Autonomous Spacecraft Rendezvous using Tube-based Model Predictive Control: Design and Application*

Caroline Specht † and Abhiraj Bishnoi ‡ and Roberto Lampariello §
DLR, German Aerospace Center, Weßling, 82234, Germany
JSC, Jülich Supercomputing Centre, Jülich, 52428, Germany

As the concentration of large space debris increases, how rendezvous maneuvers involving these typically non-cooperative, freely-tumbling bodies are planned and executed is evolving. The rendezvous must be carefully planned, employing up-to-date *in situ* data to identify the inertial and motion parameters of the target body, and executed in a manner which accounts for the remaining uncertainty in these parameters. This paper presents an extension of the TRACE pipeline used in the ROAM/TumbleDock Astrobee experiment campaign, which sequences the target state estimation, motion planning, controller design, and maneuver execution tasks while additionally providing logical loop-back avenues to previous tasks, increasing the chances of a successful maneuver. The pipeline's performance is analyzed in simulation, utilizing: target state estimates generated in a previous activity on a dedicated on-ground testbed; online motion planning, based on non-linear programming and warm-started using a trajectory library generated offline with a novel GPU-based method; and Tube-based Model Predictive Control to robustly track the planned trajectory. Tube-based Model Predictive Control is an actively evolving subject, distributed over multiple publications and various research interests. The necessary theory and considerations for practical implementation of the method are consolidated; its use, features, and limitations in the proposed task are demonstrated.

I. Introduction

Rendezvous and docking maneuvers are a common occurrence on-orbit. However, as the concentration of orbiting bodies which are refuse - spent rocket bodies, exhausted satellites, and other large debris - increases, the manner in which such maneuvers are envisioned is changing. The task is impacted by the fact that these bodies are typically non-cooperative and freely tumbling in an unknown or incompletely characterized manner. Interest in automating these safety-critical maneuvers is increasing. Safe task execution is a highly dynamic and constrained process; and a

*Pre-print of manuscript submitted to the Journal of Guidance, Control, and Dynamics. Submitted on 6 Oct 2022.

†PhD Candidate and Research Scientist, Institute of Robotics and Mechatronics, Department of Autonomy and Teleoperation, email: caroline.specht@dlr.de.

‡Research Engineer, Jülich Supercomputing Centre, Technology Division, email: a.bishnoi@fz-juelich.de.

§Senior Research Scientist, Institute of Robotics and Mechatronics, Department of Autonomy and Teleoperation, email: roberto.lampariello@dlr.de.

well-planned strategy of target observation, motion planning, and rendezvous execution is, as such, necessary.

There is a growing body of work which places these tasks in sequence, with differing scopes and levels of autonomy. Some earlier works, for example [1], tackled the preliminary steps of motion and parameter estimation. Others used assumed knowledge of the target and proposed motion planning [2] or control techniques [3]. Eventually work progressed to propose guidance methodologies for rendezvous to tumbling targets, consider uncertainty sources, integrate robotic systems, and consider more complex planning approaches [4–9].

In [10], initial results for the first known autonomous on-orbit rendezvous with a tumbling target are presented, using a target identification and rendezvous pipeline first presented in [11] called Tumbling Rendezvous via Autonomous Characterization and Execution (TRACE). This TRACE pipeline was developed for the ROAM/TumbleDock experiment campaign which involved the autonomous rendezvous of two Astrobee robots on-board the ISS simulating a larger on-orbit scenario. One robot played the role of a servicer, or "chaser", satellite and the other that of a non-cooperative tumbling target. The pipeline incorporates the observation of the target motion and estimation of its inertial and motion parameters, the planning of the motion of the chaser robot to a predefined point relative to the target robot in the target robot's body frame, or mating point (MP), and its controlled motion to this point into a single-shot process. These processes are introduced sequentially in the following, with major emphasis placed on the last step – controller design.

The first step is the observation of the target satellite and a characterization of its inertial and motion parameters. A good estimate of these quantities, based on data collected in orbit at the time of the maneuver, is important to its success, but some level of uncertainty will inherently remain. In [12], an overview of the state of the art of methods used to accomplish this estimation is articulated and the uncertainty in the evolution of the MP resulting from uncertainty in the estimated quantities is discussed. Results obtained in [12] are used here for this first step of the pipeline.

The next step is to plan the chaser trajectory to the MP based on the parameters estimated in the previous phase. In [13], fuel-optimal trajectories are developed for managing satellite fleets interacting with non-tumbling, cooperative targets using mixed-integer linear programs; collision avoidance and plume impingement constraints are modeled as keep-out zones. A tumbling target modeled as a convexified sphere is considered in [14], and the generation of time- and energy-optimal trajectories using an approach based in the Gauss pseudospectral approach is discussed. In the guidance algorithm proposed in [8], the translational motion is planned using a sequential convex programming procedure, which convexifies the non-convex constraints and rapidly generates solutions through repeated solution of the guidance problem as the maneuver progresses. In [15], fuel-optimal trajectories for the final stages of proximity maneuvers are autonomously generated through the successive solution of a non-linearly, convexly constrained optimal control problem which allows the target to be in any general orbit. In [5], the rendezvous to a non-cooperative tumbling non-convex target is formulated as a nonlinear program (NLP), with a penetration-depth-based collision avoidance constraint, which was solved offline first in a coarse global search followed by a smoothing task. The motion planner presented in this paper extends this NLP and a look-up table (LUT) is used to warm start its online execution. A GPU is used in this

implementation to accelerate the offline LUT computation, specifically with respect to collision detection computations.

While the rasterization capabilities of GPUs have long been used for collision detection [16], their use in optimization-based motion planning has received limited attention. In [17], a GPU is used to interleave optimization-based planning with execution, utilizing a pre-computed Euclidean Distance Transform on a discrete, voxel-based grid for collision detection. The potential of using a GPU to optimize performance of optimization-based motion planning for humanoid robots using a distance-based collision avoidance approach is explored in [18]. By contrast, this work uses oriented bounding polytopes and custom intersection tests to analyze collision detection in the generation of the LUT.

The final part of the pipeline is the controlled execution of the maneuver. As control tasks increase in complexity and autonomy requirements intensify, online compatible control methods for highly constrained systems and environments have risen in prominence. In recent years, interest in these algorithmic control (AC) methods has increased in the aerospace guidance and control community [19]. Notably, on-orbit robotics tasks have increasingly considered on-board, online control methods, [6, 11, 20–24] being a small sample. The increasing availability and capability of on-board computation permits the application of these methods on aerospace systems for safe and reliable control action [19, 25].

Model predictive control (MPC), a subset of AC methods, foregoes the mathematical recipe of a control law for a control action determined at each time step in an iterative process [25]. MPC methods allow for uncomplicated incorporation of constraints and provide strong theoretical results pertaining to stability and feasibility[26], even under a certain degree of modelling error or instrumentation uncertainty [27, 28]. This stability has been studied in [27] and the resources within. However, the presence of uncertainty in the system model or its operational environment results in a loss of the guarantees of controller stability [27], and arbitrary perturbations may destabilize the system. Robust MPC methods build on nominal MPC to account for disturbances and modeling errors and to restore these guarantees. Open-loop robust frameworks are generally overly conservative and closed-loop predictions often result in controllers with a relatively high computational complexity [26]. Tube-based model predictive control (TBMPC) is a robustified methodology which aims to reduce online complexity, making it suitable for real-time applications.

TBMPC was formalized first for the linear regulation problem in [29]; introducing robustness to the MPC of a perturbed system by solving the nominal MPC problem whilst additionally constraining the difference between the nominal and perturbed systems. TBMPC is therefore able to provide robust control with a minor increase in online computational expense. This quality makes TBMPC attractive for many applications, and many variations, including application to the tracking problem, have subsequently become popular in the control community. To achieve this, the method is built on the theory of robust positively invariant (RPI) sets to develop a tube within which the system is restricted to evolve and to determine a suitable restriction of the system constraints. By solving the nominal problem under suitably tightened constraints and centering the tube on the nominal trajectory, the nominal system should be controllable in such a way that constraints placed on both the nominal and the perturbed systems are satisfied at all times. RPI sets are themselves an active area of research. The practical and efficient calculation of the RPI sets can

be difficult, complicating the implementation of the control method. One contribution of this paper aims to make the theory more accessible and transparent.

In recent years, TBMPC has been used to study various aspects of robotic satellite rendezvous maneuvers. Navigation and thruster timing uncertainties are investigated in [30] and [31], respectively. In [6], uncertainty in the state of a non-cooperative tumbling target and the resulting uncertainty in a collision-free rendezvous trajectory are considered. Finally, [11] incorporates TBMPC for tracking with additive disturbances as the control method of choice into the TRACE Pipeline, while [32] propeses a method to more efficiently handle the uncertainty in this formulation.

This paper complements and extends the works presented in [10], [11], and [5] and presents multiple contributions. This work presents a detailed description and analysis of the motion planner presented in [11]. This motion planner noticeably improves the NLP presented in [5] – improving efficiency of the optimization problem formulation, including new plume and pointing constraints, generating trajectories online, and providing a warm start to the optimization problem from an offline-generated LUT. The theory and practical considerations for the implementation of tube-based model predictive control for tracking, the robust control method of choice for the TRACE pipeline, which is spread across multiple publications and various related but independent research interests, are consolidated and exposited. The modularity of the phases of the TRACE pipeline is demonstrated and exploited (a) to substitute the target parameter estimation method with that presented in [12], which provides additional information pertaining to the uncertainty in the estimated inertial and motion parameters of the target, and (b) to form a "closed-loop" pipeline. The TRACE pipeline is in fact extended to include logical loop-back avenues to previous steps. In the open-loop implementation, a failure at any point would result in a maneuver failure. Through the addition of controlled retreats, the extended pipeline mitigates points of failure by using knowledge gained as a result of the pipeline failure. Most notably, failures resulting from incompatibility between the derived controller and designed motion plan can be recuperated. These failures occur when the derived controller parameters become overly conservative through inappropriate tightening of the motion constraints and the designed motion plan becomes infeasible. This process leads to automated *GO/ABORT* conditions, which will only permit eventual ingress if the maneuver is expected to succeed, based on the success of each preceding phase. This paper provides results from the pipeline in simulation and a numerical analysis of its composite parts.

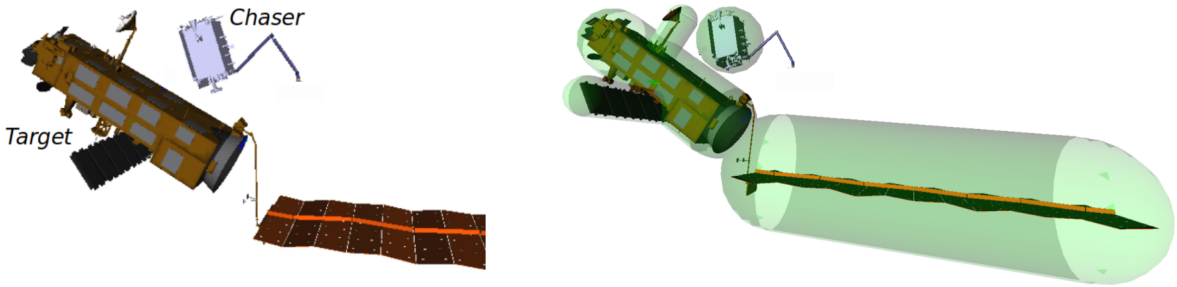
The remainder of this paper is structured as follows: Section II describes the satellite rendezvous task. Sec. III presents a unified exposition of the theory for tube-based model predictive control for tracking. Sec. IV discusses the novel iterative pipeline to be used in the approach maneuver of a spacecraft to a free-tumbling target. The results of the implementation of this pipeline in simulation and a numerical analysis of its composite parts are presented and discussed in Sec. V. Some concluding remarks are given in Sec. VI.

II. Rendezvous task

The rendezvous task under consideration is modeled on the robotic capture and de-orbit of Envisat as proposed in the planned ESA mission e.Deorbit [33]. The "target", Envisat, is a large Earth observation satellite (launch mass approximately 8200 kg), which is no longer in service and is freely-tumbling and non-cooperative. In this rendezvous task, a chaser satellite approaches the target to a predefined final relative position, referred to as the Mating Point (MP), defined in the body frame of the target and considered to be suitable for robotic capture, as demonstrated in Fig. 1.

The approach maneuver, sketched in Fig. 2, is considered in three spatial dimensions. The orientation dynamics of the chaser are assumed to be suitably controlled such that the line of sight with the target is maintained. The pair of satellites are considered to be traveling in a low Earth orbit. The target body frame $\{O^t, \boldsymbol{x}^t\}$ and the orbital frame $\{O^o, \boldsymbol{x}^o\}$ are centered on the center of mass (CoM) of the target. The orbital frame is assumed to be inertial and, as such, does not rotate. The MP is described in the target body frame by ${}^t r_{MP}$. The chaser is modeled as a point mass with a spherical body geometry and body frame traveling in the same orbit as the target, phased a known distance from it at the commence of the maneuver. The chaser approaches the MP along the dotted path in Fig. 2; the current position relative to the target CoM is given by ${}^o x_p(t)$.

Several aspects of this maneuver make it particularly challenging. The focus here is on those relating to feasibility of the motion constraints. While the motion planner provides certain guarantees of feasibility and safety on the trajectories that it produces, these trajectories have been generated based on a prediction of the target motion, which are in turn based on the estimates of the target inertial and motion parameters [12]. In the e.Deorbit scenario, the target Envisat is taken to be tumbling at a rate of $|{}^o\boldsymbol{\omega}_t| \leq 5$ deg/s. However, substantial uncertainty is known to be present in the target dynamics model, particularly in the inertia ${}^t\mathrm{I}_t$ and in the angular velocity ${}^o\boldsymbol{\omega}_t$ [12, 33]. Identification of these



(a) (b)

Fig. 1 (a) The tumbling target satellite Envisat and a chaser spacecraft with robot arm in synchronized flight at the pre-selected MP, suitable for grasping. (b) The hull geometry used for collision avoidance, shown with Envisat's appendages and full span of the solar panel.

parameters, based on up-to-date *in situ* data, can be conducted to reduce, but not entirely eliminate, this uncertainty, giving rise to motion prediction errors. The result of this is that the controlled trajectory in the trajectory tracking phase must depart from the planned nominal reference trajectory, calling into question the feasibility of the controlled trajectory in light of the motion constraints. The goal of the robust controller is to extend the feasibility of the planned

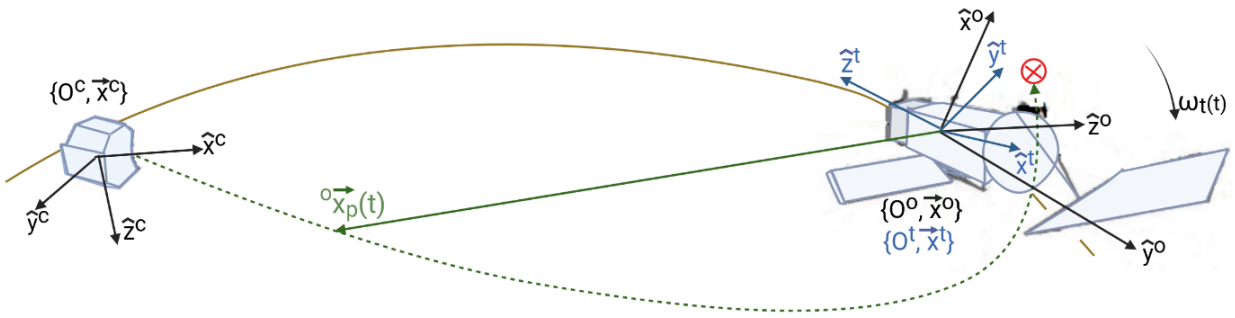


Fig. 2 Chaser, centered on $\{O^c, \boldsymbol{x}^c\}$, approaching the target along the dotted path to the MP $\bigotimes$. The target is centered on the orbital $\{O^o, \boldsymbol{x}^o\}$ and body frames $\{O^t, \boldsymbol{x}^t\}$ and traveling on an orbit indicated by the solid arc.

trajectory provided by the motion planner to a region around this trajectory, as a function of a given bounded uncertainty. It is required that the chaser meets the MP at the end of the maneuver and that it tracks the relative motion between the chaser and target, in a manner specified by the motion planner, as seen in the target body frame. This will allow the collision avoidance properties of the nominal trajectory to be maintained.

A. Target dynamics

The target is seen as a free-tumbling rigid body with uncertain inertial properties. The rotational motion of the target can be described using the Euler equations of motion. The position ${}^o\boldsymbol{r}_t = [{}^o r_{tx}, {}^o r_{ty}, {}^o r_{tz}]^\top$, orientation quaternion $\boldsymbol{q}_t = [q_{t\theta}, q_{tx}, q_{ty}, q_{tz}]^\top$, and angular velocity ${}^o\boldsymbol{\omega}_t = [{}^o\omega_{tx}, {}^o\omega_{ty}, {}^o\omega_{tz}]^\top$ are defined to make up the state of the target, as seen in the orbital frame. The rotational equations of motion of the target can be defined using

$$ {}^o\dot{\boldsymbol{\omega}}_t = -\mathrm{I}_t^{-1}[\boldsymbol{\omega}_t \times \mathrm{I}_t\boldsymbol{\omega}_t + \boldsymbol{\tau}_t] \tag{1} $$

$$ \dot{\boldsymbol{q}}_t = \frac{1}{2}\mathrm{H}(\boldsymbol{\omega_t})\boldsymbol{q}_t, \tag{2} $$

where $\boldsymbol{\tau}_t$ indicates the actuating torques on the target, m_t is its mass, and H relates the quaternion and angular velocity.

