AIAA SciTech
13-17 January 2014, National Harbor, Maryland
AIAA Guidance, Navigation, and Control Conference

AIAA 2014-1290

Admissible Subspace TRajectory Optimizer (ASTRO) for Autonomous Robot Operations on the Space Station

Gregory E. Chamitoff,* Alvar Saenz-Otero,† Jacob G. Katz,‡ and Steve Ulrich§
Massachusetts Institute of Technology, Cambridge, Massachusetts, 02139

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 designing a trajectory for a robot to navigate within complex surroundings that include numerous obstacles (generalized shapes) and constraints (geometric and performance limitations). The methodology employs a unique transformation that effectively changes a complex optimization problem into one with a positive definite cost function that enables high convergence rates for complex geometries, enabling its application to real-time operations. This strategy was implemented on the Synchronized Position Hold Engage Reorient Experimental Satellite (SPHERES) test-bed on the International Space Station (ISS), and iterative experimental testing was conducted onboard the ISS during Expedition 17 by the first author.

I. Introduction

Real-time 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 robotic systems, self-guided aircraft, and autonomous spacecraft 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 technology 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. Likewise, given a continuously evolving internal model, it is necessary to plan and re-plan an optimal approach for conducting a mission or performing a task. This may also involve simultaneous cooperative planning with other autonomous vehicles.

To accomplish this, algorithms to solve complex multi-dimensional optimization problems in near real-time for spacecraft proximity operations have been studied extensively. Of special interest are trajectory optimization techniques with not only terminal constraints, such a position and velocity, but interior-point constraints for example. The complex constraints can be presented by several mathematical forms, such as equality constraints and second-order inequality constraints. For example, thrust limited maneuvers with path constraints based on the primer-vector theory has been considered by Taur et al.,[2] where the full adjoint equations and corresponding optimality constraints are imposed for the rendezvous and intercept transfers of a spacecraft between two circular, coplanar orbits with a minimum periapsis constraint for the chaser vehicle.

Trajectory optimization based on adaptive artificial potential functions (AAPF) was proposed by Munoz and Fitz-Coy[3] for rapid path-planning of spacecraft autonomous proximity operations. The AAPF method permits attractive potential with time dependent weights which are defined by an adaptive update law. Simulation results demonstrated that the developed methodology enables proximity operations with increased performance compared to the standard artificial potential function in an evolving environment.

*Research Affiliate MIT, NASA Astronaut - ISS Expedition 17/18 Flight Engineer & Science Officer, Professor, University of Sydney and Texas A&M University. Associate Fellow AIAA.

†Research Scientist, Department of Aeronautics and Astronautics, 77 Massachusetts Avenue. Senior Member AIAA.

‡PhD Candidate, Department of Aeronautics and Astronautics, 77 Massachusetts Avenue; now GNC Engineer, Space Exploration Technologies, Los Angeles, California.

§Postdoctoral Associate, Department of Aeronautics and Astronautics, 77 Massachusetts Avenue; now Assistant Professor, Department of Mechanical and Aerospace Engineering, Carleton University, Ottawa, Canada. Member AIAA.

Ulybyshev[4] presented trajectory optimization methods for low-thrust spacecraft proximity maneuvering at near-circular orbits with interior-point constrains. The methods used discretization of the spacecraft trajectory on segments and sets of pseudo-impulses for each segment. Then, a matrix inequality on the sum of the characteristic velocities for the pseudo-impulses is used to transform the problem into a large-scale linear programming form, and terminal boundary conditions were presented as a linear matrix equation. In Massachusetts

Hadaegh et al.[5] imposed a minimum relative distance constraint via an indirect optimization method using the relative motion equations while ignoring all orbital forces. While such indirect optimization methods are known to provide mathematically elegant solutions and can handle path constraints, they sometimes can be cumbersome, especially as the constraints get more complicated in their algebraic expression. Typically, the addition of each path constraint requires the detailed derivation of the associated Euler-Lagrange (adjoint) equations and a corresponding set of optimality conditions that are not trivial to derive or to implement. Similarly, Jezewski[6] used interior path-constraints in an indirect formulation and also includes the effect of J_2 in the equations of motion and adjoint equations. Haufler et al.[7] used parameter optimization to handle a larger set of path constraints. In addition to the minimum periapsis constraint of Jezewski,[6] a minimum and maximum relative distance between the target and chaser constraint, and a maximum magnitude of the impulses constraint are considered. Hadaegh and Singh[8] also used a method performing parameter optimization to enforce collision avoidance for deep space applications (i.e. no orbital mechanics effects) by representing the trajectories as splines.

Richard et al.[9] described a technique that used mixed-integer linear programming (MILP) in which the optimization objective was to find a trajectory that satisfies the logical constraints and minimizes fuel use.[10] The examples include the use of a microsatellite to inspect the International Space Station (ISS). Breger and How[11] also used mixed-integer linear programming (MILP) to enforce collision avoidance with a variety of shapes, evaluating a series of active constraints (determined via binary variables) at grid points. The keep out zones were simplified to a set of planar faces roughly enclosing the volume to be avoided and the binary variables are used to eliminate all the planar faces that are currently inactive as a path constraint. At most, all but one face was removed leaving a single, simple linear inequality constraint to be evaluated at the corresponding grid point.

Luo et al.[12,13] used multi-objective genetic algorithms to handle path-constrained proximity operations. Collision avoidance with a sphere of safety constraint and a line of approach cone are imposed. With the proposed approach, any trajectory that is considered as violating the imposed constraint was removed from the population after each generation. A fixed, equally-spaced grid is utilized to determine the satisfaction of the constraint over the trajectory after the trajectory is finished integrating.

Ranieri,[14] solved the path-planning optimization problem by using the sparse optimal control software (SOCS), a direct collocation optimization package developed by Boeing that utilizes implicit integration of the equations of motion. This optimization software has the capability to handle path constraints through the use of a series of mesh refinements of the implicit integration grid that ensures that the path constraint is evaluated and satisfied throughout the integration. The grid is made progressively denser as any path constraint boundaries are approached to ensure that the path constraints are not violated between the grid points.

