Acta Astronautica 142 (2018) 170–183

Contents lists available at ScienceDirect

Acta Astronautica

journal homepage: www.elsevier.com/locate/actaastro

Real-time maneuver optimization of space-based robots in a dynamic environment: Theory and on-orbit experiments

Gregory E. Chamitoff [a,*,1], Alvar Saenz-Otero [b], Jacob G. Katz [b,2], Steve Ulrich [b,3], Benjamin J. Morrell [c], Peter W. Gibbens [c]

[a] Texas A&M University, College Station, TX, 77843, USA
[b] Massachusetts Institute of Technology, Cambridge, MA, 02139, USA
[c] University of Sydney, Sydney, New South Wales 2006, Australia

ARTICLE INFO

Keywords:
Space robotics
Trajectory optimization
Obstacle avoidance
International Space Station
Microgravity science
SPHERES

ABSTRACT

This paper presents the development of a real-time path-planning optimization approach to controlling the motion of space-based robots. The algorithm is capable of planning three dimensional trajectories for a robot to navigate within complex surroundings that include numerous static and dynamic obstacles, path constraints and performance limitations. The methodology employs a unique transformation that enables rapid generation of feasible solutions for complex geometries, making it suitable for application to real-time operations and dynamic environments. This strategy was implemented on the Synchronized Position Hold Engage Reorient Experimental Satellite (SPHERES) test-bed on the International Space Station (ISS), and experimental testing was conducted onboard the ISS during Expedition 17 by the first author. Lessons learned from the on-orbit tests were used to further refine the algorithm for future implementations.

1. Introduction

Real-time guidance and navigation for autonomous vehicles of all kinds is a challenging technical problem, but one that holds great promise for enabling major leaps in future capability. The next generation of autonomous robotic systems, such as self-guided aircraft and space-based robots will need to have the capability to analyze and interpret their own environment in order to design and conduct their mission [1]. Computing power and sensor technologies have advanced to the stage that it has become possible for an autonomous machine to gather data and construct an internal model of its surroundings fast enough to use that information for real-time decision making. Given a continuously evolving internal model, it is necessary to plan and replan an optimal approach for conducting a given maneuver, especially where dynamic obstacles are involved. To accomplish this, near real-time trajectory optimization techniques with not only terminal constraints, such as position and velocity, but interior-point constraints, representing static and/or dynamic obstacles, physical boundaries and performance restrictions, have been studied extensively (for instance [2–11]). Interior-point constraints, which can be mathematically represented in forms such as equality constraints, second order inequality constraints and logical constraints, are particularly important for many spacecraft proximity operations, where the reaction time available to cope with unknown or off-nominal situations is greatly reduced. However, interior-point constraints represent complex, nonlinear, non-convex optimization problems, which are challenging to solve in a time-efficient way [3–5,12,13].

To address this problem, this paper presents a novel trajectory optimization approach, referred to as the Admissible Subspace TRajectory Optimizer (ASTRO) algorithm, which overcomes these challenges by transforming the problem into a higher dimension parameter space that is well suited for real-time generation of feasible, near-optimal trajectories. The ASTRO algorithm is capable of planning three dimensional trajectories for space-based robots to navigate in close proximity to complex surroundings that include numerous static and dynamic obstacles, path constraints and performance restrictions.

* Corresponding author.
E-mail addresses: chamitoff@tamu.edu (G.E. Chamitoff), alvarso@mit.edu (A. Saenz-Otero), jgkatz@alum.mit.edu (J.G. Katz), Steve.Ulrich@carleton.ca (S. Ulrich), benjamin.morrell@sydney.edu.au (B.J. Morrell), peter.gibbens@sydney.edu.au (P.W. Gibbens).
[1] Former NASA Astronaut.
[2] Now at Space Exploration Technologies, Los Angeles, California.
[3] Now at Department of Mechanical and Aerospace Engineering, Carleton University, Ottawa, Canada.

https://doi.org/10.1016/j.actaastro.2017.10.001
Received 16 January 2016; Received in revised form 3 April 2017; Accepted 1 October 2017
Available online 28 October 2017

The core components of ASTRO were tested on the International Space Station (ISS) with the Synchronized Position, Hold, Engage, and Reorient Experimental Satellites (SPHERES) research facility, which consists of three autonomous robotic spacecraft maneuvering in a 6-degrees-of-freedom microgravity environment. To do so, a custom interface was designed to enable the ISS crew to use MATLAB directly on a Space-Station Support Computer to program and execute the algorithm, performing real-time commanding and telemetry communications with the SPHERES robots. ISS operational rules typically do not allow the crew to change the software of any system on the ISS directly. As such, the development of ASTRO aboard the ISS was itself a space first: the first time that an astronaut onboard the ISS was a Principle Investigator for the research being conducted, and the first ever use of MATLAB aboard the ISS. A series of test cases were conducted to demonstrate the ability of ASTRO to incorporate increasingly complex constraints while producing admissible (feasible) trajectories for a SPHERES robot to follow. Initially, all constraints were simulated virtually, but the robot actually flew physically inside the habitable volume of the US Laboratory on the ISS. Simplified validation tests were first performed with simulated static obstacles, after which a second SPHERES robot was used to present a real, dynamic obstacle. The second robot was programmed to get in the way of the first one, which was using the ASTRO algorithm for real-time path planning. Specifics on the implementation and the experimental results are presented in this paper.

The core contributions of this paper are: the formulation of ASTRO, the on-orbit implementation, testing and lessons learned.

The remainder of this paper is organized as follows. Section 2 provides an overview of existing approaches for solving complex, nonlinear trajectory optimization problems. Section 3 elaborates the main contributions of this work, and Sec. 4 details the theoretical development. Preliminary simulation results in Sec. 5 demonstrate the effectiveness of the proposed approach for a complex path planning problem. In Sec. 6, the on-orbit implementation and experimental results from the ISS are presented and discussed. Sec. 7 describes improvements to the approach driven by the results of the on-orbit tests, before conclusions in Sec. 8.

2. Background

There are a range of existing approaches for trajectory optimization that can be categorized into a number of areas. These include the indirect methods, which use calculus of variations techniques to *indirectly* solve the problem with Lagrange multipliers and Hamiltonians, and direct methods, which use approximation techniques to solve the complete optimization. Common direct methods include shooting and multiple shooting, collocation and the more recent evolutionary algorithms. Collocation methods are a widely used, and have a number of readily available tools (such as GPOPS on MATLAB [14], and SOCS from Boeing [15]). These methods represent the trajectory with a set of basis polynomials that are forced to match discrete points along the continuous trajectory. Evolutionary algorithms, such as genetic algorithms use "survival-of-the-fittest" concepts to generate the solutions, and are becoming more and more popular [16]. Refer to references [16–19] for a review of the different optimization methods and associated advantages and disadvantages. Across the variety of trajectory optimization methods, there are a range of techniques to deal with the challenge of interior point constraints and solving, in near real-time, the associated nonlinear, non-convex optimization problem.

Obstacle constraints are one type of interior point constraint that is very important in trajectory optimization for proximity operations. In particular, dynamic obstacle constraints are important for autonomous collision avoidance and coordinating the paths of multiple vehicles. One common approach to address the dynamic constraint problem is to represent obstacles as convex polygons, and use binary variables to turn inequality constraints for each side on and off to have only the closest sides to the path active [4,8,20–24]. The binary variables become part of the overall path optimization performed as a MILP, or a Mixed Integer Non-Linear Program (MINLP). Richard et al. [20] for example uses MILP to solve for a minimum-fuel path for microsatellites inspecting the outside of the ISS. Multiple vehicles can be included in the optimization, modeled as obstacles to each other. As more obstacles and more vehicles are introduced, the problem becomes increasingly complex, and computationally expensive to solve. While branch and bound algorithms [4,20] can be used to reduce the number of solutions that need to be calculated, the complexity of the optimization problem still makes it difficult to solve in real-time.

Blackmore [3], also uses branch and bound techniques but in a disjunctive convex program, where the overall non-convex problem is split into convex sub-problems. These sub-problems gradually build up to the complete constraint set, through branch and bound algorithms. Other techniques have been applied to similarly break up and simplify the problem, and then gradually build up the complexity, such as Lu and Liu [7] with Second-Order Cone Programming (SOCP). The work of Eren [25] is an example where the problem is modified with several different strategies to overcome each of the sources of non-convexity. Another example by Kobilarov [5] uses a technique called homotopy, where the initial solution is obstacle free, and then the problem is iteratively deformed to the final orientation by growing the size of obstacle constraints. These aforementioned simplification techniques, while effective, have only been demonstrated with static obstacles and a single vehicle.

Other branching techniques, such as genetic algorithms [26,27] and Rapidly expanding Random Trees (RRTs) [28] have been used to solve path planning problems with obstacles. Luo et al. [26] for example use spheres of safety and cones around lines of approach as constraints. Different branches, or generations of solutions are culled if they violate the obstacles. Generally these techniques have been demonstrated for static obstacles only, and plan the path of a single vehicle [26,28]. Extension of RRT methods such as RRT* [10] and MRRT-S [9,29] show impressive results for robotic arms in three dimensions with obstacle constraints [9,29,30], yet generally have longer computation times than what is practical for real time operation of a mobile robot in a dynamic environment.

Potential fields, as employed by Munoz and Fitz-Coy [31] have been used in a variety of forms to plan trajectories around obstacles [1,6,12,32–35]. The goal location is given an attractive potential and obstacles present repulsive potentials. Most of the techniques are limited to two dimensional problems with a static obstacle field. Xue [6] includes dynamic obstacles and planning for multiple vehicles, where an avoidance strategy is implemented on top of a potential field once an obstacle is within close proximity. Gyroscopic forcing in a potential field has been shown by Chang [35] to be effective to maintain separation between pairs of vehicles in a swarm of many vehicles. Both of these approaches generate trajectories for all vehicles together in the one optimization routine. This makes the problem quite large, and hence difficult to solve quickly.

A different approach to overcoming the non-convexities of path planning around obstacles is to use a collocation approach, and include consideration of collision avoidance in the cost function for the resulting parameter optimization problem. It is in this category that the approach employed by ASTRO can be viewed. ASTRO uses Legendre polynomials to represent the trajectory which enables a transformation of a complex cost function into a simple, convex form and to enable rapid computation of the trajectory integrals and derivatives. Obstacle constraints are considered with penalty functions in the parameter optimization of the transformed problem. Such an approach can plan three dimensional trajectories with static and dynamic constraints in a manner that has potential for real-time operation.

Since the early development of ASTRO in 1994 [36], there has been an emerging field of pseudospectral trajectory optimization which employs similar techniques to ASTRO by representing the trajectory with Legendre polynomials and their derivatives [37]. This class of methods was first described by Elnagar in 1995 [38] and was expanded by Ross and Fahroo [39]. It has since been shown to be effective for a range of

complex problems [16,18,19]. The pseudospectral approach has been demonstrated for operation in three dimensions with obstacle constraints [11,40,41], and with potential for real-time operation [11], yet without considering dynamic obstacles. Hurni et al. [2] did demonstrate real time operation with dynamic obstacles, but for a two dimensional problem. The results demonstrated by these authors came after the investigations described in this paper. In retrospect, ASTRO could now be viewed as being in the class of pseudospectral trajectory optimization, but as a method particularly focused on solving the combined problem of real-time, three dimensional trajectory optimization with consideration of dynamic obstacles.

3. Primary contributions

The techniques described above cover a range of methods to manage obstacle constraints in trajectory optimization problems. There are no standout techniques that achieve all three goals of:

1. Planning three dimensional trajectories;
2. Considering dynamic obstacles; and
3. Computing in a manner that has potential for real-time operation.

Several techniques can address two of the three goals, but require further developments to achieve the third. To achieve all three goals is the aim of the development of ASTRO.

In this context, the primary contribution of this work is the development and then on-orbit testing of ASTRO. In the development of ASTRO, there are two core innovative components. (1) The formulation of the problem to use Legendre polynomials to transform a complex, non-linear cost function into a simpler form that can be quickly computed. (2) The use of a subspace projection technique to ensure boundary conditions are always met throughout an iterative optimization, enabling rapid generation of collision free trajectories – an important strength of the technique.