B. Chaser dynamics

The Euler-Hill system dynamics will be attributed to the translational motion of the chaser [34]

$$ {}^o\ddot{x}_c = 2T_{orb}\,{}^o\dot{z}_c + u_x \tag{3} $$

$$ {}^o\ddot{y}_c = -T_{orb}^2\,{}^o y_c + u_y \tag{4} $$

$$ {}^o\ddot{z}_c = 3T_{orb}^2\,{}^o z_c - 2T_{orb}\,{}^o\dot{x}_c + u_z, \tag{5} $$

where T_{orb} represents the orbital period, ${}^o\boldsymbol{x}_c = [{}^o x_c, {}^o y_c, {}^o z_c, {}^o\dot{x}_c, {}^o\dot{y}_c, {}^o\dot{z}_c]^\top$, $\boldsymbol{u}_c = [u_x, u_y, u_z]$ indicates the chaser actuation, and $\boldsymbol{q}_c$ and $\boldsymbol{\omega}_c$ for the chaser are comprised similarly to those for the target. The nominal state, and likewise the real state, is composed of the chaser position and velocity. Therefore, the chaser state equation is given by

$$ {}^o\dot{\boldsymbol{x}}_c = \mathrm{A_c}\,{}^o\boldsymbol{x}_c + \mathrm{B_c}\boldsymbol{u}_c, \tag{6} $$

which defines the state and actuation matrices A_c and B_c that are derived from (3)-(5). The rotational dynamics of the chaser are likewise indicated by

$$^{o}\dot{\boldsymbol{\omega}}_c = -\mathrm{I}_c^{-1}[\boldsymbol{\omega}_c \times \mathrm{I}_c\boldsymbol{\omega}_c + \boldsymbol{\tau}_c] \tag{7}$$

where I_c, $\boldsymbol{F}_c$ and $\boldsymbol{\tau}_c$ respectively indicate the inertia, actuating forces, and torques on the chaser. The mass of the chaser is indicated by m_c.

C. Uncertainty description

When uncertainty in the motion of the target exists, the relationship between the target body and orbital frames will not be as expected. As a result, the relative state of the chaser in the orbital frame must alter to maintain the desired properties designed into a rendezvous trajectory. In this work, uncertainty in the target motion is considered to arise from imperfect knowledge of the target inertia and/or its angular velocity

$$\mathrm{I}_t = \mathrm{I}_{t,N} + \delta\mathrm{I}_t \tag{8}$$

$$\boldsymbol{\omega}_t(t) = \boldsymbol{\omega}_{t,N}(t) + \delta\boldsymbol{\omega}_t(t), \tag{9}$$

where the subscript N indicates the nominal quantity. A close relationship [12] exists between $\delta\mathbf{I}_t \neq 0$ and $^{o}\delta\boldsymbol{\omega}_t(t) \neq 0$, with similar effect when either is true: a perturbation arises in the orientation of the target body in the orbital frame

$$\boldsymbol{q}_t(t) = \boldsymbol{q}_{t,N}(t) + \delta\boldsymbol{q}_t(t). \tag{10}$$

As the nominal trajectory should be maintained in the target body frame, the real trajectory of the chaser in the orbital frame will alter as a function of $\delta\boldsymbol{q}_t(t)$. The update to the chaser trajectory is highly nonlinear: Let the nominal position and velocity references be given in the orbital frame as $^{o}\boldsymbol{z}_p(t)$ and $^{o}\boldsymbol{z}_v(t)$, respectively. The relative nominal position in the target body frame is given by

$$^{t}\boldsymbol{z}_p(t) \;=\; \mathrm{R}^{to}(\boldsymbol{q}_{t,N}(t, \boldsymbol{\omega}_{t,N}(t)))\;\; ^{o}\boldsymbol{z}_p(t), \tag{11}$$

where $\mathrm{R}^{to}(\bullet)$ is the rotation matrix from the orbital to the target body frame. To preserve the nominal relative motion in the target body frame, the orbital frame motion should follow

$$^{o}\boldsymbol{z}'_p(t) \;=\; \mathrm{R}^{ot}(\boldsymbol{q}_t(t, \boldsymbol{\omega}_t(t)))\;\; ^{t}\boldsymbol{z}_p(t), \tag{12}$$

where $'$ indicates the perturbed state. The same series of transforms can be made to obtain $^{o}\boldsymbol{z}'_v(t)$ from $^{o}\boldsymbol{z}_v(t)$.

III. Tube-based Model Predictive Control for tracking: theory and design

The theory and algorithms used in the design of a tube-based model predictive controller for tracking are discussed in this section. These ideas will then be applied to the rendezvous task described in the previous section.

A. Nominal system definition and control problem

The nominal system to be controlled is defined here in terms of a linear time invariant (LTI), constrained, discrete-time model-based system. This system has a state represented by a vector $\boldsymbol{z}_k$ at time step k of dimension n equal to the

number of system states. Similarly, let the control input be given by a vector $\boldsymbol{v}_k$, of dimension m equal to the number of inputs to the system. The open-loop nominal system is then given by

$$z_{k+1} = \mathrm{A}z_k + \mathrm{B}\boldsymbol{v}_k \tag{13}$$

where the successor state of the system is z_{k+1} and the state and control matrices are respectively represented by the matrices $\mathrm{A} \in \mathbb{R}^{nxn}$ and $\mathrm{B} \in \mathbb{R}^{nxm}$. It can be assumed that the system $(\mathrm{A}, \mathrm{B}) \in \mathbb{R}^{nxn} \times \mathbb{R}^{nxm}$ is controllable.

As MPC is based in optimal control methods, it can inherently handle a constrained control problem [26]. State and control constraints can therefore easily be incorporated, respectively

$$z \in \mathbb{Z} \subset \mathbb{R}^n \tag{14}$$

$$\boldsymbol{v} \in \mathbb{V} \subset \mathbb{R}^m. \tag{15}$$

These are the nominal system constraints. All constraints considered in this work are polytopic.

The optimization task of the nominal model predictive controller possesses an objective function which is generally of the form

$$J(\mathbf{z}, \mathbf{v}) = \sum_{i=0}^{N-1} \|z_i - \bar{z}_i\|_{\mathrm{Q}}^2 + \|\boldsymbol{v}_i - \bar{\boldsymbol{v}}_i\|_{\mathrm{R}}^2 + J_f\,(z_N, \bar{z}_N). \tag{16}$$

Here, $\bar{z}$ and $\bar{\boldsymbol{v}}$ are the reference state and control values, respectively, the length of the prediction horizon is given by N, and $\mathbf{z} = [z_0, ..., z_N]$ and $\mathbf{v} = [\boldsymbol{v}_0, ..., \boldsymbol{v}_{N-1}]$ indicate the series of state and input on the prediction horizon. The first two terms of the summation are called the stage cost and $J_f\,(z_N, \bar{z}_N)$ is referred to as the terminal cost. The stage cost functions $\|z_i - \bar{z}_i\|_{\mathrm{Q}}^2 = (z_i - \bar{z}_i)^\top \mathrm{Q}(z_i - \bar{z}_i)$ and $\|\boldsymbol{v}_i - \bar{\boldsymbol{v}}_i\|_{\mathrm{R}}^2 = (\boldsymbol{v}_i - \bar{\boldsymbol{v}}_i)^\top \mathrm{R}(\boldsymbol{v}_i - \bar{\boldsymbol{v}}_i)$ are positive definite, squared, weighted Euclidean norms. In the tracking case, $\bar{z}_i = z_{ref,i}$ and $\bar{\boldsymbol{v}}_i = \boldsymbol{v}_{ref,i}$, corresponding to planned reference values at each time step. The cost function considers at each step on the prediction horizon the weighted contribution of the deviation of the predicted states and controls from the reference values. The goal of the optimization is then to bring the predicted values to the reference values. The terminal cost $J_f\,(z_N, \bar{z}_N)$ is also a positive definite function and is often of the form $\|z_N - \bar{z}_N\|_{\mathrm{P}}^2 = (z_N - \bar{z}_N)^\top \mathrm{P}(z_N - \bar{z}_N)$, where P is the weighting matrix of the terminal constraint, and $\bar{z}_N$ is the state reference at time N. In conventional MPC, P is often selected to be the solution to the Riccati equation.

The optimization part of a general nominal model predictive controller is then of the form

$$\min_{\mathbf{z},\mathbf{v}} J(\mathbf{z}, \mathbf{v}) \tag{17a}$$

$$s.t.\ z_i \in \mathbb{Z}, i = 0; ..., N \tag{17b}$$

$$\boldsymbol{v}_j \in \mathbb{V}, j = 0, ..., N-1. \tag{17c}$$

The common tactic in MPC is to determine the series $\mathbf{v}$ of optimal control actions $\boldsymbol{v}_j$ for the next N steps using the optimization task in (17a)-(17c), but to only apply the first control action to the system.

B. Perturbed System Definition

In this paper, the focus is on bounded, additive disturbances $\boldsymbol{w}_k$. It is not necessary to know the exact value of $\mathbf{w} = [\boldsymbol{w}_{t0}, ..., \boldsymbol{w}_k, ..., \boldsymbol{w}_{tf}]$ at the time that the controller is designed, but it is necessary to know the bounded set $\mathbb{W}$ to

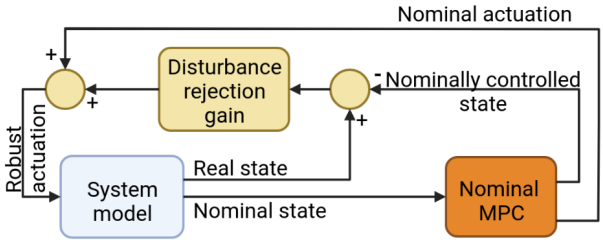


Fig. 3 A block diagram representation of TBMPC

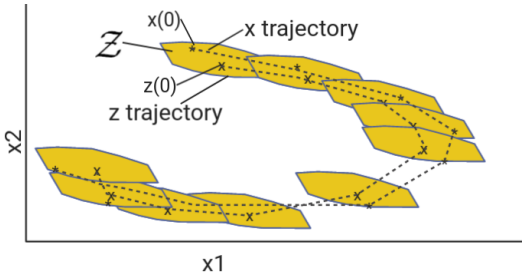


Fig. 4 A sample evolution of a system, encapsulated by a tube

which the uncertainty will belong [26]. In this paper, polytopic sets $\mathbb{W}$ are considered, and

$$\boldsymbol{w} \in \mathbb{W} \subset \mathbb{R}^n. \tag{18}$$

Choosing $\mathbb{W}$ to be polytopic can result in a lower computational complexity of the controller design process.

Incorporating such an uncertainty into the system model, gives the open-loop perturbed system

$$\boldsymbol{x}_{k+1} = \mathrm{A}\boldsymbol{x}_k + \mathrm{B}\boldsymbol{u}_k + \boldsymbol{w}_k, \tag{19}$$

where $\boldsymbol{x}$ denotes the perturbed system state and $\boldsymbol{u}$ is the perturbed system input. The state disturbance $\boldsymbol{w}$ directly affects the evolution of the state and can include external disturbances to the system, parameter uncertainty, and uncontrolled dynamics. The perturbed system has state and control constraints which can be described by

$$\boldsymbol{x} \in \mathbb{X} \subset \mathbb{R}^n \tag{20}$$

$$\boldsymbol{u} \in \mathbb{U} \subset \mathbb{R}^m. \tag{21}$$

C. Tube-based Model Predictive Control

Several approaches to TBMPC have been developed for tracking and regulation control problems in recent years, but they all have the same fundamental parts. As illustrated in Fig. 3, TBMPC is composed of a nominal model predictive controller with slightly modified constraints and cost function and a feedback term. In TBMPC for tracking, the purpose of the nominal MPC is to control the nominal system to track a reference trajectory, using classical MPC methods. The feedback term then adds the necessary additional control action to the nominal controller output, to account for the perturbations. The common form of the LTI feedback control action is given by

$$\boldsymbol{u}_k = \boldsymbol{v}_k + \mathrm{K}_{dr}(\boldsymbol{x}_k - z_k), \tag{22}$$

where $\mathrm{K_{dr}} \in \mathbb{R}^{m\times n}$ is a disturbance rejection gain. The second term of (22) is therefore a disturbance rejection controller minimizing the deviation of the real system $\boldsymbol{x}_k$ from the nominal system z_k.

The solution of an MPC task is dependent a given realization of the generic uncertainty. TBMPC is motivated by knowledge that both the open- and closed-loop formulations of the robust control problem in the presence of uncertainty generate a tube of trajectories [28]. Each trajectory in this bundle corresponds to a particular realization of the uncertainty. The idea of such a tube is represented in Fig. 4. The tube is made up of a series of n dimensional sets $\mathcal{Z}$, depicted in yellow. The dashed line starting at the initial state of the system $z(0)$ and connecting the "x" markers at

the center of each $\mathcal{Z}$ indicates the nominal trajectory, or the evolution that the system would follow if no uncertainty is present. The edges of the sets $\mathcal{Z}$ indicate the bounds within which the bundle of trajectories must lie. The dashed line starting at $\boldsymbol{x}(0)$ and connecting the asterisks represents the trajectory corresponding to some realization of the system uncertainty. The type of set used to compile the tube is called robust positively invariant (RPI). RPI sets play a special role in robust MPC methods[29, 35]. Pursuant to the goals of a presentation of a unified theory, a special note is made of RPI use and determination in TBMPC.

In TBMPC, the term RPI is used consistently to mean disturbance invariant, implying that the set is invariant to the realization of the bounded disturbance. This means that if a system of the form in (19) is considered, then a set O is referred to as RPI for the system if for every initial state $x_0 \in O$ and for all $w \in \mathbb{W}$, the solution $x_k \in O, k > 0$.

The tube is composed of a particular type of RPI set, called the minimal robust positively invariant set (mRPI). The mRPI is an RPI set that is contained in every other closed RPI set of the system. Effectively, it is the smallest RPI set for the system, and has the special meaning that if a given current state $\boldsymbol{x}_k$ is located within the mRPI at time step k, then, in the presence of any realization of $\boldsymbol{w} \in \mathbb{W}$, the successor state $\boldsymbol{x}_{k+1}$ will also reside within mRPI. The exact mRPI, $\mathcal{F}_\infty$, can only be determined under certain circumstances, so an approximation $\mathcal{Z}$ is used in practice, such that $\mathcal{Z} \approx \mathcal{F}_\infty \subseteq O$. One instance of this set $\mathcal{Z}$ is superimposed on the nominal trajectory at each via point. The size and shape of this set is in principle derived from the system dynamics and the definition of the set $\mathbb{W}$. Furthermore, the mRPI influences the necessary adjustment from the real system constraints $\mathbb{X}$ and $\mathbb{U}$ to the nominal system constraints $\mathbb{Z}$ and $\mathbb{V}$ such that both systems satisfy their respective constraints at all time and for any disturbance realization within $\mathbb{W}$.

The center of the tube will be coincident with the nominal (unperturbed) system response. The boundary of the tube then encloses all trajectories which satisfy all of the constraints imposed on the optimization procedure, which will be discussed in subsequent sections. Through suitable design of this tube, satisfaction of the constraints can be guaranteed for every realization of the disturbance yielding a trajectory residing within the tube. The entire tube need not be considered at each prediction step. As a result of the design process, which will be discussed in the next section, there is a definition of the outer boundary of the tube at each time step. By considering the evolution of the system in a step-wise fashion, the boundary of the entire tube will have been accounted for. [28]

1. The robust control strategy

The control strategy is two-fold: an MPC element for the control of the nominal system and an ancillary linear controller acting on the discrepancy between the actual state $\boldsymbol{x}$ and nominal state z. Let this error be indicated by $\boldsymbol{e}_k = \boldsymbol{x}_k - z_k$. The control law can be re-written in the form

$$\boldsymbol{u}_k = \boldsymbol{v}_k + \mathrm{K_{dr}}\boldsymbol{e}_k, \tag{23}$$

such that

$$\mathrm{A_K} = \mathrm{A} + \mathrm{BK_{dr}} \tag{24}$$

is Hurwitz stable for the closed-loop system [28, 29]. Now, for the perturbed, closed-loop system

$$\boldsymbol{x}_{k+1} = \mathrm{A_K}\boldsymbol{x}_k + \boldsymbol{w}_k, \tag{25}$$

let $O \in \mathbb{R}^n$ be an RPI set such that

$$\mathrm{A_K}\mathcal{Z} \bigoplus \mathbb{W} \subseteq \mathcal{Z} \subseteq O \tag{26}$$

is satisfied, where $\bigoplus$ indicates the Minkowski set addition. Then, if the current real system state is such that

$$\boldsymbol{x}_k \in \{z_k\} \bigoplus \mathcal{Z} \tag{27}$$

and the control action in (23) is applied, it follows that the successor state

$$\boldsymbol{x}_{k+1} \in \{z_{k+1}\} \bigoplus \mathcal{Z} \tag{28}$$

for all admissible disturbance sequences [29].