Another highly-efficient solution to address the autonomous path-planning optimization problem makes use of second-order cone programming (SOCP) methodologies, which consist of convex optimization problems with linear cost functions, equality constraints, and second-order conic inequality constraints.[15,16] To find numerical solutions to SOCP problems, interior-point algorithms are often used. As explained by Lu and Liu,[17] such an algorithm has polynomial complexity and upper bounds on operations and the number of iterations required to find the solution, all of which can be determined a priori, even though convergence is typically achieved much faster than those conservative bounds indicate (they guarantee convergence to the global optimal solution if the inequality constraints have an interior point). See Nesterov and Nemirovsky[18] for a detailed treatment on interior-point polynomial algorithms in convex programming. These features make the SOCP methodology appealing for potential real-time applications. In particular, Wang and Boyd[19] and Mattingley and Boyd[20] considered a special class of convex optimization well suited for real-time operations to which SOCP belongs. Lu and Liu[17] developed an SOCP-based method to solve the problem of autonomous trajectory planning for spacecraft rendezvous and proximity operations. In their work, the authors formulated the problem as a nonlinear optimal control problem, subject to various state and control inequality constraints and equality constraints on interior points and terminal condition. A relaxed problem was formed and then solved by a successive solution process, in which the solutions of a sequence of constrained sub-problems with linear, time-varying dynamics are sought. After discretization, each of these problems becomes a second-order cone programming problem. Their solutions, if they exist, are guaranteed to be found by a primal-dual interior-point algorithm. The efficacy of the proposed methodology was demonstrated by numerical simulations.

The methods described above include various techniques to address the challenges of real-time computation and nonlinear constrained non-convex optimization problems. The approach presented in this paper, called the Admissible Subspace TRajectory Optimizer (ASTRO) algorithm, overcomes these challenges by transforming the problem into a parameter space that is well behaved for real-time optimization. The ASTRO algorithm is capable of designing a planned trajectory for a spacecraft robot to navigate within complex surroundings that include numerous obstacles and constraints. Obstacles consist of generalized shapes and include a class of moving objects as well. Constraints include both geometric and performance-based limitations on the robot's motion. Typically, the solution of a general nonlinear optimization problem like this is very difficult and time-consuming to solve. Depending on the technique used, it can also get *caught* in local minima and may never find the globally optimal solution. It's a significant challenge to design a system that can find reliable solutions in real-time. To cope with these difficulties, the original contribution of this work is the development of an algorithm that employs a unique transformation that effectively changes a complex optimization problem into one with a positive definite cost function that enables very fast convergence to a solution. This is accomplished through mathematical definitions for how the constraints should be modeled, and by taking advantage of some unique properties of orthogonal polynomials. By expressing certain parameters with these polynomials, it becomes possible to transform a complex cost function into a simple one in higher dimension. The result is that very high convergence rates can be obtained even for complex geometric problems that would otherwise appear to be quite nonlinear.

In order to test the ASTRO algorithm on the ISS, a custom interface was designed to enable the crew to use MATLAB directly to program and execute the algorithm. Following standard ISS procedures and operational rules it is never possible for the crew to change the software of the SPHERES satellites onboard the ISS themselves. The development of ASTRO aboard ISS was itself a space first: the first ever use of MATLAB aboard the ISS, and the first time that an astronaut conducted research on the ISS based on their own research on the ground. The advantage was that by integrating MATLAB with the SPHERES C-code interface (on a Space-Station Support Computer), it was possible to perform real-time commanding and telemetry communications with the SPHERES satellites. A series of test cases were conducted to demonstrate the ability of ASTRO to incorporate increasingly complex constraints while producing admissible (feasible) trajectories for the SPHERES satellite to follow. All constraints were simulated virtually, but the satellite actually flew physically inside the habitable volume of the US Laboratory on the ISS. In order to test the ability to navigate around moving obstacles, a second SPHERES satellite was specifically programmed to attempt to get in the way. Specifics on the implementation and the experimental results are presented in the paper.

The remainder of this paper is organized as follows. Section II presents the theoretical development of the ASTRO algorithm including conditions for guaranteed convergence. Section III demonstrates the effectiveness of ASTRO in simulation for a complex path planning problem. Section IV covers the experimental setup onboard the ISS and presents an analysis of results from several test cases. Finally, a conclusion is provided in Section V.

II. ASTRO Algorithm

The ASTRO algorithm effectively solves a guidance problem, while an inner-loop controller handles the task of trajectory tracking. In order to assure that the trajectory solution is feasible, maneuvering capabilities of the vehicle are incorporated as performance constraints within the trajectory 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 (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 (geometric) and performance constraints. The second stage uses any remaining computation time to refine the solution towards the optimal path.

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 is parameterized using Legendre polynomials.

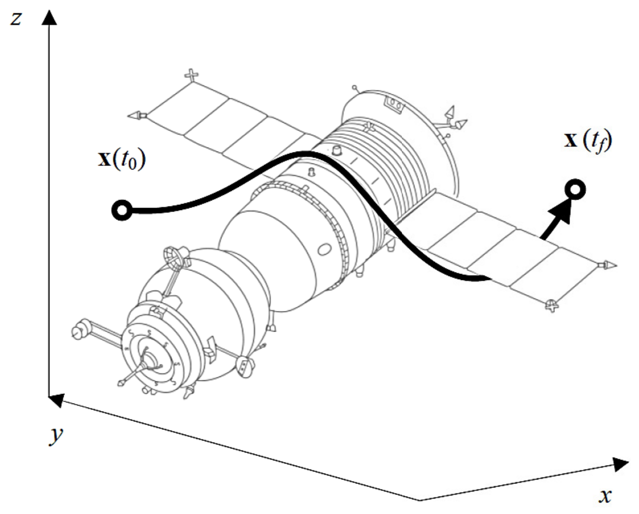

Figure 1. The ASTRO algorithm optimizes the path from $\mathbf{x}(t_0)$ to $\mathbf{x}(t_f)$.

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 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 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. Finally, by taking advantage of certain properties of Legendre polynomials, the constraint and path-length penalty functions can be defined to meet certain conditions that ensure a positive definite cost function with respect to the higher dimensional space of parameter errors. This property guarantees asymptotic convergence directly to an admissible solution, and then to the optimal solution.

