012.

[12] P. Lin, S. Chen, and C. Liu, "Model predictive control-based trajectory planning for quadrotors with state and input constraints," *International Conference on Control, Automation and Systems*, 2016.

[13] D. Bicego, J. Mazzetto, R. Carli, M. Farina, and A. Franchi, "Nonlinear Model Predictive Control with Actuator Constraints for Multi-Rotor Aerial Vehicles," *arXiv preprint arXiv:1911.08183*, 2019.

[14] P. Roque and R. Ventura, "Space CoBot: Modular design of an holonomic aerial robot for indoor microgravity environments," *IEEE International Conference on Intelligent Robots and Systems*, 2016.

[15] T. A. Howell, B. E. Jackson, and Z. Manchester, "ALTRO: A Fast Solver for Constrained Trajectory Optimization," *IEEE International Conference on Intelligent Robots and Systems*, pp. 7674–7679, 2019.

[16] A. B. Younes, D. Mortari Prof., J. D. Turner, and J. L. Junkins Prof., "Attitude error kinematics," *Journal of Guidance, Control, and Dynamics*, vol. 37, no. 1, pp. 330–335, 2014.

[17] B. Houska, H. J. Ferreau, and M. Diehl, "An auto-generated real-time iteration algorithm for nonlinear MPC in the microsecond range," *Automatica*, vol. 47, no. 10, pp. 2279–2285, 2011.

[18] H. J. Ferreau, "An Online Active Set Strategy for Fast Solution of Parametric Quadratic Programs with Applications to Predictive Engine Control," Ph.D. dissertation, Heidelberg University, 2006.

[19] M. Diehl, H. G. Bock, J. P. Schlöder, R. Findeisen, Z. Nagy, and F. Allgöwer, "Real-time optimization and nonlinear model predictive control of processes governed by differential-algebraic equations," *Journal of Process Control*, vol. 12, no. 4, pp. 577–585, 2002.

[20] S. Gros, M. Zanon, R. Quirynen, A. Bemporad, and M. Diehl, "From linear to nonlinear MPC: bridging the gap via the real-time iteration," *International Journal of Control*, no. October, pp. 1–19, 2016.

[21] F. Furrer, M. Burri, M. Achtelik, and R. Siegwart, *RotorSA modular gazebo MAV simulator framework*. Springer International, 2016, vol. 625.