For the on-orbit testing of ASTRO, the key contributions are (1) the development of the architecture to implement and test the algorithm on the SPHERES free flying robotic test facility onboard the ISS, (2) the validation of core capabilities of the algorithm and (3) the lessons learned from the experiments. The final contributions of this paper are the refinements to the ASTRO algorithm addressing the lessons learned from the ISS tests to improve computation time and the handling of dynamic obstacles.

4. ASTRO algorithm

The ASTRO algorithm effectively solves a guidance problem, while an inner-loop controller handles the task of trajectory tracking. To ensure that the trajectory can be followed, the maneuvering capabilities of the vehicle are incorporated as acceleration and velocity constraints within the optimization. In other words, the solution is not allowed to ask for maneuvers that the vehicle cannot achieve given its mass properties and the limitations of its actuators (e.g. thrusters). The optimization strategy is based on a two-stage approach. The first stage is designed to identify admissible trajectories as quickly as possible. These trajectories meet the initial and final boundary conditions while avoiding all spatial (obstacles or boundaries) and performance constraints. The second stage uses any remaining computation time to refine the solution towards an optimal trajectory.[4]

The optimization itself is accomplished by using an augmented cost function that penalizes a combination of path length and active constraint violations. The flight path velocity is parameterized using Legendre polynomials. Initial parameter estimates are obtained by performing a low-order spline to meet the boundary conditions of a relaxed problem (i.e. without constraints). A gradient descent optimization approach can then be used to search for an optimal trajectory. A special feature of the ASTRO algorithm is its ability to perform the parameter optimization by projecting numerical gradient information onto the sub-space of parametric variations that enforce the boundary conditions. As such, each iteration meets the boundary conditions perfectly, and subsequent iterations move closer to a solution that satisfies the other constraints as well. Relative weights in the cost function are used to assure that constraint violation gradients dominate the parameter search while any constraints remain active. These properties of the algorithm enable rapid convergence to an admissible solution, and then to an optimal or sub-optimal (local minima) solution.

4.1. Problem statement

The general guidance problem addressed by the ASTRO algorithm is that of generating an optimal flight path for an autonomous space-based robot through its environment in order to accomplish some task. It must start with the robot's current position and velocity and end up at some final desired state. Along the way it must avoid geometric constraints (physical objects), remain within restricted volumes and maneuver within the performance limitations of its actuators (thrusters etc.). This problem is illustrated in Fig. 1.

The definition of optimality in this case is taken to be a combination of: minimizing the path length between the initial position at the initial time $\mathbf{x}(t_0)$ and the final position at the final time $\mathbf{x}(t_f)$, and minimizing the variations in velocity along the path. This definition of optimality gives the shortest path that has sufficiently smooth changes in velocity (reducing the accelerations that need to be applied by the actuators and hence reducing the cost of actuation). The definition of optimality is chosen to facilitate the transformations detailed in this paper. The initial and final velocity vectors are specified as $\dot{\mathbf{x}}(t_0)$ and $\dot{\mathbf{x}}(t_f)$. The boundary conditions can be expressed as follows

$$\begin{aligned} f_{BC_1} &= (\mathbf{x}(t_0), \dot{\mathbf{x}}(t_0)) = 0 \\ f_{BC_2} &= \left(\mathbf{x}(t_f), \dot{\mathbf{x}}(t_f)\right) = 0 \end{aligned} \tag{1}$$

The cost function to be minimized is the integral of the velocity squared over time, which can be viewed as minimizing the integral of the velocity magnitude (path length) with additional penalties on large *variance* in the velocity. The cost function is expressed by

$$\overline{S}^2 = \int_{t_0}^{t_f} \left(\dot{x}(t)^2 + \dot{y}(t)^2 + \dot{z}(t)^2\right) \mathrm{d}t \tag{2}$$

and the functions $x(t)$, $y(t)$ and $z(t)$, (i.e. the path specified by the vector $\mathbf{x}(t)$) are to be determined by the algorithm. This is complicated by the environmental obstacles, and these can be described by functions of the same variables. As a simple example, the definition of a wall could be $x < 5$, meaning that if $x = 5$ the robot has hit the wall. If the trajectory $\mathbf{x}(t)$ includes portions for which $x \geq 5$, then that amounts to a constraint violation. An admissible solution is one that meets the initial and final boundary conditions at $\mathbf{x}(t_0)$ and $\mathbf{x}(t_f)$ and does not violate any of the geometric constraints along the trajectory. In addition there are performance constraints on the ability of the robot to maneuver, and these can be expressed as functions of the higher derivatives of the trajectory. For another simple example, limiting the required acceleration along one axis could be described by $\ddot{x}(t) < 100$, meaning that an achievable acceleration due to thruster magnitude and mass properties is less than 100 units. As with the geometric constraints, an admissible solution for the trajectory and its higher derivatives must not violate these performance constraints. In general, the geometric and performance constraints can be described by

$$f_{c_i}(\mathbf{x}(t), \dot{\mathbf{x}}(t), \ddot{\mathbf{x}}(t)) \leq 0, \quad \forall i \in [1, n], \quad \forall t \in \left[t_0, t_f\right] \tag{3}$$

[4] The solution trajectory might be sub-optimal, being caught in a local minima, due to the inherently non-convex nature of planning trajectories with obstacles.

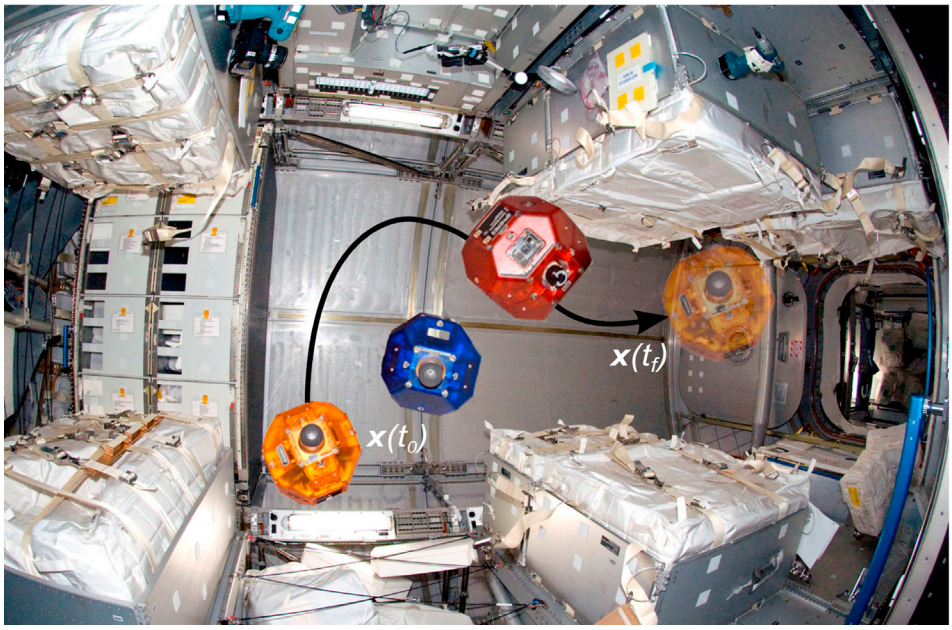

Fig. 1. The ASTRO algorithm optimizes the path from $\mathbf{x}(t_0)$ to $\mathbf{x}(t_f)$.

where n is the number of constraint functions. Note that the ASTRO algorithm is designed to find an admissible trajectory – a solution that satisfies Eqs. (1) and (3), very quickly, and then spends any remaining computation time to optimize further by minimizing Eq. (2). This two-stage search is achieved by constructing an augmented cost function of the form

$$J = f_s\left(\overline{S}^2\right) + \sum_{j=1}^{n} K_j \max_{t\in[t_0,t_f]} f_{c_j}^2 \tag{4}$$

where $f_s(\overline{S}^2)$ is the path length cost function from Eq. (2). The coefficients K_j represent the relative weights W_j for each constraint function, and are given by

$$K_j = \begin{cases} 0, & \text{if } f_{c_j} \le 0 \\ W_j, & \text{if } f_{c_j} > 0 \end{cases} \tag{5}$$

These values only need to be chosen to assure that the constraint violations dominate the cost function when the constraints are active. Once a solution is found that drives the second term in Eq. (4) to zero, then the resulting trajectory is admissible.

4.2. Legendre polynomial parameterization

In general, Eq. (4) represents a difficult and nonlinear optimization problem. The ASTRO algorithm transforms this into a simpler problem by parameterizing the velocity by Legendre polynomials and normalizing the time interval such that certain orthogonality properties of Legendre polynomials can be used to its advantage. The parameterization is expressed as follows

$$\dot{x}_i(t') = \sum_{k=0}^{N} C_{ik} P_k(t') \tag{6}$$

where

$$t' = 2\left[\frac{t - t_0}{t_f - t_0}\right] - 1 \tag{7}$$

The C_{ik} terms are coefficients of the Legendre polynomials of order k that represent the velocity trajectory $\dot{x}_i(t)$. Time is normalized such that $t' \in [-1, 1]$. The trajectory $x_i(t)$ is represented by a sum of the indefinite integrals of the Legendre polynomials, where the constant of integration is set so that each polynomial is equal to zero at $t = \pm 1$. The position trajectory is then expressed as:

$$x_i(t') = \left(\frac{t_f - t_0}{2}\right) \sum_{k=0}^{N} C_{ik} \int P_k(t')\mathrm{d}t' \tag{8}$$

Over the normalized time interval, and taking advantage of the orthogonality of Legendre polynomials, the cost function in Eq. (4) can be reduced to the following

$$\overline{S}^2 = \sum_{i=1}^{3} \sum_{k=0}^{N} \left\{ C_{ik}^2 \int_{-1}^{1} [P_k(t')]^2 \mathrm{d}t' \right\} \tag{9}$$

In Eq. (9), P_k are the standard Legendre polynomials and therefore the integral can be evaluated off-line. The result is that the cost function is reduced to a sum of parameters weighted by fixed values. Due to the orthogonality of Legendre polynomials, all of the cross terms are zero, and this reduces the computation by a factor of N, which is the minimum order of the polynomials required to find a solution [36]. This simplification of the cost function is a key component of the algorithm that enables rapid computation.

Since the flight path, velocity along that path, and all geometric and performance constraints can be expressed as functions of $\mathbf{x}(t)$ and higher derivatives, referring to Eqs. (6) and (9), the full cost function in Eq. (4) can be written as

$$J = \sum_{j=1}^{n+1} \left[f_j(\mathbf{C})\right]^2 \tag{10}$$

where the rows of $\mathbf{C} = \begin{bmatrix} C_{11} & \cdots & C_{1n} \\ \vdots & \ddots & \vdots \\ C_{31} & \cdots & C_{3n} \end{bmatrix}$ correspond to coefficients of the Legendre polynomials that parameterize the curves for $\mathbf{x}_i(t)$, and the f_j represent the maximum violation along a given trajectory for each constraint function. The $n+1$ term represents the path length cost term $f_{n+1}(\mathbf{C}) = f_s(\overline{S}^2)$.

4.3. Formulation as a convex search space

The advantage of transforming the cost function from Eq. (4) into Eq. (10) is that it can be shown that the optimization of path length can now be performed on a convex space using a simple gradient-descent procedure. In the case where obstacle constraints are included, the convexity of the complete problem can not be maintained, yet the formulation aids in the optimization of the solution once the constraints are satisfied. In Eq. (10), J is a convex (positive-definite) function of $f_j(\mathbf{C})$, but it is not, in general, convex with respect to the parameters C_{ik} or to the parameter errors $(C_{ik} - C^*_{ik})$, where C^*_{ik} are the optimal values of the coefficients for the optimal trajectory.

If the cost function could be restricted, however, such that J is positive definite with respect to $(C_{ik} - C^*_{ik})$, then a gradient $(\partial J/\partial C_{ik})$ based search algorithm could be made to converge to the optimal solution. This can be seen as follows.