To ensure that the constraints $\boldsymbol{x}_k \in \mathbb{X}$ and $\boldsymbol{u}_k \in \mathbb{U}$ are satisfied for all time by the perturbed system, the constraints of the nominal problem are defined in terms of the perturbed system and the size and shape of $\mathcal{Z}$. Therefore $\mathbb{Z}$ and $\mathbb{V}$ are defined using a constraint tightening procedure, such that

$$\mathbb{Z} = \mathbb{X} \ominus \mathcal{Z} \tag{29}$$

$$\mathbb{V} = \mathbb{U} \ominus \mathrm{K}_{dr}\mathcal{Z}, \tag{30}$$

where $\ominus$ indicates the Pontryagin set difference [28, 29].

D. Design of tube-based model predictive controller for tracking

The TBMPC design process results in a slightly modified nominal model predictive controller accompanied by a cluster of additional parameters. The evolution of Fig. 5 is followed to describe each step in detail.

1. System definition and path planning

The first step is to define the nominal (13) and real (19) system model and dynamics with constraints (20) and (21) and bounded uncertainty set (18). The constraints for the nominal system will be designed using a constraint tightening procedure in a later design step to ensure that the real system always satisfies (20) and (21) while the nominal system satisfies (14) and (15). There are many ways in which the uncertainty set can be determined, depending on the system that is to be controlled. Methods commonly used include simulating the system with a desired parameter perturbed in a Monte Carlo search [6] or the numerical approximation of the uncertainty in some parameters can be made [30]. At this point, a reference trajectory should also be obtained using some motion planning measure (see Sec.IV.C).

2. Disturbance Rejection Gain

Before proceeding, the disturbance rejection gain must be calculated. This parameter only needs to be determined once in TBMPC design and is fundamentally important to the robustness of the controller.

Consider for the synthesis of this gain that $\boldsymbol{u} = \mathrm{K}_{dr}\boldsymbol{x}$. The simplest manner in which to choose K_{dr} is to "manually" select it to be equal to the LQR gain of the nominal system. The appropriateness of this gain does vary, and could theoretically be tuned near to the value achieved by the methods detailed in the following through appropriate selection

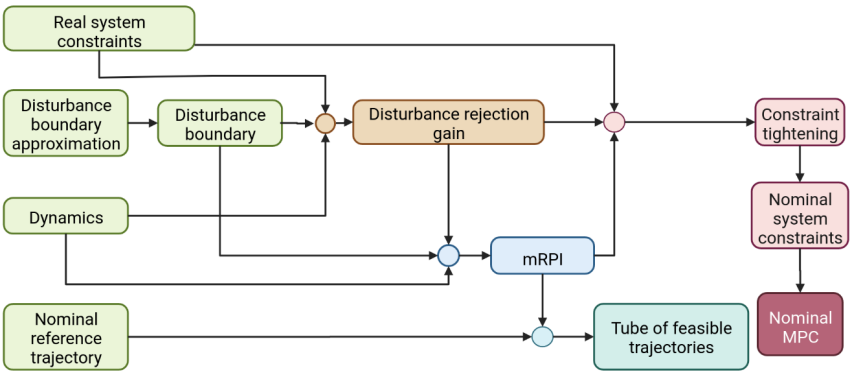


Fig. 5 A flow chart representation of the tube-based model predictive control design.

of the weights Q and R. However, this limits the choice of these weighting matrices, which are often selected in accordance with the tolerable deviation from the reference, the constraints or the robustness of the resulting controller are not explicitly considered, and the empirical method turns to trial and error until the associated response and mRPI are subjectively satisfactory. To optimize K_{dr}, it can be designed using a disturbance rejection criterion, as described in [36]. This criterion will help to ensure that the mRPI set $\mathcal{Z}$ exists, will be of a minimal size, and will be of a size and shape such that the tightened constraint set will be non-empty. This criterion also ensures a larger domain of attraction.

This portion of the design task can be briefly described as the minimization of an ellipsoid

$$\mathcal{E}(\mathrm{P}_e, 1) = \{\boldsymbol{x} \in \mathbb{R}^n | \boldsymbol{x}^\top \mathrm{P}_e \boldsymbol{x} \leq \mathbf{1}\}, \tag{31}$$

where P_e is a positive definite matrix which uniquely defines the ellipsoid. To satisfy the robustness criterion, the ellipsoidal set must be RPI. To this end, the K_{dr} optimization problem will be formulated as a Linear Matrix Inequality (LMI) subject to a number of constraints, which account for the state, control, and dynamic requirements of the system.

The RPI requirement is formulated as the condition

$$(\boldsymbol{x}_{k+1})^\top \mathrm{P}_e(\boldsymbol{x}_{k+1}) \leq \mathbf{1}, \forall \boldsymbol{x}_k \in \mathcal{E}(\mathrm{P}_e, 1), \boldsymbol{x}_{k+1} = \mathrm{A}_K \boldsymbol{x}_k + \boldsymbol{w}_k, \forall \boldsymbol{w}_k \in \mathbb{W}. \tag{32}$$

The minimization of the volume of the ellipsoid $\mathcal{E}(\mathrm{P}_e, 1)$ can be reformulated to explicitly consider this condition using the so-called S-procedure. This procedure is used when it is desired to combine several quadratic inequalities into one single inequality. The result of the application of this procedure is the inequality

$$(\mathrm{A}_K \boldsymbol{x} + \boldsymbol{w})^\top \mathrm{P}_e (\mathrm{A}_K \boldsymbol{x} + \boldsymbol{w}) + \lambda(1 - \boldsymbol{x}^\top \mathrm{P}_e \boldsymbol{x}) \leq \mathbf{1}, \quad \forall \boldsymbol{w} \in vert(\mathbb{W}), \tag{33}$$

where $vert(\mathbb{W})$ indicates the vertices of the set $\mathbb{W}$. This inequality is satisfied if there exists a scalar $\lambda \geq 0$. Applying the Schur complement to this inequality yields the target LMI:

$$\begin{bmatrix} \lambda \mathrm{P}_e - (\mathrm{A} + \mathrm{B}\mathrm{K}_{dr})^\top \mathrm{P}_e (\mathrm{A} + \mathrm{B}\mathrm{K}_{dr}) & -(\mathrm{A} + \mathrm{B}\mathrm{K}_{dr})^\top \mathrm{P}_e \boldsymbol{w} \\ -\boldsymbol{w}^\top \mathrm{P}_e (\mathrm{A} + \mathrm{B}\mathrm{K}_{dr}) & 1 - \lambda - \boldsymbol{w}^\top \mathrm{P}_e \boldsymbol{w} \end{bmatrix} > 0, \quad \forall \boldsymbol{w} \in vert(\mathbb{W}), \ \lambda \geq 0. \tag{34}$$

In this step, A_K has be expanded to make clear the presence of the targeted parameter K_{dr}.

Instituting the variable changes

$$\mathrm{W} = \gamma \mathrm{P}_e^{-1} \tag{35}$$

$$\mathrm{Y} = \mathrm{K}_{dr}\mathrm{W}, \tag{36}$$

the LMI becomes

$$\begin{bmatrix} \lambda \mathrm{W} & * & * \\ 0 & 1-\lambda & * \\ \mathrm{AW} + \mathrm{BY} & \boldsymbol{w} & \mathrm{W} \end{bmatrix} > 0, \ \ \forall \boldsymbol{w} \in vert(\mathbb{W}). \tag{37}$$

This constraint is symmetric about the diagonal, and the asterisks indicate symmetric elements. This variable change results in an LMI which is jointly convex in W and Y.

Now the constraints required to make the ellipsoidal set constraint admissible and robustly positively invariant can be itemized. Recall that all state and actuation constraints and the disturbance set are polytopic, and are written in the form

$$\mathbb{X} = \{\boldsymbol{x} \in \mathbb{R}^n | \mathrm{A}_x \boldsymbol{x} \leq \boldsymbol{b}_x\} \tag{38}$$

$$\mathbb{U} = \{\boldsymbol{u} \in \mathbb{R}^m | \mathrm{A}_u \boldsymbol{u} \leq \boldsymbol{b}_u\} \tag{39}$$

$$\mathbb{W} = \{\boldsymbol{w} \in \mathbb{R}^n | \mathrm{A}_w \boldsymbol{w} \leq \boldsymbol{b}_w\}, \tag{40}$$

or, equivalently, in their normalized H-representation

$$\mathbb{X} = \{\boldsymbol{x} \in \mathbb{R}^n | |\boldsymbol{h}_i^\top x| \leq 1, i = 1, ..., n_{rx}\} \tag{41}$$

$$\mathbb{U} = \{\boldsymbol{u} \in \mathbb{R}^m | |\boldsymbol{l}_j^\top u| \leq 1, j = 1, ..., n_{ru}\} \tag{42}$$

$$\mathbb{W} = \{\boldsymbol{w} \in \mathbb{R}^n | |\boldsymbol{g}_i^\top w| \leq 1, i = 1, ..., n_{rx}\} \tag{43}$$

where n_{rx} and n_{ru} are the number of rows in their corresponding matrices, and $\boldsymbol{h}_i$, $\boldsymbol{l}_j$, and $\boldsymbol{g}_i$ are the rows of the matrices derived from the normalization of A_x, A_u, and A_w with respect to $\boldsymbol{b}_x$, $\boldsymbol{b}_u$, and $\boldsymbol{b}_w$, as appropriate. The state and actuation constraints will also be reformulated as LMIs, so that they can be applied to the optimization problem.

The actuation constraint can be slightly reformulated to consider the driving control to be

$$\mathbb{U} = \{\boldsymbol{u} \in \mathbb{R}^m | |\boldsymbol{l}_j^\top \mathrm{K}_{dr} \boldsymbol{x}| \leq \rho_j, j = 1, ..., n_{ru}\}.$$

This reformulation indicates that for all $\boldsymbol{x} \in \mathcal{E}(\mathrm{P}_e, 1)$, $|\boldsymbol{l}_j^\top \mathrm{K}_{dr} \boldsymbol{x}| \leq \rho_j$ for all rows of $\boldsymbol{l}$ and $\rho_j \in (0, 1]$. The parameter ρ_j has the purpose of relaxing the actuation constraint in the determination of K_{dr} so that the tightened set of admissible control inputs $\mathbb{V}$ is not empty. The method description in [36] does not specify how to select ρ_j, though it is shown that the selection of a smaller value ρ means that the robust control input will be subject to tighter constraints and the set $\mathbb{V}$ will be larger, and larger values of ρ results in smaller $\mathbb{V}$. The parameter ρ therefore controls the trade-off between good disturbance rejection (ρ large) and good nominal controller performance (ρ small). To apply this constraint to the ellipsoidal set minimization, the Schur complement is applied to the condition

$$\boldsymbol{l}_j^\top \mathrm{K}_{dr} \mathrm{P}_e^{-1} \mathrm{K}_{dr}^\top \boldsymbol{l}_j \leq \rho_j^2, j = 1, ..., n_{ru} \tag{44}$$

and making the same variable changes as in (35) and (36) yields the LMI

$$\begin{bmatrix} \rho_j^2 & * \\ \mathrm{Y}^\top \boldsymbol{l}_j & \mathrm{W} \end{bmatrix} > \mathbf{0}, j = 1, ..., n_{ru}. \tag{45}$$

Finally, a similar procedure for the state constraint LMI formulation transforms $\boldsymbol{x} \in \mathcal{E}(\mathrm{P}_e, 1), |\boldsymbol{h}_i^\top \boldsymbol{x}| \leq 1$ to

$$\begin{bmatrix} 1 & * \\ \mathrm{W}\boldsymbol{h}_i & \mathrm{W} \end{bmatrix} > \mathbf{0}, i = 1, ..., n_{rx}. \tag{46}$$

Now the optimization task can be compiled. The volume of the ellipsoid is proportional to $det(\mathrm{P}_e^{-1}) = det(\mathrm{W})$. This formulation would require the minimization of a concave function and would be intractable for all but the simplest systems. An alternative [36] is to adopt a parameter $\gamma > 0$ such that $\mathcal{E}(\mathrm{P}_e, 1) \subseteq \sqrt{\gamma}\mathbb{X}$ to measure the minimization of the ellipsoid. This scaling of the state constraint must be incorporated into the state constraint LMI, such that

$$\begin{bmatrix} \gamma & * \\ \mathrm{W}\boldsymbol{h}_i & \mathrm{W} \end{bmatrix} > \mathbf{0}, i = 1, ..., n_{rx}. \tag{47}$$

The optimization task then becomes a minimization of γ to a value in the range $(0, 1]$.

Using standard operations of LMIs, the task is fully formulated as a convex optimization problem

$$\min_{\mathrm{Y},\mathrm{W},\gamma} \gamma, \quad s.t.\ (37),\ (45),\ (47).$$

The optimization is then conducted such that each vertex of $\mathbb{W}$ and each half-space of the state and control constraints, $\mathbb{X}$ and $\mathbb{U}$, is considered. For feasible solutions of this optimization task, the ellipsoid is given by $\mathrm{P}_e = \mathrm{W}^{-1}$ and the disturbance rejection gain is given by $\mathrm{K}_{dr} = \mathrm{Y}\mathrm{W}^{-1}$.

In the fashion that this optimization task is assembled, ρ_i and λ can be chosen values and used as tuning parameters. This may require some iteration to find the best balance between good disturbance rejection and good nominal control performance in the case of ρ and that between invariant set contraction and satisfaction of (33). Additionally, it is not immediately clear how to choose some $\lambda \geq 0$. To narrow down the range within which λ should lie, it is possible to place an upper bound on the value. This is done by calculating a factor λ_∞ by substituting the dynamics for the unconstrained infinite horizon controller $\mathrm{A}_K = \mathrm{A}_\infty = \mathrm{A} + \mathrm{B}\mathrm{K}_\infty$ and $\mathrm{P}_e = \mathrm{P}_\infty$ equal to the solution of the Algebraic Riccati equation into (37) and then determining the minimal value of λ_∞ for which (37) is satisfied. Then $0 \leq \lambda \leq \lambda_\infty$, which provides a constrained range within which to find an acceptable value of λ. It should also be noted that the smaller the value of λ, the greater the contraction of the ellipsoidal set.

Alternatively, this LMI could be reformulated as a bilinear matrix inequality (BMI). In this case, λ should be included as another optimization variable. Unfortunately, under this formulation, the problem is not jointly convex in W and λ, and the optimization is more difficult to solve.

3. The mRPI

Now the mRPI set $\mathcal{Z}$ can be approximated based on the derived disturbance rejection gain K_{dr}. The size and shape of this set are additionally influenced by the system dynamics and the bounded disturbance set $\mathbb{W}$.

The boundary of the mRPI set can be given by iterative Minkowski summations

$$\mathcal{F}_s = \bigoplus_{i=0}^{s} \mathrm{A}_K^i \mathbb{W} = \bigoplus_{i=0}^{s} (\mathrm{A} + \mathrm{BK}_{dr})^i \mathbb{W} \tag{48}$$

with $s \in \mathbb{N}_+$ tending to infinity, where $\mathcal{F}_s = \mathcal{F}_\infty$[36, 37]. Unfortunately, the mRPI set is very difficult to determine exactly. In fact, it is only possible to determine $\mathcal{F}_\infty$ using (48) exactly if A_K is nilpotent [38]. It is, however, possible to make an outer-approximation of the set. As such, the estimate $\mathcal{F}_\infty \subseteq \mathcal{F}_s \subseteq \mathcal{F}_\infty \bigoplus \epsilon\mathbb{B}^n$ is used, where ϵ is a chosen error bound and $\mathbb{B}$ is a polytopic norm-ball of the form $\mathbb{B}^n = \{\boldsymbol{b} \in \mathbb{R}^n : \|\boldsymbol{b}\|_\infty \leq 1\}$. For every $\epsilon > 0$ there exists an $s \in \mathbb{N}+$ such that $\mathcal{F}_s$ is an inner-approximation of $\mathcal{F}_\infty$ [37, 38].

Clearly it is necessary to determine an appropriate value for s such that $\mathcal{F}_s \approx \mathcal{F}_\infty$. This is an iterative process in which a series of linear programs is conducted. This series of operations is based in the idea that, if $\mathbb{W}$ contains the origin, there exists a finite $s \in \mathbb{N}_+$ and a scalar $\alpha \in [0, 1)$ which satisfy

$$\mathrm{A}_K^s \mathbb{W} \subseteq \alpha \mathbb{W}. \tag{49}$$

If this is satisfied, then there exists a convex, compact, RPI set for the defined system, given by

$$\mathcal{F}(\alpha, s) = (1 - \alpha)^{-1} \mathcal{F}_s. \tag{50}$$

Considering also the requirement that $\mathcal{F}_s \subseteq \epsilon\mathbb{B}^n$, the equations

$$\alpha(s) = \min \alpha \quad s.t. \ \ (49), \ \alpha \in [0, 1) \tag{51}$$

$$\beta(s) = \min \beta \quad s.t. \ \ \mathcal{F}_s \subseteq \beta\mathbb{B}^n \tag{52}$$

will enable the determination of (α, s) pairs which satisfy (49). It can be noted that the value of $\alpha(s)$ determined in (51) only satisfies its boundaries of $\alpha(s) \in [0, 1)$ when s is sufficiently large. The programs in (51) and (52) are therefore solved for incrementing s until the condition [38]

$$\epsilon \geq \alpha(1 - \alpha)^{-1} \max_{x \in \mathcal{F}_s} \| x \|_\infty \geq \alpha(1 - \alpha)^{-1} \min_{\beta} \{\beta | \mathcal{F}_s \subseteq \beta\mathbb{B}_\infty^n\} \tag{53}$$

is met. It has been shown [37, 38] that when this condition is met, $\mathcal{F}_\infty \subseteq \mathcal{F}_s \subseteq \mathcal{F}_\infty \bigoplus \epsilon\mathbb{B}^n$ is also satisfied. At this point, the estimation

$$\mathcal{Z} = (1 - \alpha(s))^{-1} \mathcal{F}_s \tag{54}$$

can be made, where $\mathcal{F}_s$ is determined using (48).