II.A. 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), and it must 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 mean the most direct, minimum length, path between the initial position at the initial time $\mathbf{x}_0(t_0)$ to the final position at the final time $\mathbf{x}_f(t_f)$. The initial and final velocity vectors are also specified as $\dot{\mathbf{x}}_0(t_0)$ and $\dot{\mathbf{x}}_f(t_f)$. These boundary conditions can be expressed as follows

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

The cost function to be minimized is the path length expressed by

$$S = \int_{x_0}^{x_f} \mathrm{d}s = \int_{t_0}^{t_f} \sqrt{\dot{x}(t)^2 + \dot{y}(t)^2 + \dot{z}(t)^2}\mathrm{d}t \tag{2}$$

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

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

where n is the number of constraint functions. Note that the ASTRO algorithm is designed to find an admissible trajectory, that is 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

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

where $f_s^2(S)$ 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

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

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

II.B. Legendre Polynomial Parametrization

In general, Eq. (4) represents a difficult and nonlinear optimization problem. The ASTRO algorithm transforms this into a simpler problem by parameterizing the trajectory 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

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

where

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

The C_{ik} are coefficients of the Legendre polynomials of order k that represent the trajectory $x_i(t)$. Time is normalized such that $t' \in [-1, 1]$. Over this interval, and taking advantage of the orthogonality of Legendre functions, the squared path-length component of the cost function in Eq. (4) can be reduced to the following

$$\bar{S}^2 = \sum_{i=1}^{3} \sum_{k=0}^{N} \left\{ C_{ik}^2 \int_{-1}^{1} \left[P_k'(t')\right]^2 \mathrm{d}t' \right\} \tag{8}$$

In Eq. (8), $\bar{S}$ represents an upper bound to the path length, P_k' are derivatives of the standard Legendre polynomial 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 N, which is the minimum order of the polynomials required to find a solution.[21]

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 (8), the full cost function in Eq. (4) can be written as

$$J = \sum_{j=1}^{n+1} [f_j(\mathbf{C})]^2 \tag{9}$$

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 $f_{n+1}(\mathbf{C}) = f_s(\bar{S}^2)$.

II.C. Formulation as a Convex Search Space

The advantage of transforming the cost function from Eq. (4) into Eq. (9) is that it can be shown that the optimization can now be performed on a convex space using a simple gradient-descent procedure. In Eq. (9), J is a convex (positive-definite) function of $f_j(C)$, 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. (9) positive definite in the parameter errors $(C_{ik} - C^*_{ik})$, consider the revised function

$$J' = J - J^* = \sum_{j=1}^{n+1} [f_j(C)]^2 - \sum_{j=1}^{n+1} [f_j(C^*)]^2 \tag{10}$$

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. (10), J' is positive definite with respect to $(C_{ik} - C^*_{ik})$ if one of the following conditions is satisfied:

$$\frac{\partial J'}{\partial C_{ik}} = 2\sum_{j=1}^{n+1} f_j(C_{ik}) \frac{\partial f_j}{\partial C_{ik}} \text{ is monotonic} \tag{11}$$

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

Assuming these conditions on $f_j(C_{ik})$ given by Eq. (11) 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

$$[C_{ik}]_{new} = [C_{ik}]_{old} + \delta C_{ik} \tag{12}$$

where

$$\delta C_{ik} = -\alpha \left[\frac{\partial J'}{\partial C_{ik}} \right] + \beta_{ik} \tag{13}$$

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.

Fortunately, the restrictions on the form of $f_j(C_{ik})$, which apply to the path length penalty $f_{n+1}(C) = f_s(\bar{S}^2)$ and to all constraint functions $f_{c_j}(\mathbf{x}(t), \dot{\mathbf{x}}(t), \ddot{\mathbf{x}}(t))$ are not too difficult to satisfy. In particular, the path length in Eq. (8), can be simplified as follows. Define the path length function f_s as

$$f_s(\bar{S}^2) = \sum_{i=1}^{3} \sum_{k=0}^{N} I_k C_{ik}^2 \tag{14}$$

where

$$I_k = \int_{-1}^{1} \left[P_k'(t') \right]^2 \mathrm{d}t' \tag{15}$$

and now apply the condition defined in Eq. (11). The result is

$$4 \left(\sum_{i=1}^{3} \sum_{k=0}^{N} I_k C_{ik} \right)^2 + \left(\sum_{i=1}^{3} \sum_{k=0}^{N} I_k \right) \left(\sum_{i=1}^{3} \sum_{k=0}^{N} I_k C_{ik}^2 \right) \geq 0 \tag{16}$$

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

$$g(\mathbf{X}) \leq q_0 \tag{17}$$

or

$$f_c(\mathbf{X}) = \begin{cases} \left[g(\mathbf{X}) - g_0 \right]^2, \text{ if } g(\mathbf{X}) > g_0 \\ 0, \text{ if } g(\mathbf{X}) \leq g_0 \end{cases} \tag{18}$$

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

$$\mathbf{X}(t) = \begin{bmatrix} \mathbf{x}(t) - \mathbf{x}_p \\ \dot{\mathbf{x}}(t) - \dot{\mathbf{x}}_p \\ \ddot{\mathbf{x}}(t) - \ddot{\mathbf{x}}_p \end{bmatrix} \tag{19}$$

For each constraint function $f_c(\cdot)$, a sufficient condition for satisfying Eq. (11) is that

$$\left(\frac{\partial f_c}{\partial C_{ik}} \right)^2 + f_c(C_{ik}) \left(\frac{\partial^2 f_c}{\partial C_{ik}^2} \right) \geq 0 \tag{20}$$

for each of the parameters C_{ik}. Since $(\partial f_c / \partial C_{ik})^2$ is clearly non-negative, and $f_c \geq 0$ by definition, we are restricted to a class of constraint functions for which $(\partial^2 f_c / \partial C_{ik}^2) \geq 0$. Evaluating this derivative gives

$$\frac{1}{2} \frac{\partial^2 f_c}{\partial C_{ik}^2} = \left(\frac{\partial g}{\partial \mathbf{X}} \frac{\partial \mathbf{X}}{\partial C_{ik}} \right)^2 + \left[g(\mathbf{X}) - g_0 \right] \left(\frac{\partial^2 g}{\partial \mathbf{X}^2} \right) \left(\frac{\partial \mathbf{X}}{\partial C_{ik}} \right)^2 \geq 0 \tag{21}$$

which is non-negative if $\partial^2 g/\partial \mathbf{X}^2 \geq 0$, since all other terms are non-negative, including $[g(\mathbf{X}) - g_0]$ for an active constraint violation. Therefore, any constraint $g(\mathbf{X})$ that is a non-concave function of the states, derivatives, and/or acceleration defined by $\mathbf{X}(t)$ is allowable. In fact, if the matrix $\partial^2 g/\partial \mathbf{X}^2$ is strictly positive definite (i.e. $\mathbf{z}^T \left[(\partial^2 g/\partial \mathbf{X}^2\right] \mathbf{z} > 0$ for any vector $\mathbf{z} \in R^{dim(\mathbf{X})}$), then $(\partial^2 J')/(\partial C_{ik}^2)$ will also be strictly positive definite, and a gradient based search would encounter $\partial J'/\partial C_{ik} = 0$ only at $C_{ik} = C_{ik}^*$.