First, recall the definition of a positive-definite function: $V(x)$ is positive definite if $V(0) = 0, V(x) > 0$ for $x \neq 0$, and $V(x) > g(|x|)$ where $g(\cdot)$ is a non-decreasing function.

Second, define $\Delta C_{ik} = C_{ik} - C^*_{ik}$, and note that $(\partial J/\partial C_{ik}) = (\partial J/\Delta C^*_{ik})$ since $(\partial J/\partial C^*_{ik}) = 0$. Now assume that $J(\Delta C_{ik})$ is a positive-definite function of ΔC_{ik}, and the problem is to search over the space defined by C_{ik} for the optimal parameters C^*_{ik}. Since $J = 0$ only for $\Delta C_{ik} = 0$, and $J > 0$ for all other values of ΔC_{ik}, there is a unique minimum for J. Likewise, $\partial J/(\partial|\Delta C_{ik}|) \geq 0$ by the definition above, and therefore $\partial J/\partial \Delta C_{ik} = 0$ only at $\Delta C_{ik} = 0$, or at other local inflection points in the parameter space. Local minima, however, would require that $\partial J/(\partial|\Delta C_{ik}|) < 0$ over some interval between that point and C^*_{ik}, which violates the definition of positive definite. Therefore, if $J(\Delta C_{ik})$ is positive-definite, $\Delta C_{ik} = 0$ is the only extremal point, and any other point has a negative gradient $-\partial J/(\partial C_{ik})$ pointing in the direction of the global minimum (except when $-\partial J/(\partial C_{ik}) = 0$, which can be handled by a minimum step-size).

To make the cost function Eq. (10) positive definite in the parameter errors $(C_{ik} - C^*_{ik})$, consider the revised function

$$J' = J - J^* = \sum_{j=1}^{n+1} \left[f_j(\mathbf{C})\right]^2 - \sum_{j=1}^{n+1} \left[f_j(\mathbf{C}^*)\right]^2 \tag{11}$$

J^* is the cost for the optimal solution. Note that J' is not a realizable cost function, since J^* is unknown, but it simply differs from the original cost by a constant scalar. Now consider the positive-definite criteria for J' with respect to $(C_{ik} - C^*_{ik})$. Since $J' = 0$ when $C_{ik} = C^*_{ik}$, and $J' > 0$ when $C_{ik} \neq C^*_{ik}$, we have that J' is positive definite in $(C_{ik} - C^*_{ik})$ if $\partial J'/\partial|\Delta C_{ik}| \geq 0$. This condition is equivalent to saying that $\partial J'/\partial C_{ik}$ must be non-decreasing (i.e. monotonic). Therefore, in terms of Eq. (11), J' is positive definite with respect to $(C_{ik} - C^*_{ik})$ if the following condition is satisfied:

$$\frac{\partial J'}{\partial C_{ik}} = 2\sum_{j=1}^{n+1} f_j(C_{ik}) \frac{\partial f_j}{\partial C_{ik}} \text{ismonotonic} \tag{12}$$

equivalently:

$$\frac{\partial^2 J'}{\partial C_{ik}^2} = 2\sum_{j=1}^{n+1} \left(\frac{\partial f_j}{\partial C_{ik}}\right)^2 + f_j(C_{ik}) \frac{\partial^2 f_j}{\partial C_{ik}^2} \geq 0 \tag{13}$$

Assuming these conditions on $f_j(C_{ik})$ given by Eq. (13) are satisfied, then the following are true:

1. J' is a positive-definite function with respect to the parameter errors $(C_{ik} - C^*_{ik})$;
2. The point $C_{ik} = C^*_{ik}$ is a unique minimum for J';
3. For any $C_{ik} \neq C^*_{ik}$, J' can be reduced by a discrete step in C_{ik}, that is

$$[C_{ik}]_{new} = [C_{ik}]_{old} + \delta C_{ik} \tag{14}$$

where

$$\delta C_{ik} = -\alpha \left[\frac{\partial J'}{\partial C_{ik}}\right] + \beta_{ik} \tag{15}$$

with $\alpha > 0$ and $\beta_{ik} \neq 0$; and

4. The pseudo-gradient search algorithm above will converge to C^*_{ik}.

Now, since J' and J only differ by a constant, a procedure which minimizes J' also minimizes J. Thus, provided the conditions on $f_j(C_{ik})$ are met, the optimization of $J(C_{ik})$ can be accomplished by the same method.

The restrictions on the form of $f_j(C_{ik})$, which apply to the path length penalty $f_{n+1}(\mathbf{C}) = f_s(\bar{S}^2)$ and to any constraint functions $f_{c_j}(\mathbf{x}(t), \dot{\mathbf{x}}(t), \ddot{\mathbf{x}}(t))$ can be easily satisfied for the path length cost function, performance constraints and physical boundary constraints, but not for obstacle constraints. In particular, the path length in Eq. (9), can be simplified as follows. Define the path length function f_s as

$$f_s\left(\bar{S}^2\right) = \sum_{i=1}^{3} \sum_{k=0}^{N} I_k C_{ik}^2 \tag{16}$$

where

$$I_k = \int_{-1}^{1} [P_k(t')]^2 \mathrm{d}t' \tag{17}$$

and now apply the condition defined in Eq. (13). The result is

$$4\left(\sum_{i=1}^{3} \sum_{k=0}^{N} I_k C_{ik}\right)^2 + \left(\sum_{i=1}^{3} \sum_{k=0}^{N} I_k\right)\left(\sum_{i=1}^{3} \sum_{k=0}^{N} I_k C_{ik}^2\right) \geq 0 \tag{18}$$

which is clearly satisfied since all summation terms on the left are positive. For the constraint functions, most physical or performance related constraints that would be of interest can be expressed generally by

$$g(\mathbf{X}) \leq g_0 \tag{19}$$

or

$$f_c(\mathbf{X}) = \begin{cases} [g(\mathbf{X}) - g_0]^2, & \text{if } g(\mathbf{X}) > g_0 \\ 0, & \text{if } g(\mathbf{X}) \leq g_0 \end{cases} \tag{20}$$

where $\mathbf{x}_p$, $\dot{\mathbf{x}}_p$, and $\ddot{\mathbf{x}}_p$ are constants and where $\mathbf{X}(t)$ is given by

$$\mathbf{X}(t) = \begin{bmatrix} \mathbf{x}(t) - \mathbf{x}_p \\ \dot{\mathbf{x}}(t) - \dot{\mathbf{x}}_p \\ \ddot{\mathbf{x}}(t) - \mathbf{x}_p \end{bmatrix} \tag{21}$$

For instance, a spherical obstacle constraint located at $\mathbf{x}_p$ with radius r_{sphere} is represented by:

$$g(\mathbf{X}) - g_0 = -\left|\left|\left(\mathbf{x}(t) - \mathbf{x}_p\right)\right|\right| + r_{sphere} \tag{22}$$

and a limit on acceleration of a_{lim} is represented by:

$$g(\mathbf{X}) - g_0 = \left|\left|\left(\ddot{\mathbf{x}}(t) - \ddot{\mathbf{x}}_p\right)\right|\right| - a_{lim} \tag{23}$$

To maintain the convexity of the cost function, the constraint functions $g(\mathbf{X})$ need to be convex ($\partial^2 g/\partial \mathbf{X}^2 \geq 0$) [42]. While the acceleration constraints are convex, the obstacle constraints are not. Hence for a complete problem with obstacle constraints the cost function is not convex, and the algorithm is susceptible to being caught in local minima.

Yet the strength of the formulation is in facilitating rapid generation of feasible solutions that avoid the obstacles, after which the convexity of the path length cost can be used to quickly move to the nearest sub-optimal solution (local minima).

4.4. Boundary conditions and projected gradient search

With the cost function defined by Eq. (10) a gradient descent optimization method can be employed to follow the negative gradient of J with respect to the parameters C_{ik} ($-\partial J/\partial C_{ik}$) to direct towards a feasible, and then optimal trajectory. To generate admissible sub-optimal solutions as quickly as possible, a gradient projection is used to assure that the boundary conditions at t_0 and t_f are always met. In fact, these equality constraints are very useful in providing an initial *guess* for the trajectory parameters, and, more importantly, they can be used to reduce the degrees of freedom in the search space.

From the trajectory parameterization $\mathbf{x}_i(t') = \sum_{k=0}^{N} C_{ik} \int P_k(t') \mathrm{d}t$ and velocity parameterization $\dot{x}_i(t') = \sum_{k=0}^{N} C_{ik} P_k(t')$, the boundary conditions can be written as follows

$$\mathbf{X}_{BC} = \mathbf{P}_{BC}\mathbf{C} \tag{24}$$

or

$$\begin{bmatrix} x(t_0) & y(t_0) & z(t_0) \\ \dot{x}(t_0) & \dot{y}(t_0) & \dot{z}(t_0) \\ x(t_f) & y(t_f) & z(t_f) \\ \dot{x}(t_f) & \dot{y}(t_f) & \dot{z}(t_f) \end{bmatrix} = \begin{bmatrix} \int P_1(-1)\mathrm{d}t & \int P_2(-1)\mathrm{d}t & \cdots & \int P_N(-1)\mathrm{d}t \\ P_1(-1) & P_2(-1) & \cdots & P_N(-1) \\ \int P_1(1)\mathrm{d}t & \int P_2(1)\mathrm{d}t & \cdots & \int P_N(1)\mathrm{d}t \\ P_1(1) & P_2(1) & \cdots & P_N(1) \end{bmatrix} \mathbf{C} \tag{25}$$

In this equation $\mathbf{X}_{BC}$ is the matrix of boundary conditions, $\mathbf{P}_{BC}$ is a directly computable function of the Legendre polynomials, and $\mathbf{C}$ represents the matrix of coefficients to be optimized for the trajectory. There are 12 equations and $3N$ unknowns (i.e., the elements of $\mathbf{C}$). If the polynomials are limited to 4th order, then the coefficients of $\mathbf{C}$ are uniquely determined. This represents (exactly) the cubic-spline curve between the initial and final positions and velocities, which is an ideal initial guess for the trajectory. So $\mathbf{C}_{start} = \mathbf{P}_{BC}^{-1}\mathbf{X}_{BC}$ for $N = 4$.

From this initial guess it is possible to assure that all future optimization steps maintain the boundary conditions by projecting the gradient in such a way that it does not span the space of the 12 degrees of freedom already determined by the boundary conditions. For $N > 4$, there are $(3N - 12)$ additional degrees of freedom over which to search in order to avoid constraints and minimize the cost function. This is accomplished by partitioning the $\mathbf{C}$ matrix as $\mathbf{C}_{N\times 3} = \mathbf{C}_{\perp_{N\times 3}} + \mathbf{C}_{||_{N\times 3}}$, such that

$$\mathbf{X}_{BC} = \mathbf{P}_{BC}\mathbf{C}_{N\times 3} = \mathbf{P}_{BC}\left\{\mathbf{C}_{\perp_{N\times 3}} + \mathbf{C}_{||_{N\times 3}}\right\} \tag{26}$$

where

$$\mathbf{P}_{BC}\mathbf{C}_{\perp_{N\times 3}} = 0 \tag{27}$$

In other words, the columns of $\mathbf{C}_{||}$ are in the row-space of $\mathbf{P}_{BC}$, and the columns of $\mathbf{C}_{\perp}$ are in the null-space of $\mathbf{P}_{BC}$. Any variation of $\mathbf{C}_{\perp}$ that maintains $\mathbf{P}_{BC}\mathbf{C}_{\perp_{N\times 3}} = 0$ will assure that the boundary conditions are not disturbed. Given an arbitrary gradient step in the matrix $\mathbf{C}$, given by $\delta\mathbf{C} = \delta\mathbf{C}_{||} + \delta\mathbf{C}_{\perp}$, the desired component of that step is $\delta\mathbf{C}_{\perp}$, since it won't change the boundary conditions. Using the fact that $\mathbf{P}_{BC}\delta\mathbf{C} = \mathbf{P}_{BC}\delta\mathbf{C}_{\perp} + \mathbf{P}_{BC}\delta\mathbf{C}_{||} = \mathbf{P}_{BC}\delta\mathbf{C}_{||}$, the projected gradient step is

$$\delta\mathbf{C}_{\perp} = \left[\mathbf{I} - \mathbf{P}_{BC}^{T}\left(\mathbf{P}_{BC}\mathbf{P}_{BC}^{T}\right)^{-1}\mathbf{P}_{BC}\right]\delta\mathbf{C} \tag{28}$$

where

$$\delta\mathbf{C} = -\alpha[\partial J/\partial\mathbf{C}] + \beta \tag{29}$$

and the parameter adjustments are now limited to the $3N - 12$ degrees of freedom remaining. $\delta\mathbf{C}$ is the gradient step from any generalized solver, with the step size α and a non-zero constant β. $\delta\mathbf{C}_{\perp}$ amounts to a projection of the negative gradient of J onto the sub-space of parametric variations that enforce the boundary conditions. This concept is a key feature of the ASTRO algorithm and is illustrated in Fig. 2.