The preceding yields enough information to make an *a posteriori* determination of the suitability of $\mathcal{F}(\alpha, s)$ as an approximation of $\mathcal{F}_\infty$. However, practically it is more convenient to have an *a priori* determination of how large s, or conversely how small α, should be for $\mathcal{F}(\alpha, s)$ to sufficiently approximate $\mathcal{F}_\infty$. In the following, the *a priori* determination of s and α is described, and this practical interpretation will remain in focus for the remainder of the paper.

As $\mathbb{W}$ is polytopic and is described by a finite set of affine inequalities, it can be described by the finite support function

$$h_{\mathbb{W}}(\boldsymbol{a}) = \sup_{\boldsymbol{w} \in \mathbb{W}} \boldsymbol{a}^\top \boldsymbol{w}, \tag{55}$$

evaluated at $\boldsymbol{a} \in \mathbb{R}^n$.

Based on properties of closed, convex sets [37], it can be shown that

$$\mathrm{A}_K^s \mathbb{W} \subseteq \alpha \mathbb{W} \Leftrightarrow h_{\mathbb{W}}((\mathrm{A}_K^s)^\top \mathrm{A}_{w,i}^\top) \leq \alpha \boldsymbol{b}_{w,i},\ i = 1, ..., n_w. \tag{56}$$

This observation allows for efficient computation of $\alpha(s)$ and it directly follows that

$$\alpha(s) = \max_i \frac{h_{\mathbb{W}}((\mathrm{A}_K^s)^\top \mathrm{A}_{w,i}^\top)}{\boldsymbol{b}_{w,i}},\ i = 1, ..., n_w. \tag{57}$$

This same support function can also be used to set an *a priori* error bound on the approximation $\mathcal{F}(\alpha, s)$, given by

$$M(s) = \min_\beta \{\beta | \mathcal{F}_s \subseteq \beta \mathbb{B}^n\}. \tag{58}$$

Again using properties of closed, convex sets [37], the preceding equation has been shown to be equivalent to [38]

$$M(s) = \max_{j \in \{1,...,n\}} \left\{ \sum_{i=0}^{s-1} h_{\mathbb{W}}((\mathrm{A}_K^i)^\top \boldsymbol{e}_j), \sum_{i=0}^{s-1} h_{\mathbb{W}}(-(\mathrm{A}_K^i)^\top \boldsymbol{e}_j) \right\} \tag{59}$$

where $\boldsymbol{e}_j$ is the j-th standard basis vector of $\mathbb{R}^n$. For a satisfactory (α, s) pair,

$$\mathcal{F}_s \subseteq \alpha^{-1}(1 - \alpha)\beta \mathbb{B}_\infty^n \Leftrightarrow \alpha \leq \frac{\epsilon}{\epsilon + M(s)}. \tag{60}$$

The algorithm then enters an iterative loop to estimate a good value of s. First, the value of s is incremented. Then the value of $\alpha(s)$ is calculated using (57) and compared to the relative error given in the right-hand side of (60). When

$$\alpha(s) > \frac{\epsilon}{\epsilon + M(s)}, \tag{61}$$

it can be concluded that the current value of s is reasonable to proceed with.

The estimation of $\mathcal{Z}$ is now relatively straight forward through (54). The algorithm for the iterative implementation of these steps is provided in Algorithm 1. The reader is directed to [36, 38] for more detail on this process.

4. Tube of trajectories

Finding the boundaries of the tube for the full length of the trajectory is then a simple matter of an affine mapping of the mRPI set along the reference trajectory. As, practically, the boundaries of the tube only need to be considered at each via point $k = 0, ..., n_v$, where an MPC estimation and optimization iteration occurs, the mRPI can simply be mapped to the current nominal state at each of these via points using (27).

5. Tightened constraints

Recall from Sec. III.C, to ensure that both the perturbed and nominal systems robustly satisfy their respective constraints for all time, the nominal system constraints need to be determined in terms of the real system constraints and the mRPI. At this point in the design process, all quantities required to conduct the constraint tightening procedure

Algorithm 1 Approximation of the mRPI set

Require: $\mathrm{A}_K = \mathrm{A} + \mathrm{B}\mathrm{K}_{dr}$, $\mathbb{W}, \epsilon > 0$
Ensure: $\mathcal{Z}$ such that $\mathcal{F}_\infty \subseteq \mathcal{Z} \subseteq \mathcal{F}_\infty \bigoplus \epsilon\mathbb{B}_\infty^n$
  Initialize $s \leftarrow 1$, $\alpha(s)$ using (57), and $M(s)$ using (59)
  while $\alpha(s) > \frac{\epsilon}{\epsilon + M(s)}$ **do**
    $s \leftarrow s + 1$
    Determine $\alpha(s)$ using (57)
    Set $M(s)$ using (59)
  end while
  Compute $\mathcal{Z}$ using (54) and (48)

described in (29) and (30) are known.

6. Set point characterization

It is necessary to consider the admissibility of the setpoint [28], as the desired setpoint may be unreachable in the presence of disturbances to the system or when there is mismatch between the system model and the real system.

For generality in this discussion, assume that the setpoint varies over time and the reference signal is piecewise constant for each via point, as is the case when it is desired to track a trajectory, and consider the full state-space model of the system

$$\boldsymbol{z}_{k+1} = \mathrm{A}\boldsymbol{z}_k + \mathrm{B}\boldsymbol{v}_k$$
$$\boldsymbol{y}_k = \mathrm{C}\boldsymbol{z}_k + \mathrm{D}\boldsymbol{v}_k,$$

where the output equation is now included, with the output vector $\boldsymbol{y} \in \mathbb{R}^q$ at the current time-step k, matrix C of dimension $q \times n$ and a $q \times m$ feed-through matrix D. In the state-feedback case, $\mathrm{C} = \mathrm{I}_n$ is an identity matrix of dimension $n \times n$ and D is a zero-matrix of dimension $m \times n$. When speaking about the target output and the setpoint, the state and output can be distinguished by the respective subscripts s and t such that $\boldsymbol{y}_t = \mathrm{C}\boldsymbol{z}_s + \mathrm{D}\boldsymbol{v}_s$.

It is common to parameterize the system such that for any given set point $\boldsymbol{y}_t$, any permissible terminal nominal state $\boldsymbol{z}_s^* = [\boldsymbol{z}_s, \boldsymbol{v}_s]^\top$ associated with this set point must satisfy

$$\begin{bmatrix} \mathrm{A} - \mathrm{I}_n & \mathrm{B} \\ \mathrm{C} & \mathrm{D} \end{bmatrix} \begin{bmatrix} \boldsymbol{z}_s \\ \boldsymbol{v}_s \end{bmatrix} = \begin{bmatrix} \mathbf{0}_{n,1} \\ \boldsymbol{y}_t \end{bmatrix}, \tag{62}$$

where $\mathbf{0}_{n,1}$ is a matrix of zeros and is of dimension $n \times 1$. This equation indicates that the $[\boldsymbol{z}_s, \boldsymbol{v}_s]$ pair satisfies a possibly non-zero steady state and the setpoint $\boldsymbol{y}_t$. The preceding equation (62) can be written more compactly as

$$\mathrm{E}\boldsymbol{z}_s^* = \mathrm{F}\boldsymbol{y}_t. \tag{63}$$

The solution is non-trivial when the matrix pair (A, B) is controllable.

There may exist more than one $\boldsymbol{z}_s^*$ satisfying a given y_t. For the case where $rank(\mathrm{E}) = n+m = r$, the parameterization

$$\boldsymbol{z}_s^* = \mathrm{M}_\theta \theta \tag{64}$$

is proposed in [39] to describe the set of feasible steady states, where $\theta \in \mathbb{R}^{n_\theta}$ is a parameter vector characterizing

any solution and M_θ is a suitably designed matrix. It is advocated in literature [36, 39, 40] that M_θ be defined by the singular value decomposition of E from (63), as this has the benefit of generally producing a minimal parameterization of θ. Furthermore, when $r = n + m$ under this parameterization, the solution θ is unique.

Consider, then, the singular value decomposition of $\mathrm{E} = \mathrm{U}\Sigma\mathrm{V}$, with $\mathrm{U} \in \mathbb{R}^{(n+p)\times r}$, $\Sigma \in \mathbb{R}^{r\times r}$, and $\mathrm{V} \in \mathbb{R}^{(n+m)\times r}$. From this, the following can be defined for the case that $r = n + m$

$$\mathrm{M}_\theta = \mathrm{V}\Sigma^{-1}\mathrm{U}^\top \mathrm{F}\mathrm{I}_n. \tag{65}$$

Any admissible set point $\boldsymbol{z}_s^*$ is then required to satisfy the constraint $\boldsymbol{z}_s^* = \mathrm{M}_\theta\theta \in \mathbb{Z} \times \mathbb{V} = \bar{\mathbb{Z}}$. This implies that only terminal states robustly satisfying the state and control constraints are admissible.

7. The terminal set and the invariant set for tracking

The terminal set $\mathbb{Z}_f$ must also be characterized. The stability properties of the controller largely rely on its appropriate determination. It is desirable for $\mathbb{Z}_f$ to encompass as much of the nominal state constraint set as possible. In TBMPC, this set is also formulated as RPI, containing the set of possible initial states and admissible steady states and inputs which can be admissibly stabilized by the control law. It is referred to as the invariant set for tracking $\mathcal{O}_i^e$ and the maximal robust positively invariant (MRPI) set is chosen to be used as $\mathbb{Z}_f$. The remainder of this section will be used to develop $\mathcal{O}_i^e$.

For this purpose, consider the control law

$$\boldsymbol{u} = \boldsymbol{v}_s + \mathrm{K}_O(\boldsymbol{x} - \boldsymbol{z}_s), \tag{66}$$

where $[\boldsymbol{z}_s, \boldsymbol{v}_s]$ indicates a steady state that should be reached, and assume that this law asymptotically stabilizes the closed-loop system. Just as K_{dr} was designed to determine the smallest RPI set, K_O will be used to determine the largest RPI set. Considering the parameterization of the steady state which was developed in the previous section, (66) can be re-written as

$$\boldsymbol{u} = \mathrm{K}_O\boldsymbol{x} + \mathrm{L}\theta, \tag{67}$$

where $\mathrm{L} = [-\mathrm{K}_O \;\; \mathrm{I}_m]\mathrm{M}_\theta$. It is possible to obtain a larger domain of attraction, and thereby a larger invariant set for tracking, by tuning K_O to be larger [36]. It is common practice to set K_O to be the LQR gain K_{lqr} [36], and (67) becomes

$$\boldsymbol{u} = \mathrm{K}_{lqr}\boldsymbol{x} + [-\mathrm{K}_{lqr} \;\; \mathrm{I}_m]\mathrm{M}_\theta\theta. \tag{68}$$

Define now an extended state for the closed-loop dynamics

$$\boldsymbol{z}_k^e = [\boldsymbol{z}_k \;\; \theta_k]^\top \in \mathbb{R}^{n+n_\theta} \tag{69}$$

$$\boldsymbol{z}_{k+1}^e = \begin{bmatrix} \mathrm{A} + \mathrm{B}\mathrm{K}_{lqr} & \mathrm{B}\mathrm{L} \\ \mathbf{0} & \mathrm{I}_{n_\theta} \end{bmatrix} \boldsymbol{z}_k^e, \tag{70}$$

or more compactly $\boldsymbol{z}_{k+1}^e = \mathrm{A}_e\boldsymbol{z}_k^e$. The set

$$\mathcal{O}_i^e \subset \mathbb{R}^{n\times n_\theta} \tag{71}$$

is then an admissible and RPI set if it holds that

$$z \in \mathbb{Z} \tag{72}$$

$$\mathrm{K}_O \boldsymbol{x} + \mathrm{L}\theta \in \mathbb{V} \tag{73}$$

$$((\mathrm{A} + \mathrm{BK}_O)z + \mathrm{BL}\theta,\ \theta) \in \mathcal{O}_i^e \tag{74}$$

for all $(z, \theta) \in \mathcal{O}_i^e$. This means that for any initial state z_0 located within $\mathcal{O}_i^e$, the evolving trajectory of the system controlled by (68) will satisfy the invariant set $\mathcal{O}_i^e$. Thereby, the state of the system will remain within states indicated by the projection of $\mathcal{O}_i^e$ on the admissible state space, or $z_k \in Proj_z(\mathcal{O}_i^e)$.

It follows that the use of the invariant set for tracking as the terminal set results in an MPC optimization task which does not require the re-computation of the terminal set if the terminal state changes. This allows the optimal control problem and the robust constraint satisfaction to be handled independently.

Consider finally an initial definition of the terminal set as

$$\mathbb{Z}^e_{f,\lambda_{MRPI}} = \{z^e = [z, \theta]^\top | (z, \mathrm{K}z + \mathrm{L}\theta) \in \mathbb{Z} \times \mathbb{V}, \mathrm{M}_\theta \theta \in \lambda_{MRPI} \times (\mathbb{Z} \times \mathbb{V})\}, \tag{75}$$

where $\lambda_{MRPI} \in (0, 1)$.

The extended system closed loop dynamics, the LQR gain, the parameterized θ-space, and the nominal system constraints will be used to determine this set. Similarly to the mRPI, the MRPI is generally not finitely determinable. Therefore, the set must be approximated arbitrarily closely [37, 39–41]. Arbitrarily closely here means that it is possible to obtain a convex, finitely determined polyhedron by applying a factor of λ_{MRPI} arbitrarily close to 1 to obtain

$$\mathcal{O}^e_{\infty,\lambda} = \{z^e : \mathrm{A}^i_e z^e \in \mathbb{Z}^e_{f,\lambda_{MRPI}}, \forall i \geq 0\} = \{z \in \mathbb{R}^n : \mathrm{A}_O z \leq \boldsymbol{b}_O\}. \tag{76}$$

The practical process to do this is outlined in Algorithm 2. More information can be found in [37] and [42].

In Algorithm 2, the function $Pre(\bullet)$ indicates the so-called predecessor set, or 1-step set. In practice, this pre-set is formulated as the polytope

$$\begin{bmatrix} \mathrm{A}_O & \mathrm{A}_e \\ \mathrm{A}_z & \mathbf{0} \\ \mathrm{A}_v \mathrm{K}_{lqr} & \mathrm{A}_v \mathrm{L} \end{bmatrix} z \leq \begin{bmatrix} \boldsymbol{b}_O \\ \boldsymbol{b}_z \\ \boldsymbol{b}_u \end{bmatrix}. \tag{77}$$

The algorithm effectively indicates that the computation of the pre-sets continues until two consecutive pre-sets are the same. It is also important to note that this procedure finds the invariant set for tracking in the extended space. It must be projected onto the nominal state space or the nominal control space to be able to use the result for controller analysis.

8. Nominal MPC

The optimization task of the nominal MPC is slightly modified to consider an offset cost, penalizing the deviation of a predicted artificial steady state θ from an intended steady state $\bar{\theta}$ (discussed in Section III.D.6), which can be derived from the reference data. The objective function for the optimization task then becomes

Algorithm 2 Approximation of the MRPI set

Require: $\mathbb{Z}, \mathbb{V}, \mathrm{A}_e, \mathrm{K}_{lqr}, \mathrm{M}_\theta, \lambda_{MRPI}$
Ensure: $O_\infty = O_i = O_{i+1}$
$i \leftarrow 0$
Construct the constraint set for the closed-loop system

$$\left\{ \begin{bmatrix} \mathrm{A}_z & 0 & 0 \\ 0 & \mathrm{A}_z\mathrm{M}_\theta & 0 \\ 0 & 0 & \mathrm{A}_v\mathrm{M}_\theta \end{bmatrix} z \leq \begin{bmatrix} \boldsymbol{b}_z \\ \lambda_{MRPI}\boldsymbol{b}_z \\ \lambda_{MRPI}\boldsymbol{b}_v \end{bmatrix} \right\} = \{\mathrm{A}_O z \leq \boldsymbol{b}_O\}$$

Calculate 2 pre-sets for initial comparison: $O_1 \leftarrow Pre(O_0) \cap O_0$ and $O_2 \leftarrow Pre(O_1) \cap O_1$
while $O_{i+1} \neq O_i$ **do**
$i \leftarrow i + 1$
Determine the next pre-set $O_i + 1 \leftarrow Pre(O_i) \cap O_i$
end while
Approximate MRPI set $O_\infty = O_i = O_{i+1}$

$$\boldsymbol{J(z, v, \theta)} = \sum_{\boldsymbol{i=k}}^{\boldsymbol{k+N-1}} \|\boldsymbol{z_i} - \bar{\boldsymbol{z}}_{\boldsymbol{i}}\|_{\mathbf{Q}}^2 + \|\boldsymbol{v_i} - \bar{\boldsymbol{v}}_{\boldsymbol{i}}\|_{\mathbf{R}}^2 + \|\boldsymbol{z_N} - \bar{\boldsymbol{z}}_{\boldsymbol{N}}\|_{\mathbf{P}}^2 + \left\|\boldsymbol{\theta} - \bar{\boldsymbol{\theta}}\right\|_{\mathbf{T}}^2. \tag{78}$$

The new steady-state term is positive definite, and $\left\|\theta - \bar{\theta}\right\|_{\mathrm{T}}^2 = (\theta - \bar{\theta})^\top \mathrm{T}(\theta - \bar{\theta})$. The weighting matrix T is typically designed in the region of $1000 * \mathrm{P}$ [36].This modified cost function is then minimized in the optimization task

$$\min_{\boldsymbol{z,v,\theta}} \boldsymbol{J(z, v, \theta)} \;\; \boldsymbol{s.t.} \;\; \boldsymbol{(17b)}, \;\; \boldsymbol{(17c)}, \;\; \boldsymbol{z \in x \bigoplus(-\mathcal{Z})}, \;\; \boldsymbol{(z_N, \theta) \in O^e}. \tag{79}$$

The third constraint restricts the predicted nominal state within the tube of trajectories, in turn enforcing exponential stability within this tube. The fourth constraint restrains the final state on the prediction horizon to be within O^e.