To summarize, any constraint function of the form

$$g(\mathbf{X}) \leq g_0 \tag{22}$$

with $\partial^2 g/\partial \mathbf{X}^2 \geq g_0$ will satisfy the conditions required by Eq. (8) for a successful gradient-based trajectory optimization. The requirements on the mathematical form of the constraints are not too restrictive. Some example constraint functions that could easily be structured to satisfy this form include bounds on the constrained volume (walls), ellipsoid or more complex convex shapes within the environment, cylinders, planes of any orientation, maximum speeds or accelerations, maximum path curvature, moving obstacles, etc.

II.D. Boundary Conditions and Projected Gradient Search

With the cost function defined by Eq. (9) and the constraints formulated to fit the conditions of Eq. (11), the negative gradient of J with respect to the parameters C_{ik}, that is $-\partial J/\partial C_{ik}$, is always directed towards J_{min} and a descent gradient search will find the optimal trajectory. However, to assure that admissible sub-optimal solutions are generated 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 dramatically reduce the degrees of freedom in the search space.

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

$$\mathbf{X}_{BC} = \mathbf{P}_{BC}\mathbf{C} \tag{23}$$

or

$$\begin{bmatrix} x(t_0) & y(t_0) & z(t_0) \\ \dot{x}(t_0) & \dot{y}(t_0) & \dot{z}(t_0) \\ x(t_f) & y(t_f) & z(t_f) \\ \dot{x}(t_f) & \dot{y}(t_f) & \dot{z}(t_f) \end{bmatrix} = \begin{bmatrix} P_1(-1) & P_2(-1) & \cdots & P_N(-1) \\ \dot{P}_1(-1) & \dot{P}_2(-1) & \cdots & \dot{P}_N(-1) \\ P_1(1) & P_2(1) & \cdots & P_N(1) \\ \dot{P}_1(1) & \dot{P}_2(1) & \cdots & \dot{P}_N(1) \end{bmatrix} \mathbf{C} \tag{24}$$

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 the partitioning the $\mathbf{C}$ matrix as $\mathbf{C}_{N\times 3} = \mathbf{C}_{\perp_{N\times 3}} + \mathbf{C}_{|||_{N\times 3}}$, such that

$$\mathbf{X}_{BC} = \mathbf{P}_{BC}\mathbf{C}_{N\times 3} = \mathbf{P}_{BC}\left\{\mathbf{C}_{\perp_{N\times 3}} + \mathbf{C}_{|||_{N\times 3}}\right\} \tag{25}$$

where

$$\mathbf{P}_{BC}\mathbf{C}_{\perp_{N\times 3}} = 0 \tag{26}$$

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

$$\delta\mathbf{C}_\perp = \left[\mathbf{I} - \mathbf{P}_{BC}^T\left(\mathbf{P}_{BC}\mathbf{P}_{BC}^T\right)^{-1}\mathbf{P}_{BC}\right]\delta\mathbf{C} \tag{27}$$

where

$$\delta\mathbf{C} = -\alpha\left[\partial J/\partial C\right] + \beta \tag{28}$$

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 and $\delta\mathbf{C}_\perp$ 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 and loosely specifying the form of constraint functions, it becomes possible to transform a highly nonlinear and complex optimization problem into one that is convex in the new parameter space. Then by starting the optimization with a 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. Then by searching a limited subspace with reduced degrees of freedom, it can converge on the optimal solution very quickly as well.

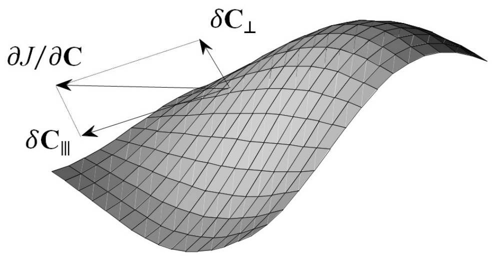

Figure 2. Gradient projection algorithm - reduces search space and assures boundary conditions.

III. Simulation Results

The ASTRO algorithm developed in this work has been tested in numerical simulations in a three dimensional scenario with multiple constraints for the trajectory optimization, as illustrated in Fig. 3. The boundary conditions are

$$\begin{aligned}
\mathbf{x}(t_0) &= \begin{bmatrix} 0 & -0.5 & 0 \end{bmatrix} \text{m} \\
\dot{\mathbf{x}}(t_0) &= \begin{bmatrix} 0 & 0 & 0 \end{bmatrix} \text{m/s} \\
\mathbf{x}(t_f) &= \begin{bmatrix} 0 & 0.5 & 0 \end{bmatrix} \text{m} \\
\dot{\mathbf{x}}(t_f) &= \begin{bmatrix} 0 & 0 & 0 \end{bmatrix} \text{m/s}
\end{aligned}$$

with $t_f = 100$ sec. In this scenario, 6 spheres and cylinders are used to create challenging geometrical constraint functions along the path from $\mathbf{x}(t_0)$ to $\mathbf{x}(t_f)$.

Perhaps a surprising result of the parameterization is that physical obstacles and complex geometry in the trajectory space cannot trap the algorithm in apparent local minima, since in the higher dimensions of the parameterized trajectory the search space is assured to be convex. In this scenario, four constraining cylinders are used to form an enclosed region that the initial guess trajectory passes through. Two spheres provide obstacles that must be avoided to find an admissible solution. 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 inside 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. Another nonlinear optimization approach would find its way around the spheres, but may get caught in a local minima with the trajectory passing through the center of the cylinders. As demonstrated in Fig. 3, with successive steps of the ASTRO algorithm the path jumps beyond the cylinders and ultimately finds the globally shortest path. This is possible because the higher dimensional optimization in the space of Legendre polynomial coefficients is designed to be convex and has no such local minima in the transformed search space.