By parameterizing the trajectory with Legendre polynomials, starting the optimization with an initial guess that meets the boundary conditions, and enforcing those conditions by a projected gradient, the ASTRO algorithm is capable of quickly finding sub-optimal solutions that meet all constraints. By searching a limited subspace with reduced degrees of freedom, it can then converge on optimal or sub-optimal (local minima) solutions very quickly as well.

5. Preliminary simulation results

To demonstrate the capabilities of the ASTRO path-planning algorithm a simple scenario is shown that involves planning a trajectory through a field of stationary geometrical obstacles between initial and final boundary conditions defined by:

$$\begin{aligned} \mathrm{x}(t_0) &= [0 \quad -0.5 \quad 0]\mathrm{m} \\ \dot{\mathbf{x}}(t_0) &= [0 \quad 0 \quad 0]\mathrm{m/s} \\ \mathrm{x}(t_f) &= [0 \quad 0.5 \quad 0]\mathrm{m} \\ \dot{\mathbf{x}}(t_f) &= [0 \quad 0 \quad 0]\mathrm{m/s} \end{aligned}$$

The trajectory is set to be completed in 100 s, i.e., $t_f = 100$ s. Four external cylindrical constraints with radius of 0.05 m, joined at 90° and centered at the origin, form a 0.26 m × 0.26 m × 0.1 m enclosed cuboid obstacle with a small opening in the middle that the initial guess trajectory can pass through. Two spherical constraints with radius of 0.1 m centered at −0.2 m and 0.2 m along the y-axis provide additional obstacles that must be avoided to find an admissible solution. Each of these bodies are obstacle constraints and hence are constraints of the type that do not satisfy the requirements for a convex search space. The selection of obstacles gives a complex, non-convex problem to solve.

As shown in Fig. 3, the initial guess is a straight line between two points and passes through both spherical constraints as well as the enclosed (but admissible) region between the four cylinders. In the dimensions of the problem, the trajectory appears to be trapped. Any perturbations to the trajectory would apparently pass through the walls of the cylinders. Yet with the weighting on the obstacle constraints making them dominate the cost function, the early update steps produce trajectories that jump outside the obstacles to a feasible solution, which is then optimized to produce one of the many equivalent sub-optimal, local minima solutions.

6. ISS on-orbit experimental results

The ASTRO algorithm has been experimentally tested on the

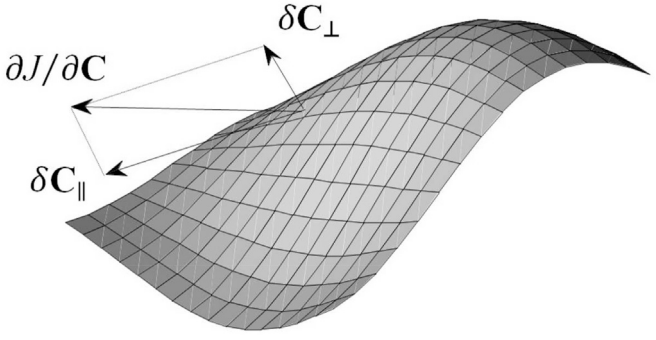

Fig. 2. Gradient projection algorithm - reduces search space and assures boundary conditions.

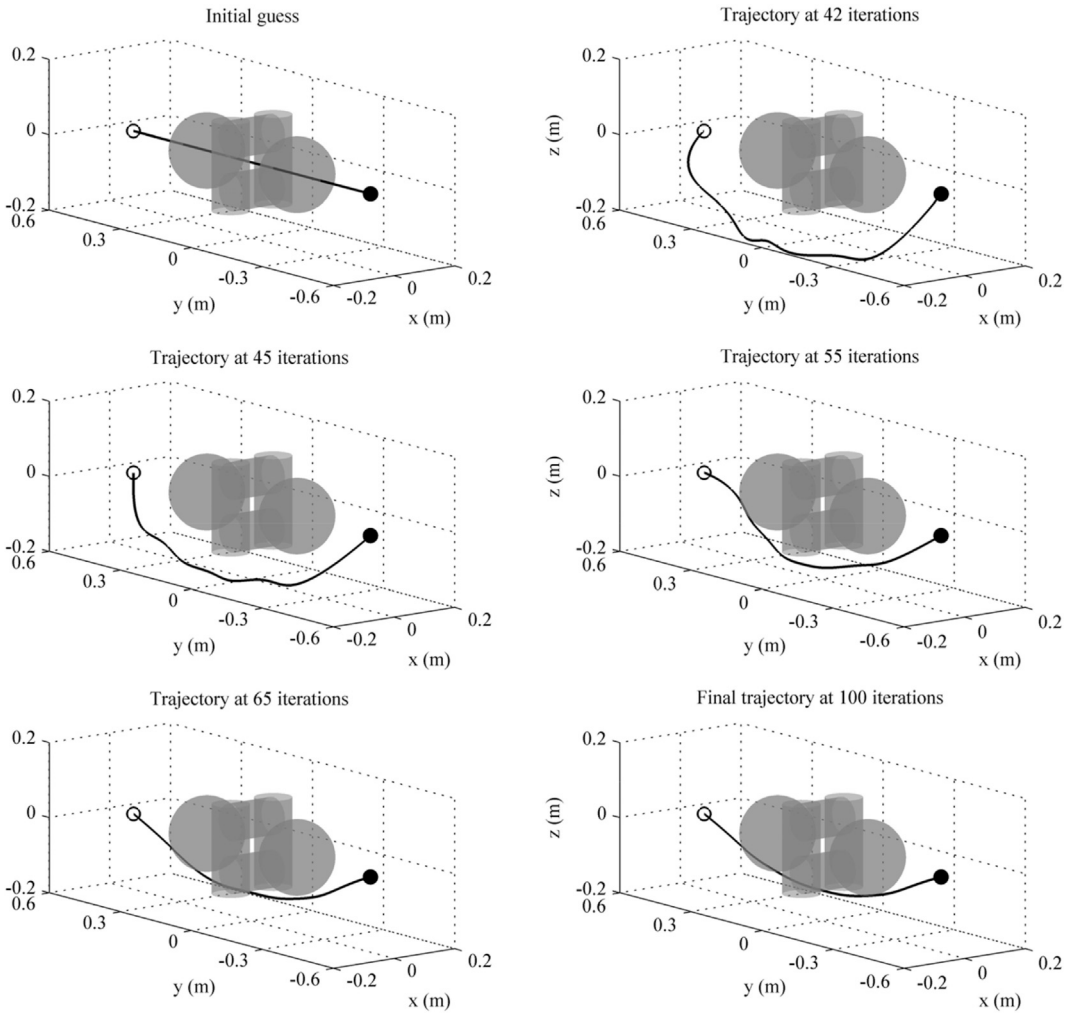

Fig. 3. Successive steps of the path-planning algorithm from initial guess to final solution.

SPHERES (Synchronized Position Hold, Engage, Reorient Experimental Satellites) [43] research facility aboard the International Space Station. SPHERES is an experimental test-bed that has been used to test a wide range of algorithms for applications such as formation-flying of satellites [44,45], attitude control and flight path control in the presence of uncertainties [46] to name a few. Future versions of space-based robots like these will one day perform independent work inside and outside space vehicles, enable complex formation flight missions, and may perform routine, hazardous or difficult tasks that are not feasible or efficient for humans to perform. The SPHERES platform is ideal for investigating the technical approach for many future robotic applications in a 6 DOF microgravity environment.

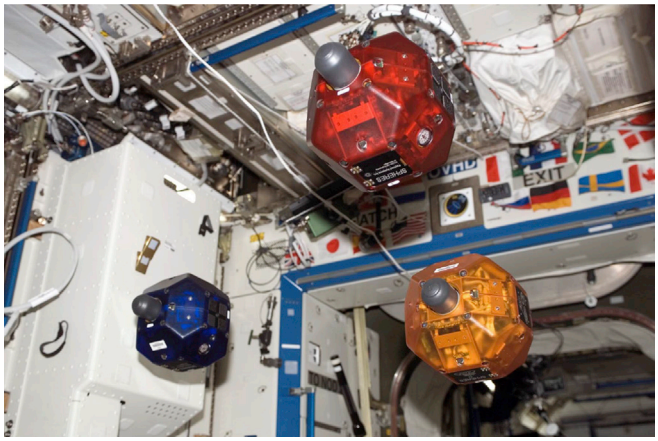

Fig. 4. Picture of three SPHERES performing a test onboard the ISS. (Photograph credit: NASA-SPHERES).

6.1. Experimental setup

The SPHERES facility operates three robotic spacecraft aboard the International Space Station. Each robot resembles a complete satellite bus with cold-gas propulsion, power, communications, processing, and metrology capabilities. The propulsion systems utilize compressed CO_2 gas with on/off thrusters. The metrology system is based on a 6DOF IMU and a pseudo-GPS system which uses ultrasound time of flight measurements. An Extended Kalman Filter (EKF) combines the inertial body measurements and the global ultrasound measurements to obtain a full 6DOF estimate of the robots within their operating volume inside the ISS [47,48]. The communication system uses a single TDMA (Timing Division Multiple Access) based RF channel to communicate its state to neighboring robots; the timing of the TDMA is controlled by the user interface (GUI) which operates on an ISS laptop. Fig. 4 shows a picture of three SPHERES robots aboard the ISS. The dynamics of each robot are well approximated by a double integrator model [49].

ASTRO research used the SPHERES Engineering GUI (instead of the

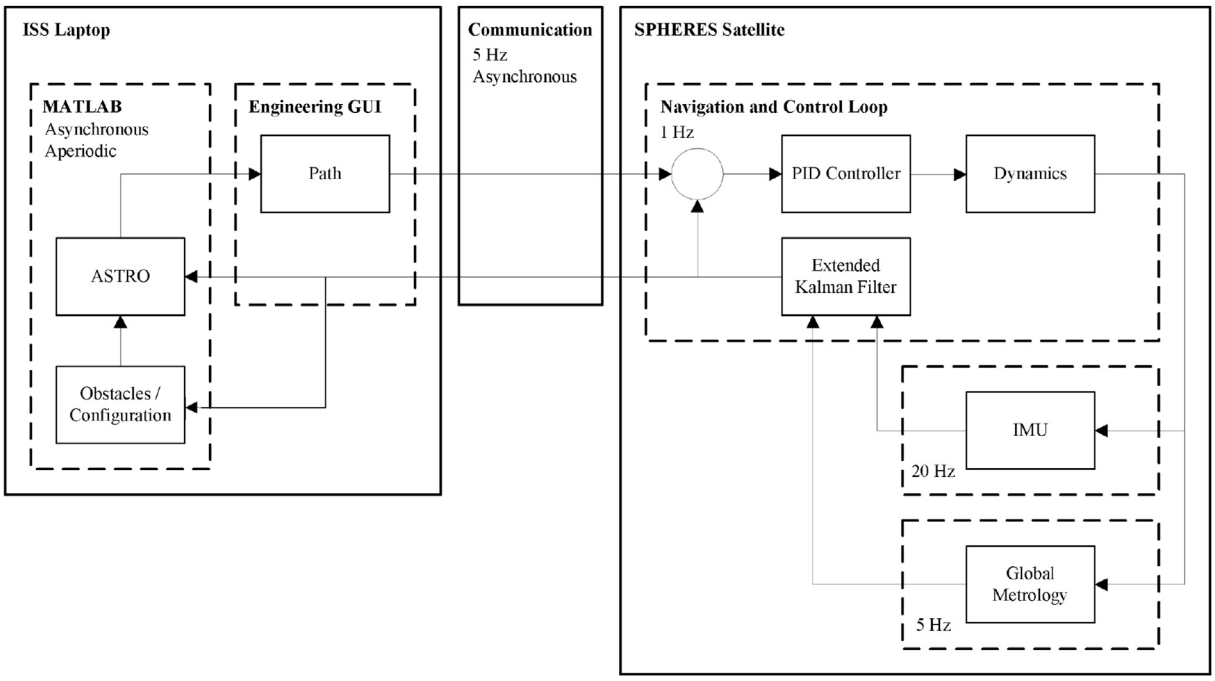

Fig. 5. SPHERES ASTRO implementation system overview.