E. A note on stability and robustness

A consideration often taken for granted in the discussion of MPC is the condition of stability. A survey on stability and optimality [27] presented axioms upon which the stability of a nominal or uncertain model predictive controller can be evaluated. The nominal and uncertain axioms are essentially the same, with the exception that the uncertain system axioms must consider stability under the full extent of the possible uncertainty. For details of the proof of these axioms, the reader is directed to [27]. It shall suffice for this note to recognize to be true that if the following conditions are satisfied, then closed-loop asymptotic stability is ensured: (1) the state constraint is satisfied under the terminal set constraint; (2) the control constraint is satisfied within the terminal set constraint; (3) any current state satisfying the terminal constraint set, the next state, under the influence of uncertainty within the given bound $\mathbb{W}$, must also lie within the terminal constraint set; and, (4) the terminal cost is a local Lyapunov function.

Under TBMPC, the terminal constraint set is by definition RPI, and designed explicitly for the system dynamics and constraints. The robustness requirements have been considered in the design of $\mathcal{Z}$ and K_{dr}. The stability conditions (1)-(3) are therefore satisfied by design. Condition (4) is not guaranteed under tracking, as it is under regulation [28].

Under regulation, the consideration is only of the stability of the origin as a set point for the system. The cost function (16) is in this case necessarily decreasing, and a local Lyapunov function [27, 28]. However, when tracking a constant or piecewise constant reference signal, (16) is not necessarily decreasing. It is possible to show that the optimal cost is a Lyapunov function and that this property therefore holds, provided that the setpoint is admissible (refer to sec. III.D.6).

IV. An iterative pipeline for the development and execution of a satellite approach maneuver to a free-tumbling target using Tube-based MPC

Based on the exposition of the robust control method of interest in the previous section, the remainder of the rendezvous development and execution can be considered.

A. Pipeline interface

There are a few options for constructing the component phases into a pipeline. For example, in [11] an open-loop approach was used, where each phase is stepped through sequentially and only once. This present work suggests a pipeline oriented for use in extravehicular active debris removal activities, and addresses challenges and possible points of failure to give the planned rendezvous the best chances of success possible. Fig. 6 suggests a success-gate system: if success criteria are not met by the results of the current phase, the next phase cannot begin. At various points, there exist possibilities to return to previous phases, with information that has already been won, to obtain better results. Each phase, indicated in Fig. 6 by rounded rectangular blocks, is stepped-through sequentially in an order dictated by the success criteria. Each phase is implemented as a white box which passes the necessary information out to be used in the following phases. The success conditions (blue diamonds) are checked against this data, and the following action chosen in accordance. Possible actions include a *GO* condition which indicates that the phase completed successfully and the necessary information for the next phase is available, an *ABORT* condition which means that the pipeline cannot continue or the necessary conditions for the maneuver to commence cannot be reached, or *GOTO* which means the pipeline could not continue to the next phase but a complete failure may be avoided by returning the pipeline to a previous phase with new information.

As the pipeline commences, the first phase of *Target Motion Estimation* determines an estimate of the target inertial and state parameters and indicates a nominal set of these parameters. If these tasks are completed, the conditions of the

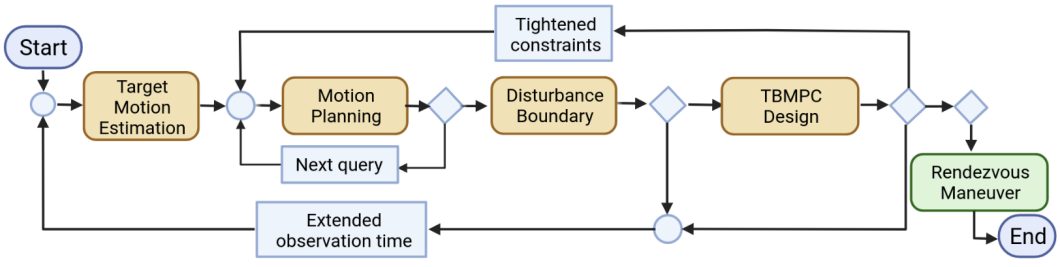


Fig. 6 Mission oriented pipeline which requires success criteria to be satisfied before the next phase can begin.

first success gate indicates a *GO* and the next phase can be entered. However, if a fatal error does occur, an *ABORT* is given and the pipeline can be re-attempted.

The second task is that of *Motion Planning*. The reference trajectory for the maneuver is generated based on the nominal query passed to it by the *Target Motion Estimation* phase. The online motion planner is unaware of the constraint tightening which will come in a later phase, and as such does not account for it. If a feasible trajectory, determined by testing if all motion constraints are satisfied, is generated, a *GO* is indicated. Otherwise, a *GOTO* this same phase can be indicated, and a trajectory generation for the next best query indicated by *Target Motion Estimation* can be attempted. If no feasible plan can be reached for the estimated target motion, an *ABORT* signal is passed.

The planned trajectory is then used in *Disturbance Boundary Determination*, which is determined here using a Monte Carlo search. Successful completion is indicated by a non-empty set $\mathbb{W}$ in each spatial dimension of position and velocity, permitting the TBMPC design to begin (*GO*). A set which has for at least one dimension a zero-limit would not allow any deviation from the nominal trajectory in these dimensions. This will trigger *GOTO Target Motion Estimation* phase to begin again with updated information about the target motion. This can be done a limited number of times before an *ABORT* is given.

Finally, the *TBMPC Design* occurs. For this problem formulation, it is imperative to determine if the nominal trajectory remains feasible under the resulting tightened constraints. If it is not, then the MPC will not be able to guide the chaser to the MP in a manner which retains the collision avoidance properties designed into the reference trajectory. If this phase has failed, two *GOTO* options are explored in this paper. *Mitigation Strategy 1* simply returns the pipeline to the *Motion Planning* phase with the set of tightened nominal constraints determined in the *TBMPC Design* attempt. A new plan is generated using these constraints and the controller design is attempted again. *Mitigation Strategy 2* exploits the multiple sets of pre-evaluated target motion prediction data which are available for the same target tumble. In fact, these data sets allow the simulated pipeline access to target motion estimations with progressively longer observation times which can be accessed when the pipeline returns to the *Target Motion Estimation* phase. This version of the pipeline would therefore re-plan the chaser motion with updated knowledge of the target. If a compatible controller is designed within a limited number of attempts, a *GO* is indicated and otherwise an *ABORT*.

B. Target Motion Estimation

The target rotational motion estimate is obtained using the method presented in [12]. This method provides (a) a statistical dispersion of the position on a spherical segment centered on the CoM of the target in which the MP could reside at the final time and (b) a set of identification parameters, including the initial rotational state and body inertia, for the propagation of the target rotational motion which would give a final MP position within this dispersion. One of these samples of identification parameters for the propagation of the target motion is taken to be the nominal for the remainder of this pipeline. The nominal motion relates to the target motion prediction that results from the average

values of the identified state and inertial parameters.

For this process, it is assumed that the chaser spacecraft possesses a visual mono-camera for capturing images of the target and that its attitude is appropriately regulated for the duration. The target is assumed to be a single rigid body, with no external forces applied, given that orbital disturbances are negligible for the observation and prediction times of interest. It is assumed here that the center of mass of the target has already been identified, for example using the methods presented in [43].

Using the images captured by the chaser, features are identified and a synthetic image generated from a geometric model of the target is aligned using a non-linear optimization technique with the captured image. This allows the current attitude of the target to be estimated with respect to the inertial frame. A series of such estimates is made over the course of an observation period, yielding a time sequence of estimated target motion. The inertial and rotational state properties are then estimated using this data series and an NLP-based identification method.

This current work makes use of results obtained during the experimental campaign presented in [12], and the method is not itself implemented here. For more detail on the methods used, the reader is directed to the cited work.

C. Motion Planning

The next phase is dedicated to planning the best trajectory from the chaser's initial position to the MP while maintaining feasibility under motion constraints. The trajectory planned in this phase corresponds to the target motion anticipated by the set of target inertial and state parameters deemed nominal and received from the preceding phase.

The planner presented here is the same used in [10], [11], and [32], with some minor parameter modifications for use in the scenario described in Sec. II rather than for implementation on the Astrobee platform. This planner was based on developments made in [5], though line-of-sight and explicit torque constraints, the additional safety consideration of plume impingement, and a warm-start recommendation for the initial guess for the planner, have now been incorporated. Through this warm-start, only a single online optimization must be conducted to generate a trajectory, in contrast to the multi-step process used in [5].

The motion planner looks to provide an optimal reference trajectory $\boldsymbol{z}_{ref}(\boldsymbol{p}, t)$ for the chaser in six-dimensional space, to include each of its six degrees of freedom. To do this, the trajectory for each translational degree of freedom is optimized with respect to mechanical energy solving a nonlinear programming problem (NLP), subject to boundary conditions on the chaser state and constraints on position, velocity, actuation, torque, line-of-sight, collision, and plume impingement. Each degree of freedom of the translational trajectory is parameterized into an order-4 b-spline and its first three derivatives, with n_p free parameters and sampled at n_v via points. Ultimately, the planning task is formulated as a discretized NLP to optimize the composite $3n_p$ free parameters $\boldsymbol{p}$ of the b-splines with respect to the listed constraints. The boundary conditions are then implicitly satisfied by the boundary conditions of the b-splines. The orientation of the chaser is dictated by its translational motion and by a line-of-

sight pointing constraint, which requires maintenance of the pointing of the x-axis of the chaser body frame at the target CoM, as described below. The chaser trajectory $z_{ref}(\boldsymbol{p}, t)$ includes position and orientation such that $z_{ref}(\boldsymbol{p}, t) = [z_{ref,p}\ \ z_{ref,o}](\boldsymbol{p}, t) = [z_{ref,x}\ z_{ref,y}\ z_{ref,z}\ q_{ref,\theta}\ q_{ref,x}\ q_{ref,y}\ q_{ref,z}](\boldsymbol{p}, t)$.

The online generation of the motion plan is conducted in two parts: a refinement process permits the use of a LUT previously compiled offline to provide a warm-start to an online optimization-based trajectory determination. The remainder of this section discusses first the online motion planning steps, followed by the offline generation of the LUT.

1. Online warm-start refinement

The main goals of this first online part of the motion planning phase are to: (a) to determine a start time of the approach maneuver to the target; (b) to provide the orientation of the target at that time; and (c) provide a warm-start to the subsequent optimization-based online trajectory planning.

Each phase of the pipeline consumes time in computation, which must be considered in the planning of the rendezvous trajectory. The time from which the nominal target motion is to be propagated is considered as t_0. The maximum anticipated computation time t_{comp} and the time span within which it is desired for the maneuver to begin t_{margin} must also be considered in the planning phase. The target motion for the given initial conditions (Sec. IV.B) are propagated from t_0 to $t_{prop} = t_0 + t_{comp} + t_{margin}$. The chaser motion is then planned to begin at some time t_{begin} between $t_{earliest} = t_0 + t_{comp}$ and $t_{latest} = t_{earliest} + t_{margin}$.

The pre-compiled LUT provides corresponding sets of initial conditions for the target motion, the composite $\boldsymbol{p}$, and the energy cost of the trajectory correspondent to these parameters to identify a suitable warm-start initial guess for the n_p free parameters. To select the best LUT entry for this purpose, the sequence of target orientations and angular velocities occurring in the timespan $[t_{earliest}\ t_{latest}]$ are compared to the sample sets of initial conditions in the LUT. The closeness of the match is determined by using the cost metrics [44]

$$J_{\mathrm{SO3}} = \min\{\|\boldsymbol{q}_{\mathrm{LUT}} - \boldsymbol{q}_{\mathrm{t}}(t)\|, \|\boldsymbol{q}_{\mathrm{LUT}} + \boldsymbol{q}_{\mathrm{t}}(t)\|\} \tag{80}$$

and

$$J_{\omega} = \|\boldsymbol{\omega}_{\mathrm{LUT}} - \boldsymbol{\omega}_{\mathrm{t}}\|, \tag{81}$$

where $\boldsymbol{q}_{\mathrm{LUT}}$ and $\boldsymbol{\omega}_{\mathrm{LUT}}$ indicate the target state query in the LUT and the target orientation and angular velocity propagated from a given set of initial conditions is given respectively by $\boldsymbol{q}_t(t)$and $\boldsymbol{\omega}_t$. Each of these metrics is normalized with their maximum value along the propagated trajectory. The three closest points of the propagated trajectory to the entries in the LUT are found. Different metrics were used to select the best matching sample from these points, e.g. the point in the LUT which: is the closest to the submitted query of initial parameters (*Closest*), has the shortest wait time for the start of the maneuver (*Quickest*), or for which the optimal energy cost of the warm start parameters is lowest (*Cheapest*). Note that the minimum energy cost has a local minimum in time, as the MP periodically comes closer to and retreats from the chaser. It is as such favorable to look for this minimum. Instead of calling the LUT function for successive

query points, until the minimum is met, successively close points along the propagated trajectory which identify the minimum are found. The online motion planner subsequently provides a reference trajectory resulting from the initial guess from the chosen point in the LUT. The start time of the maneuver t_{begin} corresponds to the time at which the target propagated trajectory is closest to this chosen LUT entry.

2. Optimization-based trajectory planning

Armed with the refined parameters with which to warm-start the trajectory optimization, the maneuver start time t_{begin}, and the initial guess to the free parameters of the b-splines, the chaser trajectory can be planned utilizing the following discretized NLP which minimizes the mechanical energy of the chaser:

$$\min_{\mathbf{p}} \; J_{me}(\boldsymbol{p}) = \Sigma({}^{o}\boldsymbol{F}^{\top}\, {}^{o}\dot{\boldsymbol{z}}_{ref})^2 dt \tag{82}$$

$$\begin{aligned} s.t.\;\; & c_{position}({}^{o}\boldsymbol{z}_{ref,p}(\boldsymbol{p}),t) \leq 0, \quad c_{velocity}({}^{o}\dot{\boldsymbol{z}}_{ref,p}(\boldsymbol{p}),t) \leq 0, \quad c_{force}({}^{o}\ddot{\boldsymbol{z}}_{ref,p}(\boldsymbol{p}),t) \leq 0, \\ & c_{torque}({}^{o}\boldsymbol{z}_{ref}(\boldsymbol{p}),{}^{o}\dot{\boldsymbol{z}}_{ref}(\boldsymbol{p}),{}^{o}\ddot{\boldsymbol{z}}_{ref}(\boldsymbol{p}),t) \leq 0, \quad c_{plume}({}^{o}\boldsymbol{r}_t,\boldsymbol{q},{}^{o}\boldsymbol{z}_{ref}(\boldsymbol{p}),{}^{c}\ddot{\boldsymbol{z}}_{ref,p}(\boldsymbol{p}),t) \leq 0, \\ & c_{spiral}({}^{o}\boldsymbol{r}_t,\boldsymbol{q},{}^{o}\boldsymbol{z}_{ref}(\boldsymbol{p}),{}^{o}\ddot{\boldsymbol{z}}_{ref,p}(\boldsymbol{p}),t) \leq 0, \quad c_{collision}({}^{o}\boldsymbol{z}_{ref,p}(\boldsymbol{p}),t) \leq 0, \\ & for \;\; t = 0,...,n_v - 1, \end{aligned}$$

where $z_{ref,p}(\boldsymbol{p},t)$ indicates the translational state of the chaser, $z_{ref,o}(\boldsymbol{p},t)$ is the orientation state of the chaser, and ${}^{o}\boldsymbol{F}$ is the applied force. The NLP has been solved using the *slsqp* algorithm provided by NLopt [45].