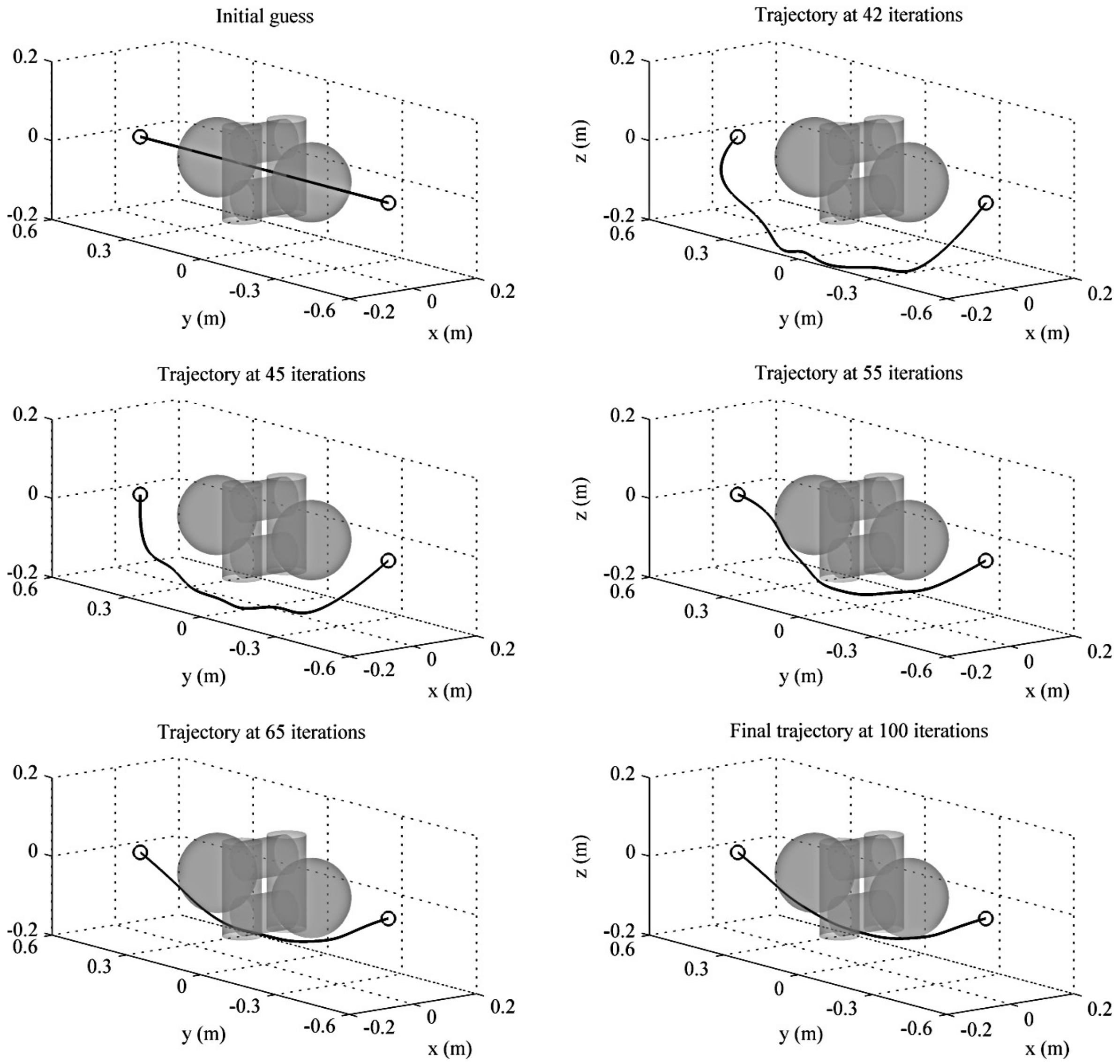

Figure 3. Successive steps of the ASTRO algorithm from initial guess to final solution, demonstrating that the transformation to a higher dimensional convex search space enables the ASTRO algorithm to find the global minimum despite complex nonlinear geometrical constraints.

IV. ISS On-Orbit Experimental Results

The ASTRO algorithm has been experimentally validated on the SPHERES (Synchronized Position Hold, Engage, Reorient Experimental Satellites)[22] research facility aboard the ISS. 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,[23,24] and attitude and flight path control in the presence of uncertainties,[25] 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.

IV.A. Experimental Setup

The SPHERES facility operates three nanosatellites aboard the International Space Station. Each satellite 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 satellites within their operating volume inside ISS.[26, 27] The communication system uses a single TDMA based RF channel to communicate its state to neighboring spacecraft; the timing of the TDMA is controlled by the user interface (GUI) which operates in an ISS laptop. Figure 4 shows a picture of three SPHERES satellites aboard the ISS. The dynamics of each spacecraft are well approximated by a double integrator model.

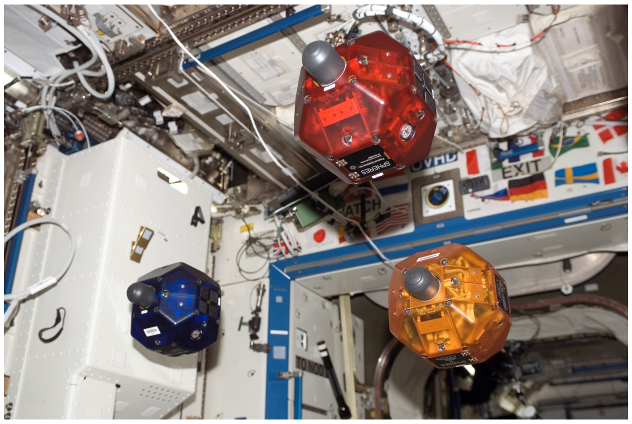
Figure 4. Picture of three SPHERES satellites performing a test onboard the ISS. (Photograph credit: NASA-SPHERES).

ASTRO research used the SPHERES Engineering GUI (instead of the GUI regularly used aboard ISSThe GUI regularly used aboard the ISS provides limited information about the satellites intended primarily for operation, but not sufficient information for direct iterative research.). 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) ability to see raw communication packets.

The state and debug information were essential to enable successful research aboard the ISS. Yet, it was not sufficient, since it was not possible to interpret the ASTRO paths when presented solely through numerical data in debug vectors. Further, all 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 satellites.

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 satellites.[28] The interfaces include access to real-time threads for estimation, control, and autonomy. However, the DSP has limited processing capabilities (it operates at 167MHz, has 16MB or 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 programing aboard the ISS, and allow the use of MATLAB through a real-time distributed-computing environment.

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 satellites. The calls to MATLAB were done via multi-threaded calls, allowing the GUI to maintain the 5Hz TDMA communications link while MATLAB ran the ASTRO algorithm. This allowed MATLAB to have longer periods of processing time.

IV.B. Real-time Traffic Control Model for Distributed Satellite Systems

The resulting distributed computing system created a multi-frequency environment where the SPHERES satellites 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 (which could take up to 10 seconds), (3) show visualization data, and (4) send the new path target points to the satellite. The data are transmitted by the GUI on the next SPHERES TDMA communications cycle. In parallel, the SPHERES satellites and GUI implemented a 1Hz PID control loop which had been proven to work in prior SPHERES tests aboard the ISS. As with all SPHERES tests, the satellites provided their own estimation utilizing an Extended Kalman Filter (EKF) which merges IMU (20Hz) and global metrology ultrasound time-of-flight information (5Hz).