GUI regularly used aboard ISS[5]). The Engineering GUI provides real-time state and debug information to scientists. For this test session the scientist was aboard the ISS. The SPHERES Engineering GUI provides real-time information about: (1) 13-element state vector (position, velocity, quaternion, and angular velocity), (2) three types of *debug* vectors (all numerical), (3) detailed SPHERES status information, and (4) the ability to see raw communication packets.

The state and debug information were essential to enable successful research aboard the ISS. Yet, they were not sufficient for these tests, since it is not possible to interpret the ASTRO paths when presented solely through numerical data in debug vectors. Further, many of the obstacles and constraints were virtual elements, not physically present aboard the ISS. Therefore, the GUI functionality had to be augmented to provide tools which visualized the output of ASTRO, the obstacles, and the paths followed by the SPHERES robots.

Traditional algorithm development with SPHERES is programmed in C via the Texas Instruments Code Composer Studio IDE. This IDE provides the interfaces to the C6701 DSP used aboard the SPHERES robots [50]. The interfaces include access to real-time threads for estimation, control, and autonomy. However, the DSP has limited processing capabilities (it operates at 167 MHz, has 16 MB of RAM, and 256k of available FLASH). In addition, the IDE does not provide any standard mathematical routines which would enable fast reconfiguration of algorithms. Further, the C++ development environment did not provide the desired visualization routines to enable real-time evaluation of the performance of the ASTRO algorithm. Due to the limitations of the DSP development environment, the SPHERES team developed new tools for algorithm programming aboard the ISS, and allowing the use of MATLAB through a real-time distributed-computing environment. The development of this tool is an important contribution from the work presented.

The SPHERES Engineering GUI was modified to enable real-time communications with MATLAB to: (1) provide multiple visualization tools, (2) allow for easy modification of the script files that implement the algorithms and, (3) provide substantial mathematical tools to simplify development. For the purposes of ASTRO, the SPHERES Engineering GUI was simply a communications link between MATLAB and the SPHERES robots. The calls to MATLAB were done via multi-threaded calls, allowing the GUI to maintain the 5 Hz TDMA communications link while MATLAB ran the ASTRO algorithm. This allowed MATLAB to have longer periods of processing time.

6.2. *Real-time traffic control model for distributed robotic systems*

The resulting distributed computing system created a multi-frequency environment where the SPHERES robots implemented a periodic real-time path-following closed-loop control algorithm and MATLAB was used to implement a semi-periodic near-real-time path planning algorithm. This architecture is presented in Fig. 5. The MATLAB thread performed the same sequence of operations: (1) collect all data queued since the previous run, (2) trigger the ASTRO planning task, (3) show visualization data, and (4) send the new path target points to the robot. The data are transmitted by the GUI on the next SPHERES TDMA communications cycle. In parallel, the SPHERES robots and GUI implemented a 1 Hz PID control loop which had been proven to work in prior SPHERES tests aboard the ISS. As with all SPHERES tests, the robots provided their own estimation utilizing an Extended Kalman Filter (EKF) which merges IMU (20 Hz) and global metrology ultrasound time-of-flight information (5 Hz).

The SPHERES communications system automatically queries the EKF at 5 Hz and downloads that data to the SPHERES GUI via a wireless link. Therefore, the MATLAB ASTRO implementation had state information available at up to 5 Hz. In the cases when a second robot was used as a dynamic obstacle, the updated state of that robot was available at 5 Hz through the same channel. Even though the state information is available at 5 Hz, the implementation of the ASTRO algorithm is not designed as a periodic process. Rather, the implementation attempts to obtain an optimal solution, but if a maximum period is reached, the current best trajectory is returned. To maintain a minimum path update frequency, the implementation of ASTRO returns updates at a fixed time limit (regardless of optimality).

[5] The GUI regularly used aboard the ISS provides limited information about the robots and is intended primarily for operation, but does not have sufficient information for direct iterative research.

6.3. Solver implementation

With a tight deadline for the test session on the ISS, the setup of the architecture described above took priority, and led to there being limited time to implement the ASTRO algorithm. As such, only a partial version of ASTRO was implemented, one that used the MATLAB nonlinear optimization routine *fmincon* rather than the full ASTRO gradient descent method. This meant that the tests focused on the parameterization and cost function formulation of ASTRO, and unfortunately did not test the benefit of the projected subspace component.

For all tests, 7th order polynomial coefficients for each axis were used as parameters for optimization. Boundary conditions for start and end states were enforced using equality constraints, and the remaining path constraints were supplied as general nonlinear inequality constraints in the form of Eq. (3). The solver was configured to use the built-in medium-scale Sequential Quadratic Programming (SQP). The use of *fmincon* and SQP, as opposed to ASTRO's projected gradient method, degraded the efficiency of the onboard path planner, but was unfortunately necessary due to limited implementation time.

6.4. Test scenarios

A sequence of tests was devised to validate incremental features of the algorithm, and test it under various conditions. For all tests, combinations of the configurable parameters were available to the operating crew member including:

- the total time for a test to be executed, t_f;
- the final target location, $\mathbf{x}_f$;
- the time available for computation before the algorithm should be stopped and the path sent to the robot;
- the order of the Legendre polynomial series expansion; and
- a set of five constraints which could be activated or deactivated, each in the form of Eq. (20):
  - planes restricting the robot motion to stay within the ISS test volume;
  - maximum velocity;
  - maximum acceleration;
  - three spherical obstacles (configurable center and radius); and
  - three ellipsoidal obstacles (configurable center and axis sizes).

The ISS test objectives can be summarized as follows: (1) validate the distributed computation architecture, computing optimal paths on a central computer and sending target commands to the robots, (2) validate the capability of ASTRO to function as a real-time path planner starting from arbitrary initial conditions, (3) determine the limitations of ASTRO, in number of constraints, complexity, and speed, and (4) Perform path optimization with the inclusion of other robots functioning as real dynamic obstacles.

The sequence of tests designed to achieve these goals started with a single-shot path planning case with one spherical obstacle, followed by the same case using ASTRO as an on-line planner. The scenario was then made more complex with more obstacles. Next, an actual dynamic obstacle was introduced before increasing the complexity of the scenario again by mixing static and dynamic obstacles.

6.5. Experimental results

The first test verified the distributed control architecture with a simple scenario. A single 0.3 m spherical obstacle was placed between the starting point and the end goal, and the robot acceleration was constrained to 0.05 m/s^2. One complete optimization loop was executed to produce a locally optimal feasible trajectory. After convergence, the GUI sequentially transmitted targets to the robot, indexing the trajectory with total elapsed time. Fig. 6 displays the trajectory as estimated by the SPHERES global metrology system in comparison to the reference path generated from the solver. The robot successfully navigates the edge of the obstacle with close tracking of the reference trajectory.

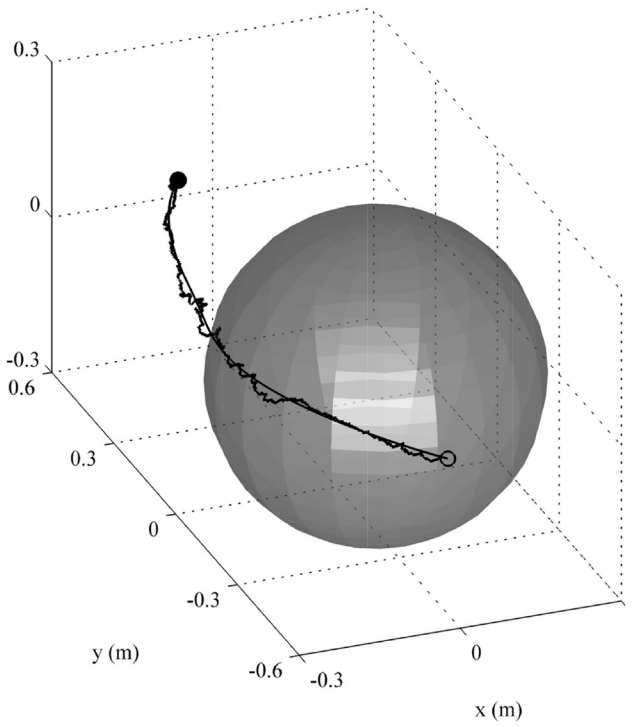

Fig. 6. Initial on-orbit demonstration, validating the distributed architecture.

In all subsequent tests, the distributed planning and control architecture was configured to run ASTRO as an online path planner as illustrated in Fig. 5. In this mode, the GUI repeatedly executed the ASTRO optimization routines using the current position and velocity of the robot as one of the boundary conditions. After the solver converged, or a time limit expired, a single target was transmitted to the robot, and the optimization was restarted. Ideally the full path would be transmitted, but this was a deficiency imposed by the implementation onboard. The bandwidth to communicate from MATLAB to the SPHERES was limited, which restricted the size of the message being sent to one future target position. Additionally, MATLAB was only able to run on a single thread, so could not run a separate process to frequently send updated targets to represent the full trajectory.

Fig. 7 shows a repeat of the initial validation test with the online planning method enabled. Darker colors in the figure show later trajectories. In comparison to the first test, the robot still navigates around the obstacle, but there is a gradual drift in the trajectory as the solver repeats the optimization. The most likely cause of this behavior is that as the robot drifts towards positive z, for instance, each replanned trajectory also moves further towards positive z at the starting point. The waypoint that is selected from these trajectories is then further towards positive z, which reduces the distance to the robot along the z direction. Without a large error, the PID controller takes much longer to integrate a command above the minimum thruster firing time and correct the drift.

The next test introduced a more cluttered obstacle field with three ellipsoids between the initial position and the final target. Fig. 8 displays trajectories created by the ASTRO solver as the robot moved to different positions in the work area. Each initial condition starting from a feasible location resulted in a feasible trajectory plan, validating the ability for the algorithm to plan on-line from arbitrary initial conditions.

In many cases, the actual trajectory followed by the robot varied significantly from the desired paths created by the solver due to an important implementation error. In each optimization cycle, the trajectory was planned from the current position of the robot and the

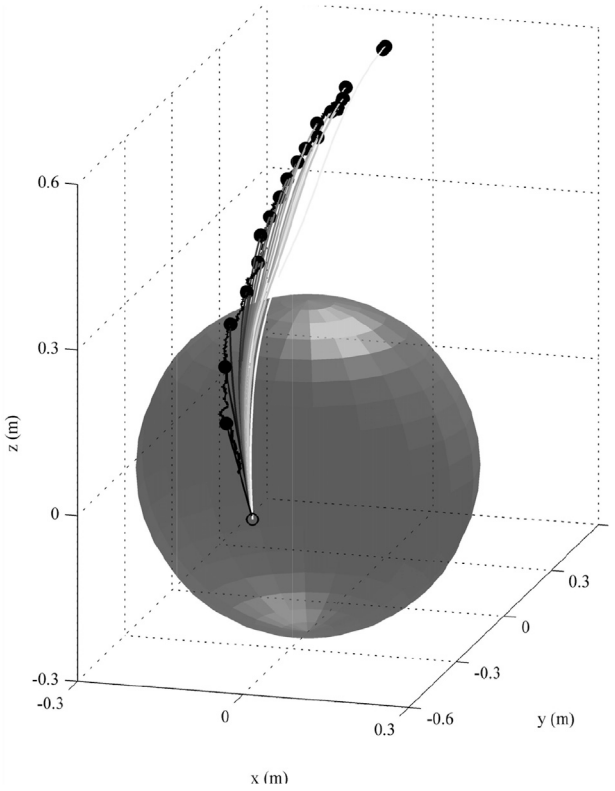

Fig. 7. Repeated optimization applied to the initial validation test, showing a gradual drift in the trajectory.

transmitted target selected by indexing the new trajectory at twice the optimization computation time. The time offset was intended to place the next target at the predicted position of the robot at the completion of the next optimization cycle. Recall that sending a single future target point rather than the full trajectory was an implementation limitation. Ideally ASTRO would send the full path each cycle. The intended behavior of this work-around is illustrated in Fig. 9. This replanning and target selection approach is based on the assumption that the trajectories have minimal curvature and don't vary substantially with each replan. When these assumptions are violated it can cause unstable oscillations in the selection of targets: an effect that is strongly accentuated when the computational time is large, leading to targets that are selected far along the trajectory.