The chaser orientation is determined by a pointing constraint of its x-axis to the CoM of the target. This orientation, specifically that which defines two degrees of freedom of the chaser, can be determined analytically, making use of knowledge of the position of the chaser relative to the target

$${}^{o}r^{tc}(t_k) = {}^{o}\,r_t(t_k) - {}^{o}\,z_{ref}(\boldsymbol{p},t_k), \tag{83}$$

at time step k and determining two of three rotation parameters

$$\phi_y = \begin{cases} \arcsin(-{}^{c}r_z^{tc}(t_k)/{}^{c}r_x^{tc}(t_{k-1})) \;\; if k > 0 \\ \arcsin(-{}^{o}r_z^{tc}(t_k)/\|{}^{o}r_x^{tc}(t_k))\| \;\; if k = 0 \end{cases} \tag{84}$$

$$\phi_z = \begin{cases} \arctan 2({}^{c}r_y^{tc}(t_k),{}^{c}\,r_x^{tc}(t_k)) \;\; if k > 0 \\ \arctan 2({}^{o}r_y^{tc}(t_k),{}^{o}\,r_x^{tc}(t_k)) \;\; if k = 0. \end{cases} \tag{85}$$

The rotation about the x-axis is chosen as a linear motion to bring the chaser in a predefined relative orientation with respect to the target.

The required actuation to obtain this time series of chaser orientations is then obtained. The time derivatives of ϕ_y and ϕ_z in (84) and (85) are obtained from

$$\dot{\phi}_y = \frac{1}{\sqrt{1-({}^{o}r_z^{tc}/\|{}^{o}r^{tc}\|)^2}} \left(-\frac{{}^{o}\dot{r}_z^{tc}}{\|{}^{o}r^{tc}\|} + \frac{{}^{o}r_z^{tc}\|{}^{o}\dot{r}^{tc}\|}{\|{}^{o}r^{tc}\|^2} \right)$$

$$\dot{\phi}_z = \frac{1}{1 + ({}^o r_y^{tc} / {}^o r_x^{tc})^2} \left(\frac{{}^o \dot{r}_y^{tc}}{{}^o r_x^{tc}} - \frac{{}^o r_y^{tc} \, {}^o \dot{r}_x^{tc}}{({}^o r_x^{tc})^2} \right).$$

The value of $\dot{\phi}_x$ is determined by a linear interpolation from known initial and final values. The chaser angular velocity is then obtained by

$$\boldsymbol{\omega}_c = \begin{bmatrix} -\sin(\phi_y) & 0 & 1 \\ \cos(\phi_y)\sin(\phi_x) & \cos(\phi_x) & 0 \\ \cos(\phi_y)\cos(\phi_x) & -\sin(\phi_x) & 0 \end{bmatrix} \begin{bmatrix} \dot{\phi}_z \\ \dot{\phi}_y \\ \dot{\phi}_x \end{bmatrix}. \tag{86}$$

The time derivative of the angular velocity $\dot{\boldsymbol{\omega}}_{\boldsymbol{c}}$ is then determined by finite differences. The resulting applied torques result from the inverse Euler dynamics, with the angular velocity and acceleration as input. Relative orientations between via points are used, to avoid jumps in the angles which result from inverse trigonometric functions for rotations $> \pi/2$, which is a nonlinear constraint in $\boldsymbol{p}$, given the above trigonometric functions.

An important safety related constraint new to this planner inhibits plume impingement of the chaser on the target. This is achieved through c_{plume}, which seeks to prevent alteration of the target motion due to the impingement of the chaser thruster plume by restricting the chaser thrusters within a given relative distance to the target, through

$$c_{plume}({}^o \boldsymbol{r}_t, \boldsymbol{q}, {}^o \boldsymbol{z}_{ref}(\boldsymbol{p}), {}^c \ddot{\boldsymbol{z}}_{ref,p}(\boldsymbol{p}), t) \; : {}^c \ddot{\boldsymbol{z}}_{ref,x}(t) \geq 0 \;\; if \; \|{}^t \boldsymbol{r}^{tc}\| \leq \|{}^t \boldsymbol{r}^{tc}\|_{min}. \tag{87}$$

The rational is that the force in the last meters cannot be negative in the chaser body frame x-direction, noting that the x-axis of the chaser is always pointing towards the target, due to the pointing constraint.

Complementing c_{plume} and $c_{collision}$ is c_{spiral}, which is necessary to avoid collisions in-between via points towards the end of the maneuver. This constraint guarantees that the relative distance between two successive via points cannot increase, but only decrease. The need for this measure arises due to the use of spheres for representing the chaser satellite, which would overlap when in close proximity. The use of box geometry was avoided, to mitigate the possibility of unstable behavior of the optimizer.

Finally, some consideration is due to the the collision constraint $c_{collision}$. In many works, for example [8, 20, 24], the shape of the target or an obstacle is convexified or additional constraints are included to facilitate collision avoidance requirements. This approach is insufficient for approaching the MP on Envisat [33], as the MP would be located within the convexified perimeter and appendages could make the line-of-sight cone prohibitively small (see Fig. 1). Here instead, each appendage is allotted an appropriately dimensioned convex polytope, which are networked together into a single body with geometric model $\mathbb{M}_t$. Figure 1b depicts $\mathbb{M}_t$ and the spherical chaser geometry $\mathbb{M}_c$. Collision can then be detected and the minimum distance required to bring the bodies out of collision, or penetration depth d, can be determined between the $\mathbb{M}_c$ and $\mathbb{M}_t$. This network of shapes and collision detection has been implemented using the Open Dynamics Engine [46]. The collision constraint is then defined as

$$c_{collision}(\boldsymbol{z}_{ref,p}(\boldsymbol{p}, t)) : d(\boldsymbol{q}, \boldsymbol{z}_{ref,p}(\boldsymbol{p}), \mathbb{M}_t, \mathbb{M}_c, t) \leq 0, \tag{88}$$

which is a nonlinear, iterative function.

To increase the computational efficiency and decrease the time expense of the motion planner, the sparsity of the b-splines has been exploited: it is possible to pin point which segments of the trajectory change, from one optimization parameter permutation to the next, and only evaluate the gradients for the modified portions. At the beginning of each permutation, the modified parameters and the associated range of via points are identified. The constraints, constraint gradients, b-splines at each derivative, and cost function need only be evaluated at those via points. This results in a theoretical [47] relative computational improvement of

$$\frac{(1 + 3 * n_p)n_v}{n_v + 3 * n_p * n_{v/p} * n_d}, \tag{89}$$

where n_v is the number of via points, $n_{v/p}$ is the number of via points occurring between one free b-spline parameter to the next, and n_d is the number of derivatives considered by the b-spline. This analysis only considers the computation of the cost function, constraints, and constraint gradients, but not the optimization process itself.

3. Offline LUT compilation

The LUT used by the motion planner have been developed offline, based on the derived knowledge [33] of the target motion, to provide an initial guess to the online trajectory optimization. The LUT is based on a six-dimensional input space, which includes three orientation parameters and three angular velocity components, expressed in the body frame of the target. Each entry also includes the energy cost $J_{me}(\boldsymbol{p})$ from the optimal solution to (82) and the parameters $\boldsymbol{p}$ for each grid point. The orientation parameters are chosen to be 3-1-3 Euler angles $\boldsymbol{\phi}$, which are randomly sampled $n_{LUT\,\phi}$ times using the method in [48]. Given that the angular velocity, when expressed in the body frame of a rotating body, is a periodic function, it can be sampled $n_{LUT\,\omega}$ times along one period. This results in a LUT with dimensions $[n_{LUT\,\phi} \times n_{LUT\,\omega}] \times [n_\phi + n_\omega + n_{Jem} + 3n_p]$, where n_ϕ, n_ω, and n_{Jem} refer respectively to the dimensions of $\boldsymbol{\phi}$, $\boldsymbol{\omega}$, and the mechanical energy cost.

Each query in the LUT is determined through a global search, which is conducted by randomly choosing a trajectory mid-point in a user-defined region of the search space and connecting the start and end points with a straight line. One hundred solutions are sought with a maximum of N_{rand} random attempts. A pruning radius is used to avoid new mid-points occurring too closely to any of those previously considered.

An important consideration in the development of the LUT is $c_{collision}$. Information from profiling experiments revealed the computation of collision avoidance constraints to be amongst the most computationally intensive tasks within the motion planner, accounting for 37% of the overall computation time. To improve the efficiency of computing an entire LUT offline, a custom CUDA-based method [49], implemented on an NVIDIA Graphics Processing Unit (GPU) was used to specifically reduce the computation time of collision avoidance constraints.

A bounding-volume-based approach [50] was used in this implementation (see Fig. 1b). The respective number of

bounding volumes used to construct the chaser and target hulls are given by $n_{bv,c}$ and $n_{bv,t}$. The necessary penetration depth computations through the duration of the trajectory can be described as a three-dimensional array, of size $n_{bv,c} \times n_{bv,t} \times n_v$. Each element of the array is mapped to a separate thread of execution on the GPU. For the scenario under consideration $n_{bv,c} = 1$ and $n_{bv,t} = 4$. At each via point, a set of penetration depth computations is performed between each pair of bounding volumes on the separate bodies, in parallel on the GPU. CUDA kernels capable of being executed on the GPU were implemented for collision detection between pairs of bounding volumes in the scene, the subsequent penetration depth computations, and for computing the least-violating penetration depth constraint to return to the optimizer, in the case of multiple simultaneous penetrations. The individual kernels were implemented based on optimized versions of algorithms [50]. For a other problem formulations, the method can be trivially extended.

The machine code for the to intersection tests of different pairs of bounding volumes can be vastly different from each other, resulting in warp divergence. To circumvent this issue, a strategy was devised involving CUDA *streams* and concurrent kernel execution. A *stream* is a sequence of operations executed on a device in the order in which they are issued by the host code. Concurrent kernel execution refers to the ability to simultaneously execute more than one kernel if the resources of the GPU allow for it. Usage of *streams* generally helps to better utilize the GPU in two ways: (a) memory copies between host and device can be overlapped by kernel execution if copying and execution occurs in different *streams*, and (b) individual kernels running in different *streams* can overlap if there are enough resources on the GPU. In this work, devices of NVIDIA Compute > 6 were used, which support concurrent kernel execution. Different CUDA kernels corresponding to penetration depth computations between each pair of bounding volumes were launched concurrently on separate *streams*, and data transfer to and from the CPU to the GPU was overlapped with kernel execution using CUDA *streams* and asynchronous copy. A synchronization barrier is set up such that after these kernels are finished executing, a final reduction kernel is executed to find the least violating penetration depth constraint for each via point to return to the optimizer. Further optimizations are performed to avoid data transfer between the GPU and CPU when bounding volumes do not change positions between optimizer iterations.

D. Disturbance boundary

The final step of preparation for the development of the tube-based model predictive controller is to determine the uncertainty boundary. The relationships (11) and (12) can be used to derive the state uncertainty as a function of $\delta\boldsymbol{\omega}_t(t)$

$$^o\mathbf{w}(\delta\boldsymbol{I}_t, t) = [^o\boldsymbol{w}_p(\delta\boldsymbol{I}_t, t),\ \ ^o\boldsymbol{w}_v(\delta\boldsymbol{I}_t, t)]^\top = [^o\boldsymbol{z}'_p(t) - ^o\boldsymbol{z}_p(t),\ \ ^o\boldsymbol{z}'_v(t) - ^o\boldsymbol{z}_v(t)]^\top. \tag{90}$$

To determine a practical state uncertainty bound for the given dispersion of uncertainty in target motion, a statistical analysis is conducted using the n_{mp} samples provided by the *Target Motion Prediction* phase. For each sample of the target inertial and state parameters, the motion constraint satisfaction of the perturbed trajectory $^o\boldsymbol{z}'$ is evaluated. If the motion constraints are satisfied, the maximum difference of the perturbed trajectory with respect to the nominal is found for each dimension of the chaser state in both the positive and negative directions. Finally, the maximum values for each

dimension of the chaser state are deduced from the n_{mp} perturbed trajectories, which become the boundaries for the bounded state uncertainty $\mathbb{W}$.

E. TBMPC design

Finally, the automatic TBMPC design can be conducted based on the theory presented in section III and the information development in the preceding phases of the pipeline. As in section III, the design implemented here can be stepped through, following Fig. 5. For this application, $n = 6$ and $m = 3$.

At this point, part of the challenge of this problem formulation and controller choice should be clear: the disturbance boundary, $\mathbb{W} \subset \mathbb{R}^6$, is nominal trajectory dependent. It is not obvious at the outset of the planning phase or the TBMPC design phase if the dimensions of $\mathbb{W}$ will be tolerable or if the requisite constraint tightening, which is dependent on the disturbance boundary, will permit the designed nominal trajectory. It is possible that the planned trajectory, determined disturbance bound, and resultant constraint shrinking create a scenario where only the planned trajectory is permissible, calling into question the extra effort, computational resource, and time necessary to determine the disturbance rejection gain, mRPI, and MRPI, as well as compiling a model predictive controller on the fly. This is often considered to be a limitation of TBMPC. However, it is argued here that this can actually be viewed as an advantageous feature. The results of the TBMPC design phase can be exploited, not just as a way to design a robust controller, but also as a litmus test of the planned maneuver and its robustness when environmental uncertainties are known to be present, but the extent of which is unknown until the time of execution.

V. Test cases and results

In this section, the developed motion planner, the TBMPC controller design, and the proposed pipeline are put to the test using the scenario described in Section II. For this evaluation, the MP is located in the target body frame at ${}^{t}r_{MP} = [1.97,\ 0,\ 2.73]$ m, $T_{orb} = 0.0012$ rad/s, and the approach maneuver duration is $t_{duration} = 600$ s. To satisfy similar parameters to the e.Deorbit scenario and to simulate final approach maneuvers, the chaser is constrained to move within $\boldsymbol{x}_{p,max} = 45$ m of the target with $\boldsymbol{x}_{v,max} = 1$ m/s, $\boldsymbol{F}_{max} = 65$ N, and $\boldsymbol{\tau}_{max} = 0.5\boldsymbol{F}_{max}$. For collision detection purposes, the chaser is taken to have a spherical geometry of radius 0.75 m and a mass of 1000 kg. The target is considered to have the geometry and inertial properties of Envisat (see Fig. 1). The b-splines are designed to have $n_p = 20$ free parameters and $n_v = 801$, based on the target appendage tip speed so that collisions will not be overlooked in the time between two via points. The TBMPC design uses a prediction horizon $N = 20$ and weighting matrices $\mathrm{Q} = 10^{-6}\mathrm{I}_6$, $\mathrm{R} = 10^{-8}\mathrm{I}_3$, and $T = 1000P$ selected using typical tuning rules. Unless otherwise indicated, all of the presented results were obtained using a computer equipped with a Intel(R) Core(TM) i7-9700 CPU.

A. Motion planning

The presented planner has been statistically evaluated for its functionality under the prescribed task using a LUT developed for a tri-axial target tumble (characterized by an initial angular velocity ${}^{o}\omega_t = [0, 3.53, 3.53]$ deg/s). The statistical evaluation includes 200 queries, sampled along the derived target tumble state, as the input target tumbling conditions. Each target state query is passed sequentially to the planner and its result evaluated. The three options for the warm start criterion, referred to in section IV.C.1, have each been considered. The success rate and timing statistics for queries to the planner are provided in Table 1. The stopping condition applied is the cost function accuracy of $1e-6$.

Table 1 Statistics of Motion Planner with optimization warm start

	Closest	***Quickest***	***Cheapest***
Success rate [%]	92	90	95
Time mean [s]	3.675	2.0623	1.3628
std. dev. [s]	3.025	0.41933	0.70629
2σ [s]	9.726	2.901	2.7753
3σ [s]	12.75	3.3203	3.4816

For comparison, Table 2a indicates the same statistical analysis when no warm start is provided to the trajectory optimization. The initial guess in this case is taken as a straight line from the chaser initial position to the MP. There is a clear time - and implicit computational - benefit to the warm start method, which has been shown to be up to 98.6 times faster than the cold start method. Similarly, Table 2b indicates that exploiting the sparsity properties of b-splines, substantial time and computational savings can be achieved. In this case, with 801 via points and a warm start provided, the theoretical achievable relative reduction of computation, from (89), is 5.76. Practically, the planner completes up to 3.6 times faster when the sparsity of the b-spline is utilized. Additional analysis indicates that the optimizer can utilize an unpredictably large proportion of the computational effort, which skews the true time and computational gains.

Table 2 Statistics of Motion Planner using alternative methods

(a) A cold start to trajectory optimization

	Cold start
Success rate [%]	50
Time mean	134.18
std. dev.	136.44
2σ	407.05
3σ	543.49

(b) Non-sparse b-spline handling

	Closest	***Quickest***	***Cheapest***
Time mean	5.11	5.09	4.71
std. dev.	2.67	2.46	2.57
2σ	10.46	10.03	9.87
3σ	13.13	12.49	12.45

Finally, in [8], it is indicated that the relative orientation of the target at the start time of the maneuver has a dramatic impact on the cost and computational complexity of the maneuver. For the spin presented in [8], there is a clear and expected periodic rise and fall in the number of sequential optimization iterations and the value of the solution cost as the MP rotates away from and back around to face the initial position of the chaser. The extension of this logic to a tri-axially tumbling target should be fairly clear and it is beyond the scope of this paper to formally prove the relationship. The methods and warm-starting criteria presented here, in a practical sense, disjoin the relationship, with the final cost

and computational effort far more reliant on the warm start criteria and the selected seed for the NLP.

Furthermore, it is still possible to observe a similar effect to that which was exploited by [8] to obtain a solution to the translational motion plan. In that work, the authors indicate that repeated seeding of the optimization problem with the previous result as the current initial guess will yield a sequence of admissible results with non-strictly decreasing cost. This decreasing cost of the optimal trajectory obtained when using the cost of the LUT query, or *Cheapest*, metric against which to choose the warm-start has been observed, with a further note of monotonically increasing cost when using the second and third seed options. It is noted that in contrast to [8], the hull of the target has not been convexified and only a single NLP is solved online.