The SPHERES communications system automatically queries the EKF at 5Hz and downloads that data to the SPHERES GUI via a wireless link. Therefore, the MATLAB ASTRO implementation had state information available at up to 5Hz. However, the implementation of the ASTRO algorithm is not designed as a periodic process. Rather, the implementation will attempt to obtain an optimal solution, but if a maximum period is reached, a sub-optimal trajectory is returned. To maintain a minimum path update frequency, the implementation of ASTRO returns updates at a fixed time limit (regardless of feasibility), but may return optimal solutions in less time.

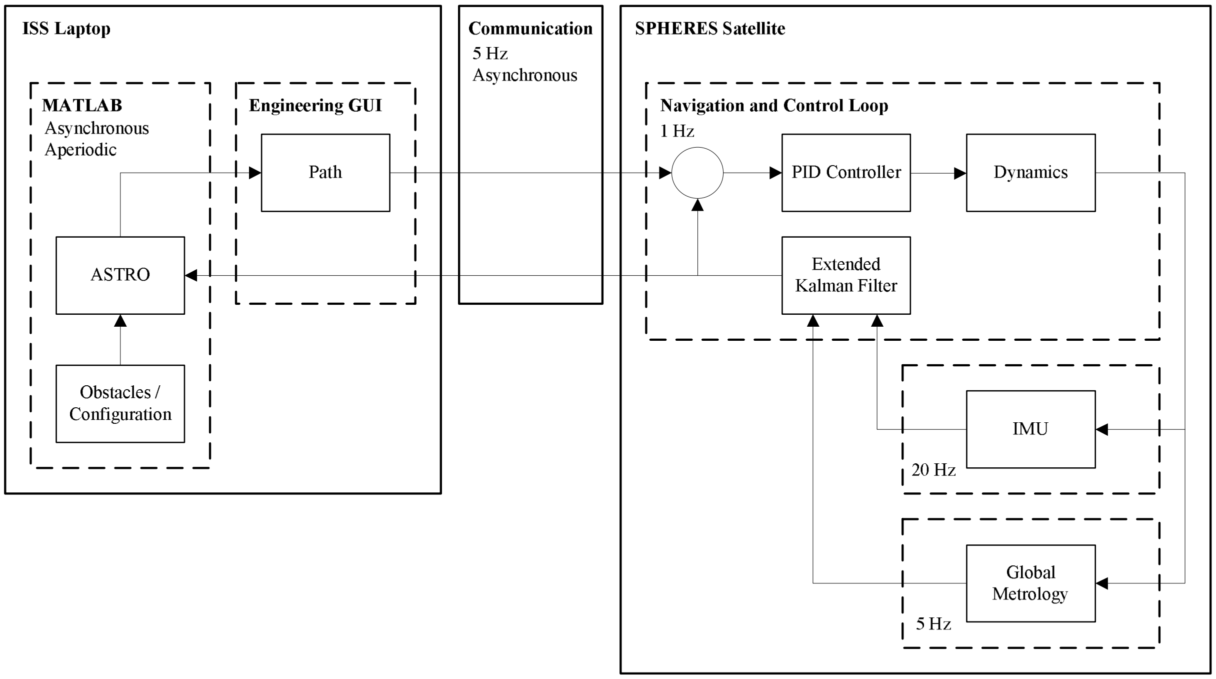

Figure 5. SPHERES ASTRO implementation system overview.

IV.C. Solver Implementation

For demonstration aboard the ISS, the ASTRO algorithm was partially implemented using the MATLAB nonlinear optimization routine *fmincon*. 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 Equation 3. The solver was configured to use the built-in medium-scale Sequential Quadratic Programming (SQP) method instead of the ASTRO gradient projection approach. SQP performs a similar search within the feasible space of the boundary constraints.

Since the ISS session, the full ASTRO algorithm with projected gradient descent has been implemented in both interpreted MATLAB and C-code versions. Combining the subspace descent with a BFGS[29,30,31,32] quasi-Newton descent method results in a 10-20x speedup over *fmincon* for equivalent problems. Compiling the algorithm produces as much as a 60x speedup.

IV.D. 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 satellite;
- the order of the Legendre polynomial series expansion; and
- a set of five constraints which could be activated or deactivated, each satisfying Eq. (22):

  planes restricting the satellite 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 session's 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 vehicles, (2) validate the capability of ASTRO to function as an on-line path planner starting from arbitrary initial conditions, and (3) determine the limitations of ASTRO, in number of constraints, complexity, and speed.

IV.E. 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 satellite acceleration was constrained to 0.05 m/s^2. One complete optimization loop was executed over 17 seconds to produce an optimal feasible trajectory. After convergence, the GUI sequentially transmitted targets to the satellite, indexing the trajectory with total elapsed time. Figure 6 displays the trajectory as estimated by the SPHERES global metrology system in comparison to the reference path generated from the solver. The satellite successfully navigates the edge of the obstacle with close tracking of the reference trajectory.

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 initial condition of the satellite as one of the boundary conditions. After the solver converged or a time limit expired, a single target was transmitted to the satellite, and the optimization was restarted. Figure 7 shows a repeat of the initial validation test with the online planning method enabled. In comparison to the first test, the satellite 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 the targeting of waypoints near to the satellite trajectory. In this case a small residual velocity perpendicular to the trajectory takes a long time to remove because the waypoint target is repeatedly moved closer to the satellite as it drifts away. 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. Figure 8 displays trajectories created by the ASTRO solver as the satellite moved to different positions in the work area. Each initial condition starting from a feasible location resulted in a feasible trajectory plan.

In many cases, the actual trajectory followed by the satellite varied significantly from the desired paths created by the solver due to an important implementation error. The single target transmitted at the end

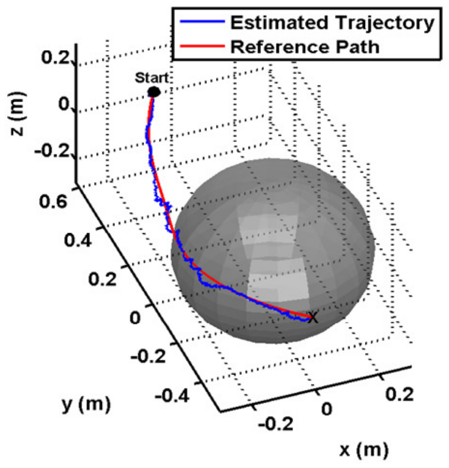

Figure 6. The first on-orbit demonstration validated the distributed architecture by sending a sequence of targets to the satellite based on a reference path generated from a single complete optimization cycle.