Fig. 10 shows the same scenario as Fig. 8 with the transmitted targets and true trajectory overlaid. With the computational lag, the trajectory can deviate significantly from the planned path if the single target point is not chosen correctly. In the first iteration, the solver produced the

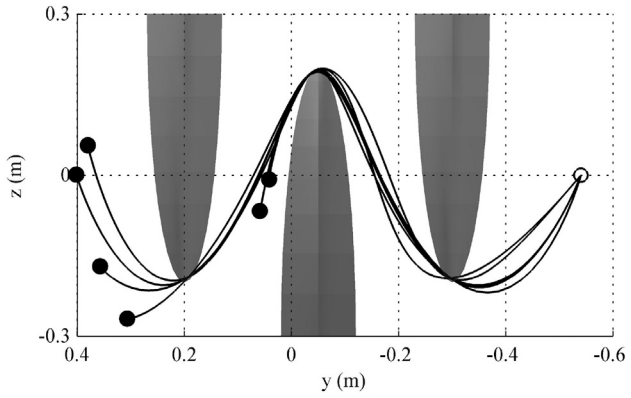

Fig. 8. Feasible trajectories from repeated optimization cycles at several robot locations in a complex obstacle field.

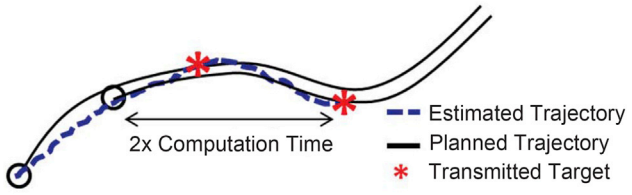

Fig. 9. Intended result of the computation lag compensation: selecting a target to be reached at the end of the next optimization cycle. The target is intended to be set ahead of the current position but near the current trajectory.

trajectory starting in the upper left corner as the robot drifted downward. As it reached the second point (the black dot at $z = 0$), the optimization completed, and it received target 1 (open circle). The large curvature, and long computation time caused the target to request an infeasible traversal of the obstacle. Note that if the full trajectory was transmitted to the robot, this would not have occurred. While in transit, the GUI optimization started at the second point, eventually producing target 2. Even the seemingly small change in starting position gave a large change in the time indexes along the trajectory, causing the target to be selected closer to the starting position and commanding the robot to oscillate across the obstacle, an effect which repeats with the third target.

The oscillating issues were made salient by the large computation times of 10–20 s. These times resulted due to the limitation of using *fmincon* and SQP rather than ASTRO's projected gradient method. Improvements in computation time by implementing the complete ASTRO, and the ability to communicate a complete trajectory, rather than a single waypoint, will ameliorate the issue observed.

A second SPHERES robot was used as a real and dynamic obstacle for the next set of tests. In the first dynamic obstacle test the obstacle robot (SPHERES 2) was commanded to move between fixed start points and end points, while SPHERES 1, using ASTRO as an online planner was tasked with moving from a starting point before SPHERES 2 to a final point past the end point of SPHERES 2 (i.e. overtaking the second robot). In each optimization cycle, the state of SPHERES 2 was communicated from the SPHERES, through the communication architecture, to MATLAB (see Fig. 5). The communicated state is used to create a set of spherical obstacle constraints for each timestep to effectively create a cylinder extending from the current position along the velocity vector to a predicted final position (assuming constant velocity). Each of these constraints are modeled with a radius equal to double the radius of the SPHERES, to represent the obstruction one SPHERES robot would present

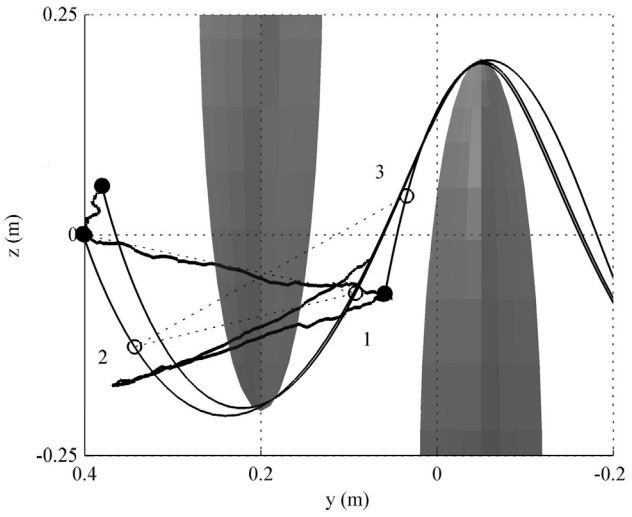

Fig. 10. Transmitted single target oscillation back and forth across obstacle at each solver iteration.

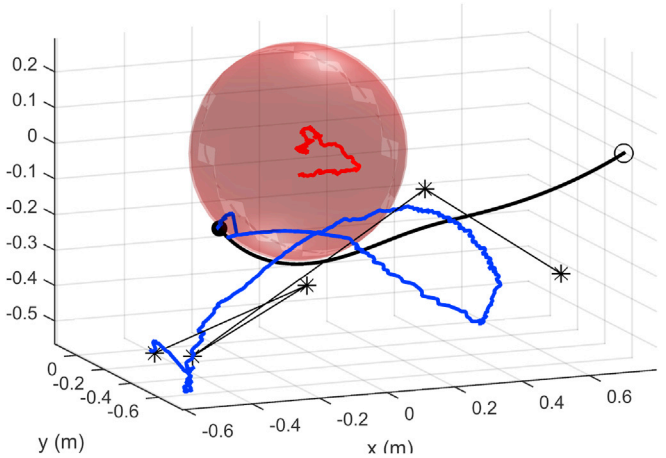

Fig. 11. Planned path (bold line) in one time instance around a real dynamic obstacle (red sphere), with history of estimated paths (blue line for SPHERE 1 and red line for SPHERE 2) and transmitted targets (black stars). Path planned from closed black circle to open black circle.

to another.

The results from this test case, while still affected by the implementation limitations discussed above, did show that the planner could consider dynamic obstacles and adjust plans accordingly. For instance in Fig. 11 the planned path (bold) has adjusted to curve around the constraint represented by the obstacle (red). In Fig. 11 the blue line is the estimated path of SPHERE 1, and the red line is the estimated path of SPHERE 2. The black stars are the history of transmitted targets. The dispersion of the these target points is still related to the "single target" issue already discussed above: an issue which meant that the actual path of SPHERES 1 did not follow the commanded targets.

A more challenging scenario with a mix of virtual static obstacles and real dynamic obstacles was then run. In this case, SPHERE 2 was set to orbit a central, virtual spherical obstacle, a path that calls for more change in the velocity vector than the previous case, and hence presents a greater challenge as a dynamic obstacle. SPHERE 1 was tasked to move diagonally through the permissible volume (a cube assigned inside the ISS), then traverse one of the sides of the permissible volume before moving back along the other diagonal (i.e. a set of three different goal locations). The test results demonstrated that the algorithm can handle complex scenarios with large changes in the motion of dynamic obstacles. In Fig. 12 it can be seen that the trajectory adjusts to curve around the dynamic constraint successfully in the two different positions of the dynamic obstacle. Note that the actual trajectory did not follow the sequence of planned paths and waypoints due to the single target issues discussed above; hence the history of target waypoints are not shown.

While the test results showed successful handling of dynamic obstacles, they also identified a number of areas for improvement. One particularly clear observation was that the method of representing the dynamic obstacle constraint over time was too conservative, taking up too much of the feasible space that in reality was free to traverse. Additionally, the modeling of the dynamic constraints by many spherical obstacles resulted in a large number of constraints to check, slowing the optimization computation.

6.6. *Lessons learned*

The on-orbit investigations were hampered by a number of unfortunate limitations:

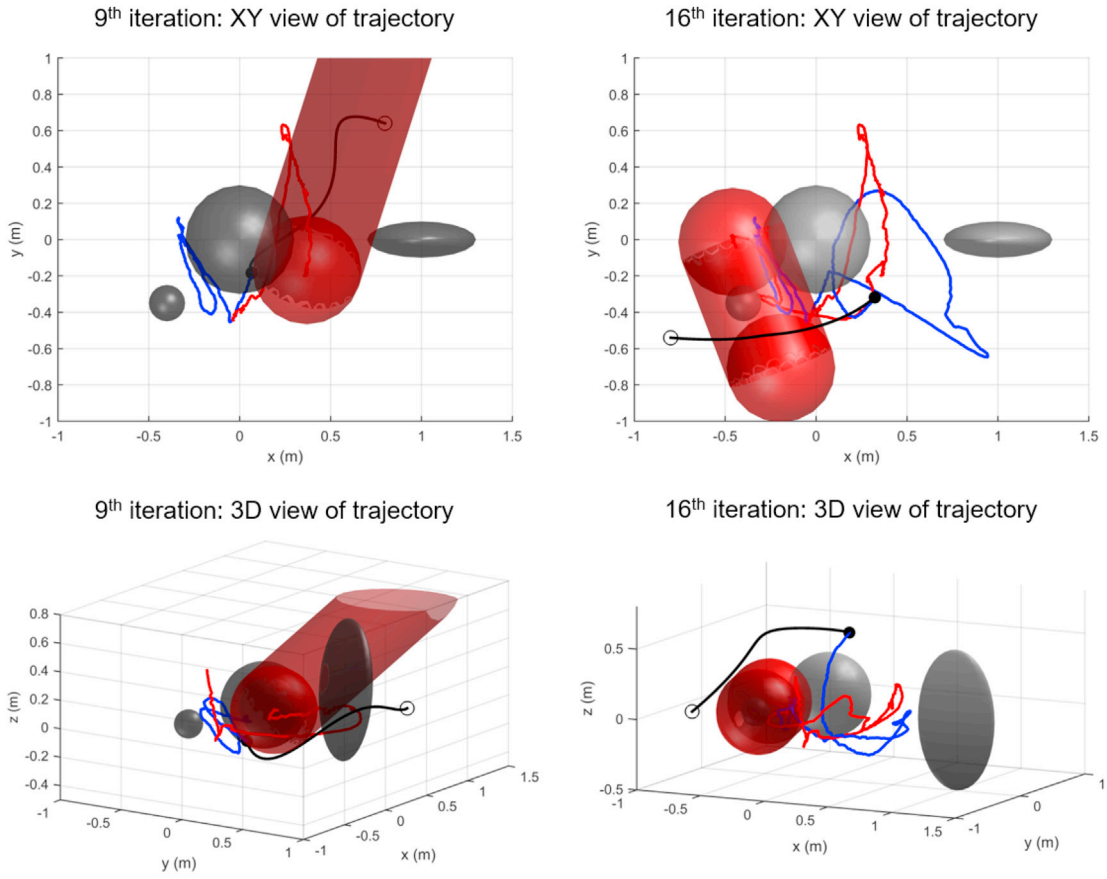

Fig. 12. Planned path (bold line) at a given time instance around a real dynamic obstacle (red shape) and simulated static obstacles (grey), with history of estimated paths (blue line for SPHERE 1 and red line for SPHERE 2). Path planned from closed black circle to open black circle.