B. The use of a GPU for computing collision avoidance constraints

A set of experiments was performed, using a server consisting of an Intel(R) Xeon(R) E5 2687W v3 processor running at 3.1 GHz and an NVIDIA 1080 (Pascal) GPU, to measure the average reduction of computation time obtained by using the implemented GPU-based method for computing collision avoidance constraints, relative to a reference baseline CPU-based implementation. The latter consisted of the motion planner with collision constraints computed using ODE [46]. For the purpose of bench-marking, a synthetic workload of 660 queries within the LUT was considered, at the same precision for both PC- and GPU-based tests. Within these experiments, the time taken for computing collision avoidance constraints as a fraction of the overall computation time was measured, revealing the factors of improvement obtained for the computation of collision avoidance constraints alone and for the LUT computation obtained using the GPU-based version versus the CPU-based version. Additionally, attention is given to the accuracy of the implemented method for computing collision avoidance constraints on the GPU in comparison to the CPU-based method, as fast but highly inaccurate results would render the method useless for motion planning purposes. The following metrics are of particular interest: the ratio of trajectories that converge using the GPU-based method, in comparison to the CPU-based method; the mean and maximum difference in the computed penetration depth constraints and cost function values, using both methods; and the degree of similarity between the trajectories themselves computed by both methods, most notably for the trajectories taken to be the "global minimum" for each query. To validate the effectiveness of the GPU-based method, LUTs computed using the GPU-based and CPU-based implementations were used to warm start the motion planner with a suitable initial guess for the same set of queries.

1. Performance

Through the use of the methods described in Sec. IV.C.3, the fraction of time taken for collision avoidance is reduced from 37% to 6%. Thus, the proposed NVDIA-based GPU implementation is found to improve performance by a factor of 6.2 over the baseline implementation using ODE. This result holds irrespective of the number of queries in the LUT. It is noted that although the fraction of time taken for collision avoidance is greatly reduced, this does not translate into

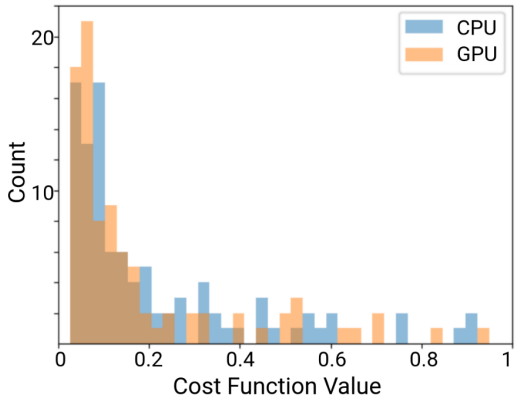


Fig. 7 Cost function histograms using the CPU- and GPU-based implementations, for a representative query.

an equivalent reduction in the overall computation time. This is because only 37% of the code could be parallelized through the use of a GPU, and this theoretically limits the overall run time reduction to a factor of 1.58. A maximum speed-up factor of 1.45 in the overall run time of the motion planner was observed.

2. Accuracy

The total number of trajectories which converges using the GPU-based implementation for a LUT of 660 queries with 100 trials per query was 60601. This number for the baseline reference implementation was 60298. These numbers show that on average, the GPU-based implementation is just as good (in fact slightly better) as the baseline reference implementation in terms of the convergence rate. Furthermore, no case was observed where the reference baseline converges to a solution, but the GPU-based version does not.

The differences in optimal cost function values are found to be within acceptable limits for motion planning purposes. The GPU-based implementation converges to a trajectory with a cost function value within 2% of the cost of an equivalent trajectory output by the CPU-based method on average and 11% maximally. Figure 7 shows overlapped cost function histograms which depict the frequency of local minima hits for both implementations, for a representative query with 100 Monte Carlo trials. It is noticed that there is a high degree of similarity between the histograms of the two approaches. This is especially true for the "cheaper" solutions, which are more likely to be globally optimal.

C. Uncertainty bound determination and TBMPC design

A similar statistical analysis which was conducted in V.A has been conducted for the determination of the uncertainty boundary determination and the TBMPC design. The following results were obtained using data provided by the *Target Motion Prediction* phase after observing the target for one quarter of its angular velocity period (polhode period), or 37.4 s [12]. In this case the target motion estimation process has provided 80 samples of target inertia, initial orientation, and initial angular velocity. Each sample is passed to the motion planner as a query and a nominal trajectory is generated using a *Cheapest* warm start to the optimization. The other samples are used to find the boundaries for $\mathbb{W}$ as described in section IV.D. The TBMPC is then designed for each of these nominal trajectories and disturbance bounds.

It was observed that up to 60 the perturbed trajectories ${}^{o}\boldsymbol{z}'$ were feasible with respect to the motion constraints for the sample of queries, though on average only 31.69 ${}^{o}\boldsymbol{z}'$ were feasible. Given the wide variation in the number of feasible perturbed trajectories per nominal trajectory, this result reflects how the disturbance bound is highly dependent on the target motion prediction data available and on which sample is chosen to serve as the nominal. Additionally, 19 of the 80 nominal target motion prediction samples and their planned trajectories obtain their disturbance boundary from 10 or fewer perturbed trajectories, as the rest violate at least one motion constraint. It would be prudent to base the nominal inertial and motion parameters on a target motion prediction sample which obtains its disturbance bound from a larger number of feasible perturbed trajectories. The analysis indicates that the target inertial and motion parameter sample which result in a propagated MP position nearest to the mean value of the propagated MP positions of all of the samples is a suitable choice for the nominal parameters. Finally, this phase has been observed to have a mean duration of 2.59 s.

Table 3 TBMPC design computation time and constraint tightening statistics

	Computation time [s]	**State Reduction** [m, m, m, m/s, m/s, m/s]	**Actuation Reduction** [N]
mean	0.901	[6.99, 23.63, 18.4, 3.99, 0.29, 0.62, 0.29]	[3.8, 21.7, 12.5]
maximum	0.994	[9.02, 25.92, 29.91, 0.41, 0.79, 0.81]	[25, 47.4, 30.1]

Table 3 indicates that a severe restriction of the nominal motion constraints is possible. It is therefore important that the planned trajectory satisfies the tightened constraints. If not, mitigation steps will need to be taken, for example planning with a different nominal query or re-planning the nominal trajectory using the tightened constraints.

D. Pipeline

In this section, the overall pipeline and the designed controller's involvement are presented in simulation. Both mitigation strategy outlined in Sec. IV.A are considered. Each simulation begins using the first set of available observation data, with an observation time $t_0 = t_{obs,Plan1} = 37.4$ s. For this implementation, $t_{comp} = 29$ s is considered so that the maximum observed computation time of all phases can easily be accommodated (see results presented in Secs. V.A and V.C). A value of $t_{margin} = 60$ s was selected. The maneuver would therefore begin between 29 and 60 s after a successful TBMPC design process. The sample with a propagated MP position nearest to the mean value, with identified target state and inertial parameters given in the second column of Table 4, is taken to be the nominal.

Table 4 Estimated target parameters

Time t_{obs}	0.25 $T_{polhode}$	0.5 $T_{polhode}$
I_t	$\begin{bmatrix} 8057.92 & 133.729 & -162.356 \\ 133.729 & 3355.16 & 51.5585 \\ -162.356 & 51.5585 & 6554.11 \end{bmatrix}$	$\begin{bmatrix} 6629.58 & 204.48 & -12.65 \\ 204.48 & 3421.54 & 33.24 \\ -12.65 & 33.24 & 5263.14 \end{bmatrix}$
$\boldsymbol{q}_{t,N}^{\mathrm{T}}$	[0.996, 0.0118, 0.0014, 0.085]	[0.99, 0.13, −0.0069, −0.031]
$\boldsymbol{\omega}_{t,N}^{\mathrm{T}}$ [rad/s]	[0.084, 0.0105, 0.02]	[0.081, 0.033, 0.006]

The first row of plots in Fig. 8 displays the results of this first attempted design. Fig. 8a reflects both the initial guess provided by the LUT and the resulting planned trajectory. Likewise, Fig. 8b shows the actuation. The bounds of each plot reflects the full extent of the position and actuation box constraints, respectively 45 m and 65 N. Both the initial guess and planned optimal trajectory satisfy these constraints, and the pipeline continues on to determine the uncertainty boundary. The second column of Table 5 indicates the disturbance bound determined for the associated planned nominal trajectory ${}^{o}\boldsymbol{z}$. As a feasible plan and a disturbance bound can be satisfactorily determined, the TBMPC design can be conducted. This phase completes unsuccessfully: the planned nominal trajectory does not satisfy the tightened constraints given in the second column of Table 6 and indicated by the black dashed lines in Figs. 8a and 8b.

The simulation now diverges into two redesign options to obtain a compatible nominal trajectory and controller pair.

1. Mitigation Strategy 1: Re-planning with tightened constraints

For this first option, the pipeline returns to the planning phase, passing the designed tightened constraints to the motion planner. A new reference trajectory is generated and the disturbance bound given in the third column of Table 5 is determined, with a start time that must now lie between $[t_{earliest}\ t_{latest}]$ with $t_0 = t_{obs,Plan1} + t_{elapse}$, where t_{elapse} is the time elapsed during the previous iteration of the *Motion Planning* phase, through the *Disturbance Bound Determination* and *TBMPC Design* phases, and up to the start of this iteration of the *Motion Planning* phase. With the new disturbance boundary, the TBMPC design completes successfully with an indication that the reference trajectory satisfies the new tightened constraints (Table 6), and the parameters of the tube-based model predictive controller are set. Figs. 9a and 9b depict the planned trajectory and actuation. The designed tightened constraints are indicated by the black lines and the tube within which the controlled trajectory must remain is given by the magenta curves.

Table 5 Disturbance boundary

	Initial Boundary	**Mitigation Strategy 1**	**Mitigation Strategy 2**
${}^{o}\mathbb{W}_x\ [m]$	5.83	4.58	4.82
${}^{o}\mathbb{W}_y\ [m]$	20.25	22.35	15.05
${}^{o}\mathbb{W}_z\ [m]$	20.84	17.78	31.31
${}^{o}\mathbb{W}_{\dot{x}}\ [m/s]$	0.31	0.15	0.38
${}^{o}\mathbb{W}_{\dot{y}}\ [m/s]$	0.19	0.20	0.38
${}^{o}\mathbb{W}_{\dot{z}}\ [m/s]$	0.76	0.48	0.6

Table 6 Tightened Constraints

	Initial Constraints	**Mitigation Strategy 1**	**Mitigation Strategy 2**
$\mathbb{Z}_x\ [m]$	38.95	39.91	39.08
$\mathbb{Z}_y\ [m]$	19.74	23.59	30.89
$\mathbb{Z}_z\ [m]$	20.15	27.27	14.65
$\mathbb{Z}_{\dot{x}}\ [m/s]$	0.68	0.83	0.6
$\mathbb{Z}_{\dot{y}}\ [m/s]$	0.79	0.79	0.6
$\mathbb{Z}_{\dot{z}}\ [m/s]$	0.21	0.49	0.39
$\mathbb{V}_x\ [N]$	64.86	43.21	43.05
$\mathbb{V}_y\ [N]$	62.41	57.52	56.3
$\mathbb{V}_z\ [N]$	64.75	43.36	41.2

2. Mitigation Strategy 2: Re-planning with extended observation time

In this strategy, the pipeline returns to the target motion estimation phase. In a real mission, the target observation would continue in parallel to the first design phase and a target state estimate would be derived from the extended observation data. In the data set used in this work, the next available set of target state estimates was obtained with an

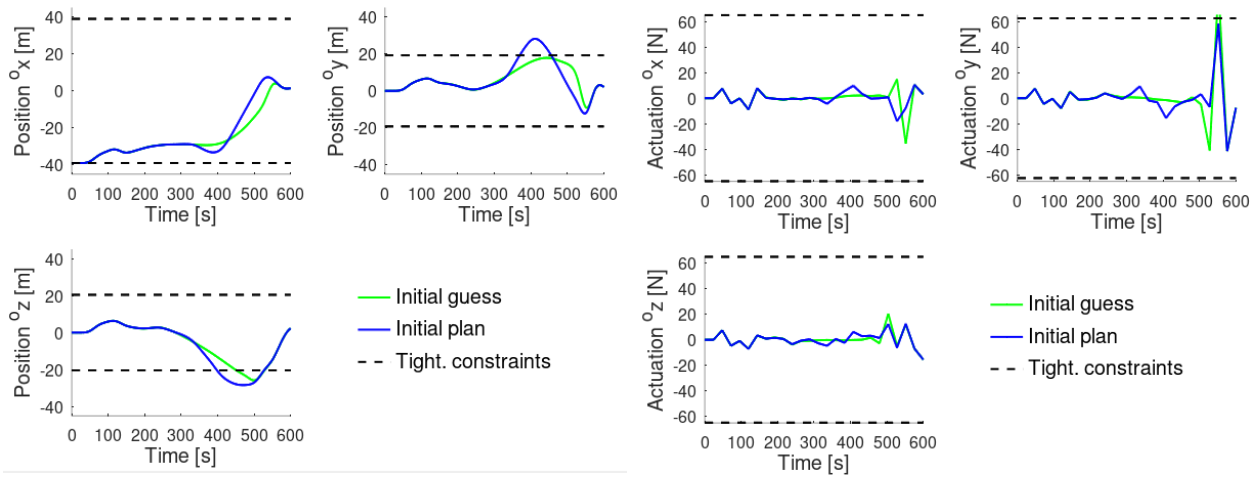


Fig. 8 Initial (a) trajectory and (b) actuation developed by the pipeline.

observation time of 0.5 $t_{polhode}$, or at $t_0 = 74.8$ s from the commence of observation. The updated nominal condition of the target state and inertial parameters is given in Table 4. A new plan is developed using the same initial constraints as the initial plan. The TBMPC design process completes successfully and the controller is set, see Figs. 10a and 10b.

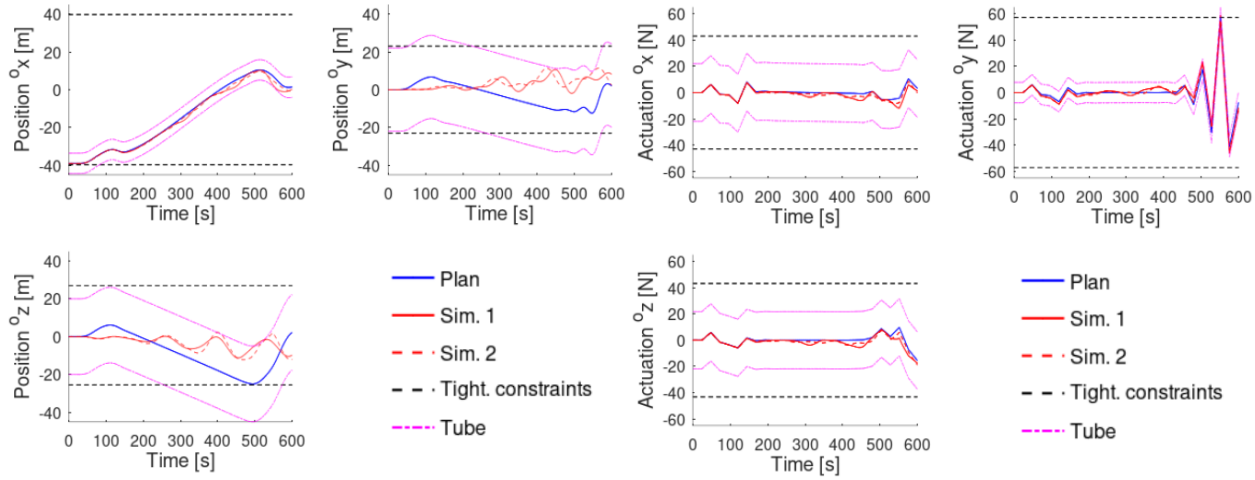
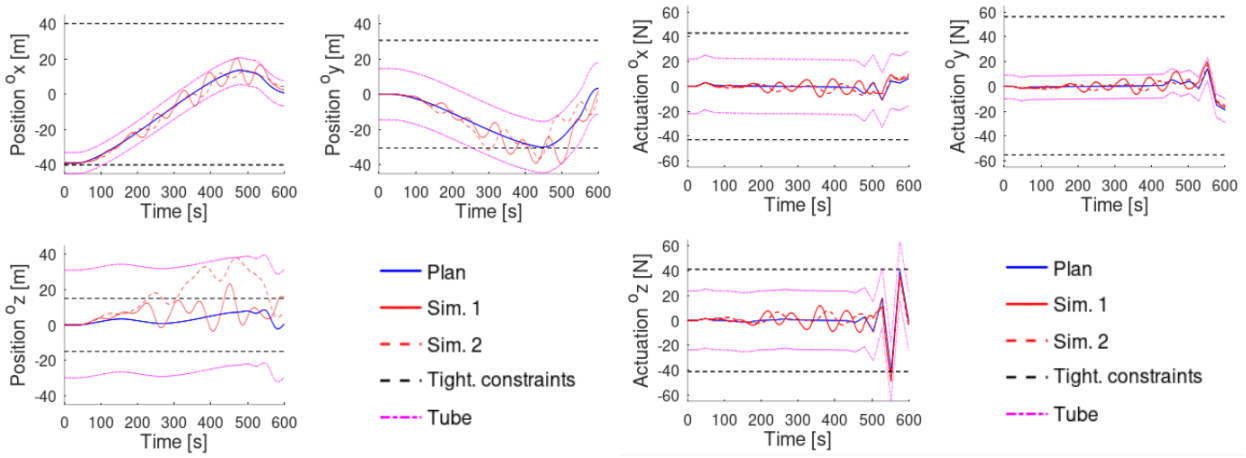


Fig. 9 Trajectory (a) and actuation (b) resulting from re-planning with tightened constraints.