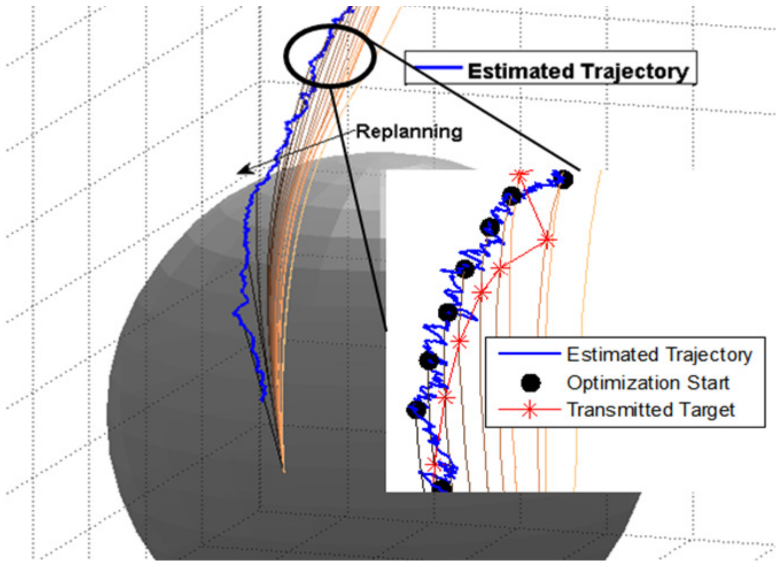

Figure 7. Repeated optimization applied to the initial validation test with darker colors showing later trajectories. A gradual drift in the trajectory can occur if the targets are moved close to the satellite state at each iteration.

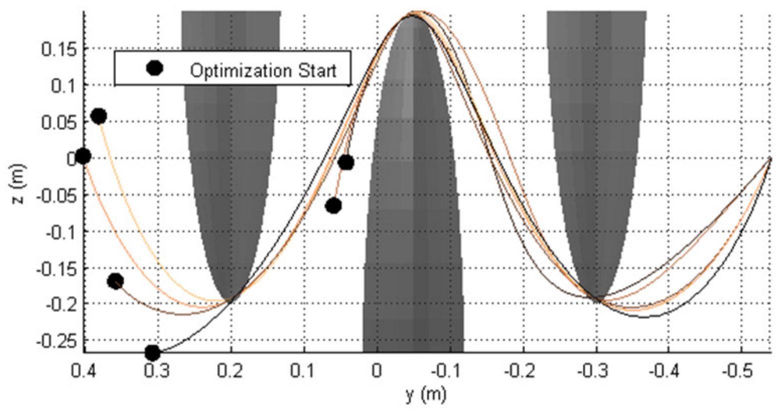

Figure 8. As the satellite moved to several locations, repeated optimizations produced feasible trajectories in a complex obstacle field.

of each optimization cycle was selected by indexing the new trajectory at the elapsed computation time plus a small offset, assuming that the satellite had been following a similar trajectory up to that time. The intended behavior is illustrated in Fig. 9.

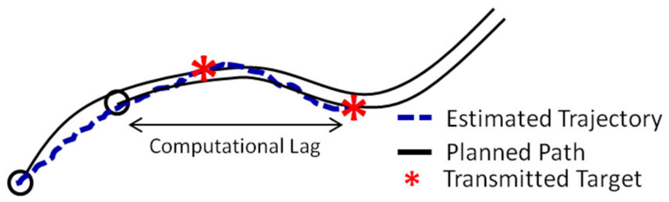

Figure 9. The intended result of the computation lag compensation was to select a target ahead of the current position but near to the current trajectory.

In the event that the two planned trajectories differed significantly or the satellite deviated from the planned path, the compensation mechanism could cause an unstable oscillation in the selection of targets. Figure 10 shows the same scenario as Fig. 8 with the transmitted targets and true trajectory overlaid. In the first iteration, the solver produces the trajectory starting in the upper left corner as the satellite drifts moves downward. As it reaches the second point, the optimization completes, and it receives target 1. While in transit, the GUI optimization starts at the second point, eventually producing target 2. The oscillation repeats with the satellite traversing through the obstacle in the process.

V. Conclusion

This research accomplished two objectives: the development of a real-time path planning approach for operation in the presence of complex 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 identifies optimal trajectories for a free-flying robot to navigate between 3D boundary conditions within surroundings that include numerous obstacles (generalized shapes) and constraints (geometric and performance limitations). This approach uses a unique transformation that effectively changes a complex optimization problem into one with a positive definite cost function that allows high convergence rates even with complex geometries. A projected gradient technique is also used to immediately find admissible solutions. Boundary conditions and constraint avoidance are assured from the very first sub-optimal path to the final optimal solution. This assures successful real-time trajectory solutions even if computation time is limited.

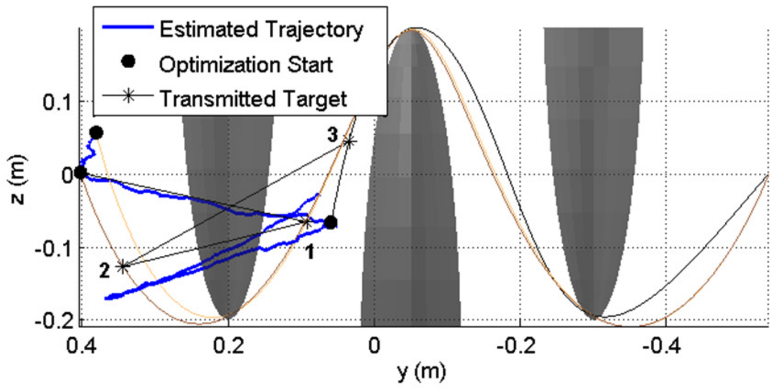

Figure 10. With a large computational lag, transmitting a single target can result in significant deviations from the planned path if the target is not chosen correctly. In this test, the targets oscillate back and forth across the obstacle at each solver iteration.

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 ASTRO algorithm performed successfully on the ISS for single-planning tests. Dynamic re-planning cases exposed some features of the implementation that need improvement. Future improvements will include: (1) transmitting the full polynomial path to the robot, rather than one waypoint at a time. This was an unfortunate limitation of the hardware and software onboard at the time, (2) re-computing trajectories only when conditions have changed (e.g. dynamic obstacles or large disturbances), (3) seeding the ASTRO initial guess from the previous solution (this avoids large trajectory changes due to multiple equivalent solution paths), and (4) using the predicted state from the trajectory and the estimated computation time as the initial condition when re-planning for dynamic obstacles.

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]McCamish, S., Romano, M., Nolet, S., Edwards, C., and Miller, D. W., "Testing of Multiple-Spacecraft Control on SPHERES During Close-Proximity Operations," *Journal of Spacecraft and Rockets*, Vol. 46, No. 6, 2009, pp. 1202–1213.