- The restricted timeline for the on-orbit test session meant that *fmincon* and SQP were used rather than the full ASTRO gradient descent algorithm.
- The limitation in communication bandwidth, and lack of multiple MATLAB threads meant only a single target could be transmitted.
- The large computation times, due to the use of *fmincon*, coupled with the need to send a single target required a lag compensation in sending the target, which led to the unstable oscillations.

Despite these limitations, the investigations were still able to achieve a number of the initial goals. The general architecture was validated, as was the use of ASTRO in planning on-line from arbitrary locations, in a manner that could deal with dynamic obstacles.

There were also a number of key lessons about the implementation of on-line path planners that may be beneficial to practitioners in general. The experiments highlighted, that in cases where computation time is limited:

- Infrequent single waypoint communication is ill-advised. A full trajectory, or a section of a trajectory should be communicated for the vehicle to follow.
- Careful consideration is required for designing the replanning strategy, in particular for:
  - Indexing the commanded targets or target trajectories, to avoid potential oscillations in commanded path.
  - Selecting where to initiate a new planning cycle from.
- Uncertainty in the computational lag makes designing the replanning strategy very challenging.
- A suitably tuned inner loop feedback controller is critical to have a reliable prediction of where the vehicle will be at a point in the future, to have smooth transitions to newly planned trajectories.

Many of the issues experienced could be addressed with relatively simple measures: implementing the full ASTRO algorithm and utilizing the more modern hardware now available to enable the communication of a full trajectory and a reduced computation time. Nonetheless, the lessons learned are still valuable for implementation of on-line path planning for other systems where the computation times are slow with respect to the vehicle dynamics.

7. Post-flight improvements

The lessons learned from the on-orbit implementation of ASTRO were used to further refine the algorithm. In addition to implementation changes addressing the limitations outlined above, there were a number of improvements that were made to the ASTRO algorithm:

- Using the full ASTRO gradient descent method;
- Compiling the optimization code; and
- Updating how dynamic obstacles are modeled, to be less conservative and more computationally efficient.

Since completion and review of the ISS onboard experiments, the full ASTRO algorithm with projected gradient descent has been implemented in both interpreted MATLAB and compiled C-Mex versions. Combining the projected gradient subspace descent, with a BFGS [51–54] quasi-Newton descent method results in a computational speed increase up to 40 times the *fmincon* implementation for equivalent problems. Compiling the algorithm then improves the speed by another 2–3 times. This speed increase would be sufficient to overcome the computation time issues experienced in the on-orbit implementation.

Example times for the preliminary simulation and more challenging simulation cases are shown in Table 1. The timing comparisons were run with 64 bit MATLAB R2014a on an Intel Core i7-3520M, 2.9 GHz processor.

The additional simulations are designed to test the capability of the full ASTRO algorithm.[6] All tests are within a restricted geometry, representative of a junction of six corridors on the ISS, have a performance constraint on the maximum acceleration (limiting to 0.005 m/s^2), and include one or more dynamic obstacles. To further decrease the calculation time, a relaxed problem with only dynamic constraints is first solved, then the geometric and performance constraints are progressively introduced and the problem re-solved using the previous solution as the initial guess. This technique resulted in a 3 times speed increase for the two robot test case.

Supplementary video related to this article can be found at https://doi.org/10.1016/j.actaastro.2017.10.001.

These simulations also utilized an improved method of modeling dynamic obstacles that makes use of the full ASTRO implementation. Instead of defining a set of constraints that restrict the whole path of a dynamic obstacle, one spherical obstacle is defined, with a center point trajectory. When checking an ASTRO planned trajectory for collisions with the obstacle constraint, each point is checked only against the dynamic obstacle position at the corresponding time. This means that: (1) the obstacle only restricts the feasible space where it is at the time, and (2) only one constraint needs to be checked, rather than up to 100, improving computation time. Additionally, the obstacle trajectory can be defined to follow any type of motion, which is of use if more was known about the dynamics of the obstacle. With uncertain estimates of the obstacle position, there is the potential that this new approach may not be conservative enough, in which case the size of the constraint could be increased.

The new method of modeling dynamic obstacles allows for more restrictive planning tasks to be achieved, such as shown in Fig. 13. This test case reflects a more challenging form of the tests performed on the ISS, with a dynamic spherical obstacle (SPHERES 2) modeled such that it moves to obstruct a robot (SPHERES 1) planning a path from $[-0.5, 0, 0]$. m to $[0.5, 0, 0]$ m. The corridors have a radius of 0.155 m. The two SPHERES are represented as spheres with radius 0.05 m. The SPHERES 1 center point is modeled to follow the planned trajectories. At regular 10 s intervals SPHERES 2 changes direction to move at a constant speed of 0.007 m/s towards the mid point of the direct line from the current location of SPHERES 1 and the target final position (i.e. SPHERE 2 is actively trying to get in the way). At the same time SPHERES 1 is modeled to observe the position and velocity of the obstacle and uses the information to replan a trajectory.

Fig. 13 depicts the path taken by SPHERES 1 (dark sphere) to get around SPHERES 2 (light sphere) within the grey corridor. The solid line is the path taken, and the dot-dash lines are interim planned trajectories before the path is replanned at each black circle.

The robot's trajectory updates to weave above and below, side to side to result in a safe path around the obstacle (stages of updates are shown in the bottom three images in Fig. 13, where the dashed black line is the current planned path). An animation of the scenario is shown in Video S1 in the supplementary material (online version only). A further series of complex planning simulations, demonstrating the enhanced capability of the algorithm are presented in Morrell's work [55,56].

Supplementary video related to this article can be found at https://doi.org/10.1016/j.actaastro.2017.10.001.

8. Conclusion

This research accomplished two objectives: the development of an on-line path planning approach for operation in the presence of complex and dynamic constraints, and the demonstration of this algorithm aboard the International Space Station in a true 6-DOF environment. As demonstrated by simulation and in space, the ASTRO algorithm rapidly identifies feasible trajectories for a free-flying robot to navigate between 3D

[6] Test case "2 Adversarial" is presented here. Test cases "2 Robots" and "6 Robots" can be seen in Video S2 and Video S3 of the supplementary online material.

Table 1
Path planning calculation times and improvements.

Test Case	Plan Cycles	Average times (s) per path plan				Improve-ment
		Standard Code		Compiled		
		fmincon	P.G.[a]	fmincon	P.G.[a]	(old/new)
1st Sim.	1	3.84	0.85	3.02	0.40	9.59
2 Adversarial	10	41.04	5.09	20.74	1.58	26.00
2 Robots	6	43.36	1.02	41.04	0.41	106.72
6 Robots	24	35.65	2.41	23.01	1.45	24.63

[a] P.G. = Projected Gradient method.

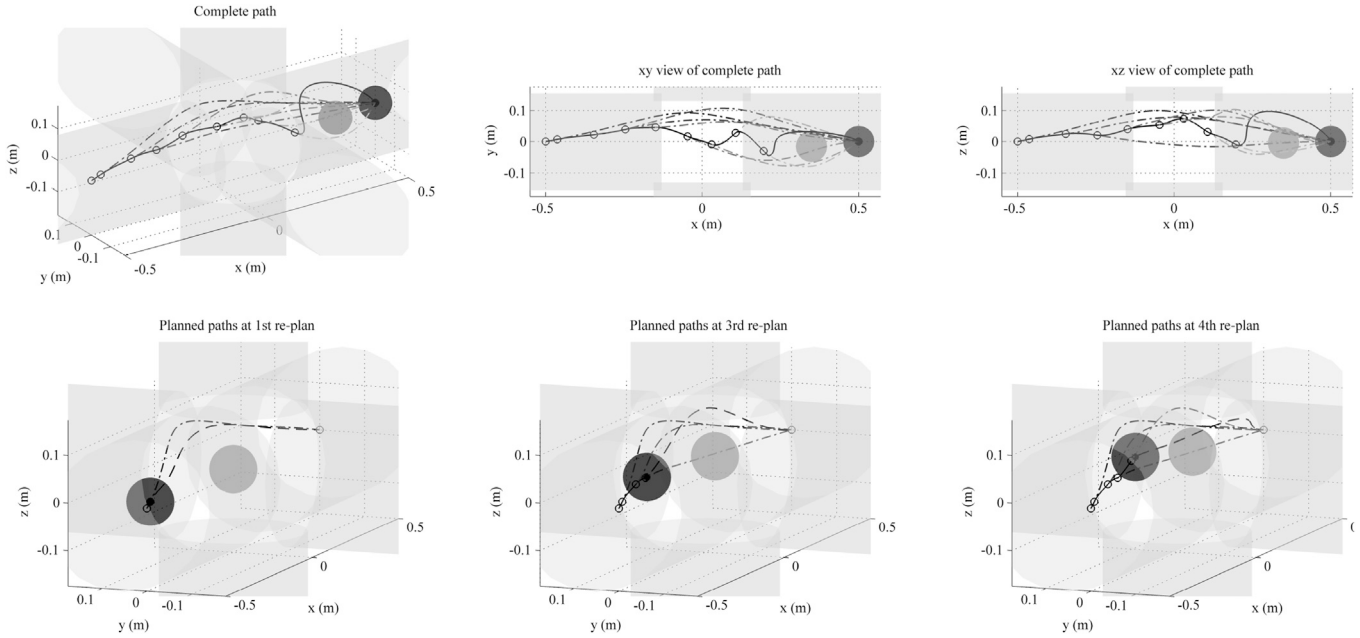

Fig. 13. Simulated planned and executed trajectories for the 2 SPHERES adversarial test case.

boundary conditions within surroundings that include numerous static and dynamic obstacles, geometric constraints and performance limitations. This approach has two core innovative components. One: the formulation of the problem to use Legendre polynomials to transform a complex, non-linear cost function into a simpler form that can be quickly computed. Two: the use of a subspace projection technique to ensure boundary conditions are always met throughout an iterative optimization, enabling rapid generation of collision free trajectories.

ASTRO was demonstrated aboard the International Space Station by combining the SPHERES facility with an onboard laptop with access to MATLAB software. The implementation created a multi-threaded asynchronous design with a periodic PID path following controller and an aperiodic implementation of ASTRO.

The ISS tests validated the system architecture, as well as the use of the ASTRO algorithm for planning trajectories from arbitrary initial conditions, and demonstrated a capability to handle dynamic obstacles. The tests also identified a number of unfortunate implementation limitations. An incomplete implementation of ASTRO led to large computation times. Limited communication bandwidth required single waypoint targets to be sent rather than full trajectories, with the targets indexed to account for computational lag. The combination of these factors led to undesirable oscillations of transmitted targets when replanning online. Nonetheless, the results provided some valuable lessons learned for implementation of computationally limited systems. To improve the on-orbit performance, it is planned to communicate the complete trajectory, utilize more powerful hardware, and implement the full ASTRO algorithm, including the improvements made following the ISS tests. These improvements have implemented the full algorithm, compiled the code, and used a more effective method of representing dynamic obstacles, all helping to improve the speed of the algorithm by more than an order of magnitude. The effectiveness of the improved algorithm has been shown in simulation with a challenging scenario including geometric and performance constraints with dynamic obstacles. The updated and improved ASTRO algorithm will be tested onboard the ISS as part of a new vision-based navigation system for space robotics.

Acknowledgments

The authors wish to give special thanks to Colonel Michael Fincke, Commander of ISS Expedition 18, for coordinating ISS crew time to enable testing of ASTRO and for his personal support during test execution.

The authors thank the entire SPHERES team of the MIT Space Systems Laboratory for supporting operations aboard the ISS. The authors especially thank Christopher Mandy for creating the first implementation of ASTRO in MATLAB which was used during the ISS operations described in this paper. In addition the authors acknowledge the leadership and guidance of the MIT SPHERES Principal Investigator, Professor David W. Miller. Lastly, the team thanks the DoD Space Test Program, the human spaceflight office at NASA Johnson Space Center, and the Marshall Space Flight Center payload operations team for their support of SPHERES operations and specifically to enable the real-time MIT to ISS communications which made the test possible. Steve Ulrich also acknowledges the financial support from the Natural Sciences and Engineering Research Council of Canada (NSERC) under the Postdoctorate Fellowship PDF-438572-2013.