3. Trajectory tracking simulation

In addition to the planned trajectory, Figs. 9 and 10 present two simulated controlled trajectories for the chaser motion for each of the successful plans. The target inertial and motion parameters used in Simulations 1 and 2 were provided by the target motion estimation data sets. Each simulation is required to track the planned trajectory, as seen in the target body frame, and complete the maneuver in 600 s according to plan. For the purposes of this simulation, no sensor noise has been considered. The maximum tracking error as seen in the target body frame is $2e - 4$ m, which is reasonable for retaining the collision avoidance properties of the planned trajectory, though difficult to appreciate in



(a) **(b)**

Fig. 10 The trajectory (a) and actuation (b) obtained by re-planning after extended observation of the target.

pictorial form. The uncertainties in the inertial parameters δI and target tumble state $\delta\omega$ are given in Tables 7 and 8.

Table 7 Parameter Uncertainty: Mitigation Strategy 1

	Simulation 1	**Simulation 2**
δI	$\begin{bmatrix} 2014.900 & 132.72 & -78.75 \\ 132.725 & 957.68 & 282.52 \\ -78.75 & 282.52 & 1562.64 \end{bmatrix}$	$\begin{bmatrix} 1403.730 & 10.307 & -41.619 \\ 10.307 & 552.180 & 26.085 \\ -41.619 & 26.085 & 1152.380 \end{bmatrix}$
$\delta\omega$ [rad/s]	$[0.0036,\ 0.0022,\ -0.0112]$	$[0.0002,\ 0.0013,\ -0.0017]$

Table 8 Parameter Uncertainty: Mitigation Strategy 2

	Simulation 1	**Simulation 2**
δI	$\begin{bmatrix} 7366.54 & 152.428 & -60.9852 \\ 152.428 & 3220.58 & 30.4125 \\ -60.9852 & 30.4125 & 6036.23 \end{bmatrix}$	$\begin{bmatrix} -1296.600 & 91.600 & 88.170 \\ 91.600 & 516.250 & -70.820 \\ 88.170 & -70.820 & -1584.870 \end{bmatrix}$
$\delta\omega$ [rad/s]	$[-0.0028,\ 0.006,\ 0.0024]$	$[-0.0053,\ 0.0218,\ -0.0012]$

Similar performance is obtained by both mitigation strategies - all four simulations complete with reasonable tracking error, on time, and reaching their expected MP. It is interesting, but likely not typical, that both options provided converging and compatible solutions after only two iterations of motion planning and controller design. It is interesting to refer back to Table 6, which indicates that the tightened state constraints, which were violated under the initial motion plan and controller design, have generally become less restrictive to the nominal trajectory under the mitigation strategies. This points to the unpredictable and nonlinear relationship of the nominal trajectory to the disturbance boundary. The tube constraints are often activated by the perturbations experienced in the last quarter of the maneuver. This occurs in all four simulations and largely correlates to the period when the chaser is in close proximity to the target.

Mitigation Strategy 1 has explicitly considered the difficulty of the constraint tightening, while *Mitigation Strategy 2* has considered the possibility that a longer observation of the target, and therefore a more accurate estimation of its parameters, reduces the uncertainty in the requisite chaser motion. Tables 5 and 6 suggest that using *Mitigation Strategy*

1 could, to an extent, be relied upon to alleviate some difficulty in the TBMPC design phase. This strategy could be used for multiple iterations, but there will be a point where the increase in computational effort will cause the time anticipated for the planning phase to be exceeded and the timing of the start of the maneuver to become unreliable. This point would need determination offline before the mission, and could be incorporated into the *ABORT* decision tree. Table 5 suggests that more accurate information about the target inertia and motion parameters does not directly ensure a smaller disturbance boundary. For best chances of mission success, a combination of both methods should be employed.

VI. Conclusion

As computational guidance and control for the active-debris removal problem shifts its focus toward autonomous planning and control, robust online methodologies have increased in importance. This paper has presented and demonstrated suitable methodologies and a pipeline in which they can be successfully applied.

The presented motion planner generates optimal trajectories along which the chaser should move to meet an appropriate mating point on a non-cooperative tumbling target satellite. These trajectories are obtained through the online resolution of an NLP, which optimizes the free parameters of a set of b-splines and is constrained by nonlinear motion constraints including actuation, plume impingement, chaser pointing, and non-convex collision avoidance constraints. The NLP is provided a warm-start from a LUT pre-generated offline. The efficiency of the motion planner has been substantially improved through the optimization problem formulation, the use of novel GPU methods to generate the LUT, and the exploitation of the sparsity of the optimized b-splines.

The TBMPC controller selected for the execution of the trajectory tracking rendezvous allows a departure from the nominal reference trajectory provided by the motion planner, while not exceeding the actuation available to the chaser satellite; therefore remaining feasible. The theories necessary for the implementation for such a controller have been consolidated and clarified. The practicality of the necessary constraint tightening procedure has been explored, demonstrating that the possible over-conservativeness of the controller design can be a useful tool in determining the feasibility of the desired maneuver.

In this paper, a closed-loop TRACE pipeline was presented for the execution of on-orbit autonomous rendezvous with non-cooperative, tumbling targets. This pipeline, constructed of modular phases, allows for logical loop-backs to previous steps, based on the success of each sequential phase. Two failure mitigation strategies are proposed, which reduce the risk of failure due to incompatibility of motion plan with respect to the automated controller design. The pipeline has been demonstrated in simulation to robustify the open-loop implementation of the original TRACE pipeline.

References

[1] Hillenbrand, U., and Lampariello, R., "Motion and parameter estimation of a free-floating space object from range data for motion prediction," *European Space Agency, Special Publication*, 2005.

[2] Lampariello, R., "Motion Planning for the On-orbit Grasping of a Non-cooperative Target Satellite with Collision Avoidance," *International Symposium on Artificial Intelligence, Robotics and Automation in Space*, Vol. 1, 2010, pp. 636–643.

[3] Aghili, F., "Optimal Control for Robotic Capturing and Passivation of a Tumbling Satellite with Unknown Dynamics," *AIAA Guidance, Navigation and Control Conference and Exhibit*, 2008. https://doi.org/10.2514/6.2008-7274.

[4] Aghili, F., "A prediction and motion-planning scheme for visually guided robotic capturing of free-floating tumbling objects with uncertain dynamics," *IEEE Transactions on Robotics*, Vol. 28, No. 3, 2012, pp. 634–649.

[5] Stoneman, S., and Lampariello, R., "A nonlinear optimization method to provide real-time feasible reference trajectories to approach a tumbling target satellite," *2016 International Symposium on Artificial Intelligence, Robotics and Automation in Space*, Beijing, P.R. China, 2016.

[6] Buckner, C., and Lampariello, R., "Tube-based model predictive control for the approach maneuver of a spacecraft to a free-tumbling target satellite," *2018 Annual American Control Conference (ACC)*, Milwaukee, WI, USA, 2018, pp. 5690–5697.

[7] Terán Espinoza, A., Hettrick, H., Albee, K., Hernandez, A. C., and Linares, R., "End-to-End Framework for Close Proximity In-Space Robotic Missions," *International Astronautical Congress (IAC)*, Washington, D.C., 2019, pp. 1–13.

[8] Virgili-Llop, J., Zagaris, C., Zappulla II, R., Bradstreet, A., and Romano, M., "A convex-programming-based guidance algorithm to capture a tumbling object on orbit using a spacecraft equipped with a robotic manipulator," *The International Journal of Robotics Research*, Vol. 38, 2019, pp. 40–72.

[9] Aghili, F., "Optimal Trajectories and Robot Control for Detumbling a Non-Cooperative Satellite," *Journal of Guidance, Control, and Dynamics*, Vol. 43, No. 5, 2020, pp. 981–988. https://doi.org/10.2514/1.G004758.

[10] Albee, K., Specht, C., Mishra, H., Oestreich, C., Brunner, B., Lampariello, R., and Linares, R., "Autonomous Rendezvous with an Uncertain, Uncooperative Tumbling Target: The TumbleDock Flight Experiments," *ASTRA 2022*, 2022.

[11] Albee, K., Oestreich, C., Specht, C., Terán Espinoza, A., Todd, J., Hokaj, I., Lampariello, R., and Linares, R., "A Robust Observation, Planning, and Control Pipeline for Autonomous Rendezvous with Tumbling Targets," *Frontiers in Robotics and AI*, Vol. 8, 2021, p. 234.

[12] Lampariello, R., Mishra, H., Oumer, N. W., and Peters, J., "Robust Motion Prediction of a Free-Tumbling Satellite with On-Ground Experimental Validation," *Journal of Guidance, Control, and Dynamics*, Vol. 44, No. 10, 2021, pp. 1777–1793.

[13] Richards, A., Schouwenaars, T., How, J., and Feron, E., "Spacecraft Trajectory Planning with Avoidance Constraints Using Mixed-Integer Linear Programming," *Journal of Guidance, Control, and Dyanamics*, Vol. 25, No. 4, 2002, pp. 755–764.

[14] Boyarko, G., Yakimenko, O., and Romano, M., "Optimal Rendezvous Trajectories of a Controlled Spacecrafte and a Tumbling Target," *Journal of Guidance, Control, and Dyanamics*, Vol. 34, No. 4, 2011, pp. 1239–1252.

[15] Lu, P., and Liu, X., "Autonomous Trajectory Planning for Rendezvous and Proximity Operations by Conic Optimization," *Journal of Guidance, Control, and Dynamics*, Vol. 36, No. 2, 2013, pp. 375–389.

[16] Pan, J., and Manocha, D., "GPU-based parallel collision detection for fast motion planning," *International Journal of Robotic Research*, Vol. 31, 2012, pp. 187–200.

[17] Park, C., Pan, J., and Manocha, D., "Real-time optimization-based planning in dynamic environments using GPUs," *2013 IEEE International Conference on Robotics and Automation*, 2013, pp. 4090–4097.

[18] Chrétien, B., Escande, A., and Kheddar, A., "GPU Robot Motion Planning Using Semi-Infinite Nonlinear Programming," *IEEE Transactions on Parallel and Distributed Systems*, Vol. 27, 2016, pp. 1–1.

[19] Lu, P., "Introducing computational guidance and control," *Journal of Guidance, Control, and Dynamics*, Vol. 40, No. 2, 2017, p. 193.

[20] Morgan, D., Chung, S., and Hadaegh, F., "Model Predictive Control of Swarms of Spacecraft Using Sequential Convex Programming," *Journal of Guidance, Control, and Dynamics*, Vol. 37, No. 6, 2014, pp. 1725–1740.

[21] Leomanni, M., Rogers, E., and Gabriel, S., "Explicit Model Predictive Control Approach for Low-Thrust Spacecraft Proximity Operations," *Journal of Guidance, Control, and Dynamics*, Vol. 37, No. 6, 2014.

[22] Petersen, C., Leve, F., and Kolmanovsky, I., "Model Predictive Control of an Underactuated Spacecraft with Two Reaction Wheels," *Journal of Guidance, Control, and Dynamics*, Vol. 40, No. 2, 2017.

[23] Mammarella, M., Capello, E., Park, H., Guglieri, G., and Romano, M., "Spacecraft proximity operations via tube-based robust model predictive control with addititve disturbances," *68th International Astronautical Congress*, Adelaide, Australia, 2017.

[24] Zagaris, C., Park, H., Virgili-Llop, J., Zappulla, R., Romano, M., and Kolmanovsky, I., "Model predictive control of spacecraft relative motion with convexified keep-out-zone constraints," *Journal of Guidance, Control, and Dynamics*, Vol. 41, No. 9, 2018, pp. 2051–2059.

[25] Tsiotras, P., and Mesbahi, M., "Toward an algorithmic control theory," *Journal of Guidance, Control, and Dynamics*, Vol. 40, No. 2, 2017, pp. 194–196.

[26] Eren, U., Prach, A., Koçer, B., Raković, S., Kayacan, E., and Açikmeşe, B., "Model predictive control in aerospace systems: Current state and opportunities," *Journal of Guidance, Control, and Dynamics*, Vol. 40, 2017, pp. 1541–1566.

[27] Mayne, D., Rawlings, J., Rao, C., and Scokaert, P., "Constrained model predictive control: Stability and optimality," *Automatica*, Vol. 36, 2000, pp. 789–814.

[28] Rawlings, J., and Mayne, D., *Model Predictive Control: Theory and Design*, Nob Hill, Madison, WI, USA, 2009.

[29] Mayne, D., Seron, M., and Raković, S., "Robust model predictive control of constrained linear systems with bounded disturbances," *Automatica*, Vol. 41, No. 2, 2005, pp. 219–224.

[30] Deaconu, G., C, L., and A, T., "Minimizing the Effects of Navigation Uncertainties on the Spacecraft Rendezvous Precision," *Journal of Guidance, Control, and Dynamics*, Vol. 37, No. 2, 2014.

[31] Louembet, C., Arzelier, D., and Deaconu, G., "Robust Rendezvous Planning Under Maneuver Execution Errors," *Journal of Guidance, Control, and Dynamics*, Vol. 38, No. 1, 2015.

[32] Oestreich, C. E., Linares, R., and Gondhalekar, R., "Tube-Based Model Predictive Control with Uncertainty Identification for Autonomous Spacecraft Maneuvers," *Journal of Guidance, Control, and Dynamics*, 2022. https://doi.org/10.2514/1.G006438.

[33] "e.Deorbit," Tech. rep., European Space Agency, 2016. URL http://www.esa.int/OurActivities/SpaceEngineeringTechnology/CleanSpace/e.Deorbit.

[34] Fehse, W., *Automated Rendezvous and Docking of Spacecraft*, Cambridge Aerospace Series, Cambridge University Press, 2003.

[35] Bemporad, A., and Morari, M., *Robust model predictive control: A survey*, Springer, London, 1999, Vol. 245, pp. 207–226.

[36] Limon, D., Alvarado, I., Alamo, T., and Camacho, E., "On the design of Robust tube-based MPC for tracking," *17th IFAC Triennial Congress*, Seoul, South Korea, 2008.

[37] Kolmanovsky, I., and Gilbert, E., "Theory and Computation of Distrubance Invariant Sets for Discrete-Time Linear Systems," *Mathematical Problems in Engineering*, Vol. 4, 1998, pp. 317–367.

[38] Raković, S., Kerrigan, E., Kouramas, K., and Mayne, D., "Invariant approximations of the Minimal Robust Positively Invariant Set," *IEEE Transactions on Automatic Control*, Vol. 50, 2005, pp. 406–420.

[39] Alvarado, I., Limon, D., Alamo, T., Fiacchini, M., and Camacho, E., "Robust tube based MPC for tracking of piece-wise constant references," *46th IEEE Conference on Decision and Control*, New Orleans, LA, USA, 2007.

[40] Limon, D., Alvarado, I., Alamo, T., and Camacho, E., "MPC for tracking of piece-wise constant references for constrained linear systems," *16th IFAC World Congress*, Prague, Czech Republic, 2005.

[41] Kerrigan, E., "Robust constraint satisfaction: Invariant sets and predictive control," Ph.D. thesis, University of Cambridge, 2000.

[42] Gilbert, E., and Tan, K., "Linear systems with state and control constraints: The theory and application of maximal output admissible sets," *IEEE Transactions on Automatic Control*, Vol. 36, 1991, pp. 1008–1020.

[43] Setterfield, T. P., Miller, D. W., Saenz-Otero, A., Frazzoli, E., and Leonard, J. J., "Inertial Properties Estimation of a Passive On-orbit Object Using Polhode Analysis," *Journal of Guidance, Control, and Dynamics*, Vol. 41, No. 10, 2018, pp. 2214–2231.

[44] Huynh, D. Q., "Metrics for 3D Rotations: Comparison and Analysis," *Journal of Mathematical Imaging and Vision*, Vol. 35, 2009, pp. 155–164.

[45] Johnson, S. G., "The NLopt nonlinear-optimization package," , 2011. URL http://github.com/stevengj/nlopt.

[46] Smith, R., "Open Dynamics Engine (ODE)," , 2009. URL http://www.ode.org/.

[47] Rackwitz, F., "Efficient Trajectory Optimization for a Free-Floating Robot," Tech. rep., German Aerospace Center (DLR), 2021.

[48] Kuffner, J., "Effective sampling and distance metrics for 3D rigid body path planning," *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04. 2004*, Vol. 4, 2004, pp. 3993–3998 Vol.4.

[49] NVIDIA Corporation, "CUDA," , 2020. URL https://developer.nvidia.com/cuda-toolkit.

[50] Ericson, C., "Chapter 4 - Bounding Volumes," *Real-Time Collision Detection*, edited by C. Ericson, The Morgan Kaufmann Series in Interactive 3D Technology, Morgan Kaufmann, San Francisco, 2005, pp. 75–123.