[2]Taur, D.-R., Coverstone-Carroll, V., and Prussing, J. E., "Optimal Impulsive Time-Fixed Orbital Rendezvous and Interception with Path Constraints," *Journal of Guidance, Control, and Dynamics*, Vol. 18, No. 1, 1995, pp. 54–60.

[3]Munoz, J. and Fitz-Coy, N., "Rapid Path-Planning Options for Autonomous Proximity Operations of Spacecraft," *AIAA Guidance, Navigation, and Control Conference*, Toronto, Ontatio, Canada, 2010, AIAA paper 2010-7667.

[4]Ulybyshev, Y., "Trajectory Optimization for Spacecraft Proximity Operations with Constraints," *AIAA Guidance, Navigation, and Control Conference*, Portland, Oregon, 2011, AIAA paper 2011-6629.

[5]Hadaegh, F. Y., Kim, Y., and Mesbahi, M., "Dual-Spacecraft Formation Flying in Deep Space: Optimal Collision-Free Reconfigurations," *Journal of Guidance, Control, and Dynamics*, Vol. 26, No. 2, 2003, pp. 375–379.

[6]Jezewski, D. J., "Optimal Rendezvous Trajectories Subject to Arbitrary Perturbations and Constraints," *AIAA/AAS*

Astrodynamics Conference, Hilton Head Island, South Carolina, 1992, AIAA paper 1992-4507.

[7]Haufler, B. R., Jezewski, D. J., and Mulder, T. A., "Operational Constraints in Optimal, Impulsive, Rendezvous Trajectories," *AAS /AIAA Spaceflight Mechanics Meeting*, Pasadena, California, 1993, AAS paper 93-140.

[8]Hadaegh, F. Y. and Singh, G., "Collision Avoidance Guidance for Formation-Flying Applications," *AIAA Guidance, Navigation, and Control Conference and Exhibit*, Montreal, Quebec, Canada, 2001, AIAA paper 2001-4088.

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

[10]Bertsimas, D. and Tsitsiklis, J., *Introduction to Linear Optimization*, Athena Scientific, 1997.

[11]Breger, L. and How, J. P., "Safe Trajectories for Autonomous Rendezvous of Spacecraft," *Journal of Guidance, Control, and Dynamics*, Vol. 31, No. 5, 2008, pp. 1478–1489.

[12]Luo, Y., Lei, Y., and Tang, G., "Optimal Multi-Objective Linearized Impulsive Rendezvous," *Journal of Guidance, Control, and Dynamics*, Vol. 30, No. 2, 2007, pp. 383–389.

[13]Luo, Y., Lei, Y., and Tang, G., "Optimal Multi-Objective Nonlinear Impulsive Rendezvous," *Journal of Guidance, Control, and Dynamics*, Vol. 30, No. 4, 2007, pp. 994–1002.

[14]Ranieri, C., "Path-Constrained Trajectory Optimization for Proximity Operations," *AIAA/AAS Astrodynamics Specialist Conference and Exhibit*, Honolulu, Hawaii, 2008, AIAA paper 2008-6275.

[15]Alizadeh, F. and Goldfarb, D., "Second-Order Cone Programming," *Mathematical Programming*, Vol. 95, No. 1, 2003, pp. 3–51.

[16]Boyd, S. and Vandenberghe, L., *Convex Optimization*, Cambridge Univ. Press, 2004.

[17]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.

[18]Y. Nesterov, Y. and Nemirovsky, A., *Interior-Point Polynomial Algorithms in Convex Programming*, Society for Industrial and Applied Mathematics, 1994.

[19]Y. Wang, Y. and Boyd, S., "Fast Model Predictive Control Using Online Optimization," *IEEE Transactions on Control Systems Technology*, Vol. 18, No. 2, 2010, pp. 267–278.

[20]Mattingley, J. and Boyd, S., "Real-Time Convex Optimization in Signal Processing," *IEEE Signal Processing Magazine*, Vol. 27, No. 3, 2010, pp. 50–61.

[21]Chamitoff, G. E., "Autonomous Guidance for the Recovery and Landing of a Remotely Piloted Vehicle," *IFAC Aerospace Control Conference*, Palo Alto, California, 1994.

[22]Mohan, S., Saenz-Otero, A., Nolet, S., Miller, D. W., and Sell, S., "SPHERES Flight Operations Testing and Execution," *Acta Astronautica*, Vol. 65, No. 7, 2009, pp. 1121–1132.

[23]Saenz-Otero, A., Katz, J. G., and Miller, D. W., "SPHERES Demonstrations of Satellite Formations aboard the ISS," *32nd Annual AAS Guidance and Control Conference*, Breckenridge, Colorado, 2009, AAS Paper 09-011.

[24]Ramirez-Riberos, J. L., Pavone, M., Frazzoli, E., and Miller, D. W., "Distributed Control of Spacecraft Formations via Cyclic Pursuit: Theory and Experiments," *Journal of Guidance, Control, and Dynamics*, Vol. 33, No. 5, 2010, pp. 1655–1659.

[25]Fejzic, A., Nolet, S., Breger, L., How, J. P., and Miller, D. W., "Results of SPHERES Microgravity Autonomous Docking Experiments in the Presence of Anomalies," *59th International Astronautical Congress*, Glasgow, Scotland, 2008.

[26]Nolet, S., "The SPHERES Navigation System: from Early Development to On-Orbit Testing," *AIAA Guidance, Navigation and Control Conference*, Hilton Head, South Carolina, 2007, AIAA Paper 2007-6354.

[27]Nolet, S., *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.

[28]Saenz-Otero, A. and Miller, D. W., "The SPHERES ISS Laboratory for Rendezvous and Formation Flight," *5th ESA International Conference on Spacecraft Guidance, Navigation and Control Systems*, Frascati, Italy, 2002.

[29]Broyden, C. G., "The Convergence of a Class of Double-rank Minimization Algorithms," *Journal of the Institute of Mathematics and its Applications*, Vol. 6, 1970, pp. 76–90.

[30]Fletcher, R., "A New Approach to Variable Metric Algorithms," *Computer Journal*, Vol. 13, 1970, pp. 317–322.

[31]Goldfarb, D., "A Family of Variable Metric Updates Derived by Variational Means," *Mathematics of Computing*, Vol. 24, 1970, pp. 23–26.

[32]Shanno, D. F., "Conditioning of Quasi-Newton Methods for Function Minimization," *Mathematics of Computing*, Vol. 24, 1970, pp. 647–656.