References

[1] S. McCamish, M. Romano, S. Nolet, C. Edwards, D.W. Miller, Testing of multiple-spacecraft control on SPHERES during close-proximity operations, J. Spacecr. Rockets 46 (6) (2009) 1202–1213.

[2] M.A. Hurni, P. Sekhavat, M. Karpenko, I.M. Ross, A pseudospectral optimal motion planner for autonomous unmanned vehicles, in: Proceedings of the 2010 American Control Conference, IEEE, 2010, pp. 1591–1598.

[3] L. Blackmore, H. Li, B. Williams, A probabilistic approach to optimal robust path planning with obstacles, in: American Control Conference, 2006, IEEE, 2006, p. 7.

[4] J. Omer, J.-L. Farges, Hybridization of nonlinear and mixed-integer linear programming for aircraft separation with trajectory recovery, intelligent transportation systems, IEEE Trans. 14 (3) (2013) 1218–1230, https://doi.org/10.1109/TITS.2013.2257758.

[5] M. Kobilarov, Discrete Geometric Motion Control of Autonomous Vehicles, University of Southern California, 2008.

[6] Y. Xue, B. Lee, D. Han, Automatic collision avoidance of ships, proceedings of the institution of mechanical engineers, Part M J. Eng. Marit. Environ. 223 (1) (2009) 33–46.

[7] P. Lu, X. Liu, Autonomous trajectory planning for rendezvous and proximity operations by conic optimization, J. Guid. Control Dyn. 36 (2) (2013) 375–389.

[8] S. Cafieri, N. Durand, Aircraft deconfliction with speed regulation: new models from mixed-integer optimization, J. Glob. Optim. 58 (4) (2014) 613–629.

[9] J. Luo, K. Hauser, An empirical study of optimal motion planning, in: Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on, IEEE, 2014, pp. 1761–1768.

[10] S. Karaman, M.R. Walter, A. Perez, E. Frazzoli, S. Teller, Anytime motion planning using the RRT*, in: Robotics and Automation (ICRA), 2011 IEEE International Conference on, IEEE, 2011, pp. 1478–1483.

[11] Q. Gong, R. Lewis, M. Ross, Pseudospectral motion planning for autonomous vehicles, J. Guid. Control Dyn. 32 (3) (2009) 1039–1045.

[12] S.M. LaValle, Planning Algorithms, Cambridge University Press, 2006.

[13] S.-H. Suh, M.-S. Kim, An algebraic approach to collision-avoidance trajectory planning for dual-robot systems: formulation and optimization, Robotica 10 (02) (1992) 173–182.

[14] A.V. Rao, D.A. Benson, C. Darby, M.A. Patterson, C. Francolin, I. Sanders, G.T. Huntington, Algorithm 902: GPOPS, a Matlab software for solving multiple-phase optimal control problems using the gauss pseudospectral method, ACM Trans. Math. Softw. (TOMS) 37 (2) (2010) 22.

[15] J.T. Betts, W.P. Huffman, Sparse Optimal Control Software SOCS, Mathematics and Engineering Analysis Technical Document MEA-LR-085, Boeing Information and Support Services, The Boeing Company, 1997. PO Box 3707, 98124–2207.

[16] B.A. Conway, A survey of methods available for the numerical optimization of continuous dynamic systems, J. Optim. Theory Appl. 152 (2) (2012) 271–306.

[17] J.T. Betts, Practical Methods for Optimal Control and Estimation Using Nonlinear Programming, SIAM, 2010.

[18] I.M. Ross, M. Karpenko, A review of pseudospectral optimal control: from theory to flight, Annu. Rev. Control 36 (2) (2012) 182–197.

[19] A.V. Rao, A survey of numerical methods for optimal control, Adv. Astronaut. Sci. 135 (1) (2009) 497–528.

[20] Arthur Richards, Tom Schouwenaars, Jonathan P. How, Eric Feron, Spacecraft trajectory planning with avoidance constraints using mixed-integer linear programming, J. Guid. Control Dyn. 25 (4) (2002) 755–764.

[21] L. Pallottino, E.M. Feron, A. Bicchi, Conflict resolution problems for air traffic management systems solved with mixed integer programming, intelligent transportation systems, IEEE Trans. 3 (1) (2002) 3–11.

[22] J.S. Bellingham, M. Tillerson, M. Alighanbari, J.P. How, Cooperative path planning for multiple UAVs in dynamic and uncertain environments, in: Decision and Control, 2002, Proceedings of the 41st IEEE Conference on, vol. 3, IEEE, 2002, pp. 2816–2822.

[23] L.S. Breger, J.P. How, Safe trajectories for autonomous rendezvous of spacecraft, J. Guid. Contr. Dynam. 31 (5) (2008) 1478–1489.

[24] A. Alonso Ayuso, L. Escudero, F. Martin Campo, Collision avoidance in air traffic management: a mixed-integer linear optimization approach, intelligent transportation systems, IEEE Trans. 12 (1) (2011) 47–57, https://doi.org/10.1109/TITS.2010.2061971.

[25] U. Eren, B. Açıkmese, D.P. Scharf, A Mixed Integer Convex Programming Approach to Constrained Attitude Guidance, 2015, pp. 1120–1126.

[26] Y.-Z. Luo, Y.-J. Lei, G.-J. Tang, Optimal multi-objective nonlinear impulsive rendezvous, J. Guid. Control Dyn. 30 (4) (2007) 994–1002.

[27] J. Ousingsawat, M.E. Campbell, On-line estimation and path planning for multiple vehicles in an uncertain environment, Int. J. Robust Nonlinear Control 14 (8) (2004) 741–766.

[28] J. Bruce, M. Veloso, Real-time randomized path planning for robot navigation, in: Intelligent Robots and Systems, 2002. IEEE/RSJ International Conference on, vol. 3, IEEE, 2002, pp. 2383–2388.

[29] R. Luna, I. Sucan, M. Moll, L.E. Kavraki, et al., Anytime solution optimization for sampling-based motion planning, in: Robotics and Automation (ICRA), 2013 IEEE International Conference on, IEEE, 2013, pp. 5068–5074.

[30] J.J. Kuffner, S.M. LaValle, RRT-connect: an efficient approach to single-query path planning, in: Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on, vol. 2, IEEE, 2000, pp. 995–1001.

[31] J.D. Munoz, N.G. Fitz Coy, Rapid path-planning options for autonomous proximity operations of spacecraft, in: Proceedings of the AIAA/AAS Astrodynamics Specialist Conference, Ontario Canada, Toronto, 2010.

[32] S.A. Masoud, A.A. Masoud, Motion planning in the presence of directional and regional avoidance constraints using nonlinear, anisotropic, harmonic potential fields: a physical metaphor, systems, man and cybernetics, Part A Syst. and Hum. IEEE Trans. 32 (6) (2002) 705–723.

[33] E. Rimon, D.E. Koditschek, Exact robot navigation using artificial potential functions, robotics and automation, IEEE Trans. 8 (5) (1992) 501–518.

[34] E. Lalish, K.A. Morgansen, T. Tsukamaki, Decentralized reactive collision avoidance for multiple unicycle-type vehicles, in: American Control Conference, 2008, IEEE, 2008, pp. 5055–5061.

[35] D.E. Chang, S.C. Shadden, J.E. Marsden, R. Olfati Saber, Collision avoidance for multiple agent systems, in: Proceedings of the 42nd IEEE Conference on Decision and Control, IEEE, 2003.

[36] G. E. Chamitoff, Autonomous guidance for the recovery and landing of a remotely piloted vehicle, In: IFAC Aerospace Control Conference, Palo Alto, California, 1994.

[37] I.M. Ross, F. Fahroo, A unified computational framework for real-time optimal control, IEEE (42nd; 2003; Maui, Hawaii), in: Proceedings of the 42nd IEEE Conference on Decision and Control Maui, Hawaii USA, December 2003, IEEE, 2003.

[38] G. Elnagar, M.A. Kazemi, M. Razzaghi, The pseudospectral Legendre method for discretizing optimal control problems, IEEE Trans. Automat. Control 40 (10) (1995) 1793–1796.

[39] F. Fahroo, I. Ross, Computational optimal control by spectral collocation with differential inclusion, in: NASA Conference Publication, Citeseer, 1999, pp. 185–200.

[40] N. Xu, W. Kang, G. Cai, B.M. Chen, Minimum-time trajectory planning for helicopter uavs using computational dynamic optimization, in: 2012 IEEE International Conference on Systems, Man, and Cybernetics (SMC), IEEE, 2012, pp. 2732–2737.

[41] K.P. Bollino, L.R. Lewis, P. Sekhavat, I.M. Ross, Pseudospectral optimal control: a clear road for autonomous intelligent path planning, in: Proceedings of the AIAA InfoTech at Aerospace Conference and Exhibit, 2007, pp. 1228–1241.

[42] G.E. Chamitoff, A. Saenz Otero, J.G. Katz, S. Ulrich, Admissible subspace TRajectory optimizer (ASTRO) for autonomous robot operations on the space station, in: AIAA Guidance, Navigation, and Control Conference, AIAA, Reston, VA, 2014, pp. 1–17.

[43] S. Mohan, A. Saenz Otero, S. Nolet, D.W. Miller, S. Sell, SPHERES flight operations testing and execution, Acta Astronaut. 65 (7) (2009) 1121–1132.

[44] A. Saenz Otero, J.G. Katz, D.W. Miller, SPHERES demonstrations of satellite formations aboard the ISS, in: 32nd Annual AAS Guidance and Control Conference, aAS Paper, Breckenridge, Colorado, 2009, pp. 09–011.

[45] J.L. Ramirez Riberos, M. Pavone, E. Frazzoli, D.W. Miller, Distributed control of spacecraft formations via cyclic pursuit: theory and experiments, J. Guid. Control Dyn. 33 (5) (2010) 1655–1659.

[46] A. Fejzic, S. Nolet, L. Breger, J. P. How, D. W. Miller, Results of SPHERES microgravity autonomous docking experiments in the presence of anomalies, In: 59th International Astronautical Congress, Glasgow, Scotland, 2008.

[47] S. Nolet, Development of a Guidance, Navigation and Control Architecture and Validation Process Enabling Autonomous Docking to a Tumbling Satellite (Ph.D. thesis), Massachusetts Institute of Technology, Department of Aeronautics and Astronautics, Cambridge, Massachusetts, 2007.

[48] S. Nolet, The SPHERES navigation system: from early development to on-orbit testing, in: AIAA Guidance, Navigation and Control Conference, aIAA Paper, Hilton Head, South Carolina, 2007, pp. 2007–6354.

[49] W. Fehse, Automated Rendezvous and Docking of Spacecraft, Cambridge University Press, Cambridge, United Kingdom, 2003.

[50] A. Saenz Otero, D. W. Miller, The SPHERES ISS laboratory for rendezvous and formation flight, in: 5th ESA International Conference on Spacecraft Guidance, Navigation and Control Systems, Frascati, Italy, 2002.

[51] C.G. Broyden, The convergence of a class of double-rank minimization algorithms, J. Inst. Math. Appl. 6 (1970) 76–90.

[52] R. Fletcher, A new approach to variable metric algorithms, Comput. J. 13 (1970) 317–322.

[53] D. Goldfarb, A family of variable metric updates derived by variational means, Math. Comput. 24 (1970) 23–26.

[54] D.F. Shanno, Conditioning of Quasi-Newton methods for function minimization, Math. Comput. 24 (1970) 647–656.

[55] Benjamin J Morrell, Peter W Gibbens, Gregory E Chamitoff, Application of a trajectory optimisation algorithm for dynamic obstacle avoidance and multiple vehicle coordination, in: Fourth Australasian Conference on Unmanned Systems, 2014.

[56] Benjamin J Morrell, Gregory E Chamitoff, Peter W Gibbens, Autonomous operation of multiple free-flying robots on the international space station, in: 25th AAS/AIAA Spaceflight Mechanics Conference, 2015.