Smooth Trajectory Generation on SE(3) for a Free Flying Space Robot

Michael Watterson[1], Trey Smith[2], and Vijay Kumar[1]

Abstract— **We propose a new optimal trajectory generation technique on SE(3) which avoids known obstacles. We leverage techniques from differential geometry and Lie algebra to formulate a cost functional which is intrinsic to the geometric structure of this space and makes physical sense. We propose an approximation technique to generate trajectories on the subgroup SO(3) and use Semidefinite Programming (SDP) to approximate an NP-Hard problem with one which is tractable to compute. From this trajectory on the subgroup, the trajectory generation on the other dimensions of the group becomes a Quadratic Program (QP). For obstacle avoidance, we use a computational geometric technique to decompose the environment into overlapping convex regions to confine the trajectory. We show how this motion planning technique can be used to generate feasible trajectories for a space robot in SE(3) and describe controllers that enable the execution of the generated trajectory. We compare our method to other geometric techniques for calculating trajectories on SO(3) and SE(3), but in an obstacle-free environment.**

I. Introduction

The set of configurations of a robot in 3D space is the special Euclidean group (SE(3)). This group has 6 degrees of freedom, 3 from translation motions and 3 from the orientation, which is itself the special orthogonal group SO(3). Trajectory generation in all 6 degrees is used in the areas of robots such as end effector manipulation ([4]), path generation for underwater robots ([14]) and space satellites ([22]). Many existing methods do not consider the geometric structure of this manifold or optimality by just interpolating between a series of intermediate points in translations and rotations separately.

Through a differential geometric lens, there is extensive literature on how to define metrics (inner products) and integration on SE(3). [30] proposes multiple metrics and connections for SE(3) and demonstrates that there is no bi-invariant metric on this manifold. [9] shows how to integrate functions on Lie groups and explicitly shows how to transform between derivatives in common coordinate choices for SO(3) and SE(3). [1] calculates the necessary conditions for a minimum jerk trajectory of SE(3) from calculus of variations on the manifold and proposes an approximation method based on projecting curves from the general linear group GL^+ to SE(3). [23] suggests methods for optimally interpolating between curves on just the rotation group using cubic interpolation on multiple parameterizations of SO(3).

There is a lot of literature on how to generate curves on flat spaces such as $\mathbb{R}^n$ with applications to robotics. For collision avoidance, it is necessary to generate optimal trajectories inside a non-convex configuration space. [12],[10], and [29] propose methods for generating optimal curves for similar differentially flat systems using mixed integer and quadratic programming techniques with a objective function which minimizes the snap or jerk of the trajectory.

A more general optimization: Semidefinite Programming (SDP) is often used in optimal control problems to generate Lyapunov functions for dynamical systems ([24]). They can also be used to find approximations for the minimum of multivariate polynomials, which is NP-hard to solve exactly. It has been shown that this approximation is close for positive definite polynomials ([21]), and exact for quadratic polynomials.

The control of a six degree of freedom holonomic robot with on board propulsion jets is mathematically equivalent to grasp manipulation of objects with point contacts with friction. From this literature, it is well known that it takes at least seven jets to stabilize our robot ([18]), but there are more to distribute the load between them. We use a controller which is stable and has region of convergence proved by [5].

There is rich literature on spacecraft guidance and control, including multi-robot trajectory planning, optimization under various cost metrics, and obstacle avoidance [26], [19]. Most of this work (with a few exceptions [11]) focuses solely on prior planning, without regard to real-time obstacle sensing and replanning. In contrast, our algorithm, inspired by recent developments in unmanned aerial vehicles, uses an obstacle model and optimization techniques that have been shown to enable both autonomous obstacle detection with COTS depth sensors and rapid trajectory replanning at multiple-Hz rates on compute-constrained platforms [27]. Thus, our approach should be able to support efficient real-time avoidance of unexpected obstacles, but this work does not focus on the integration with sensor data.

The main contributions of this paper are:

1) An SDP for SO(3) minimum jerk trajectory generation
2) An SDP with a QP for SE(3) minimum jerk trajectory generation with obstacle avoidance
3) The extension of the forward/backward relaxation algorithm of [15] to arbitrary polyhedrons
4) The application of these techniques to our specific free-flying platform.

This paper is organized in the order of these contributions. First in Section II, we define inputs and outputs of our trajectory generation technique. In Section III, we introduce the necessary Lie algebra and differential geometry needed

[1] Michael Watterson and Vijay Kumar are with the GRASP Lab, University of Pennsylvania, 3330 Walnut Street, Phialdelphia PA, USA. email: {wami, kumar}@seas.upenn.edu

[2] Trey Smith is with the NASA Intellegent Robotics Group, NASA Ames Research Center, Moffet Field CA, USA. email: trey.smith@nasa.gov

Fig. 1: Picture of Astrobee: A free flying robotic platform being designed for zero gravity flight inside of the International Space Station

to mathematically define an optimal trajectory on SE(3) as well as the required mathematics to compare our method to SO(3) trajectory generation techniques. In Section IV we first present our method to generate trajectories on SO(3) and then SE(3). In our SE(3) trajectory generation section, we assume a given decomposition of the environment, which we explain in Section V. We then describe the control for our system in Section VI, analyze our results in Section VII and Section VIII. We make a small assumption in Section V which we prove in an appendix section (X).

II. Problem Formulation

We seek to find a sufficiently smooth, optimal trajectory $A(t)$ for a rigid body from a start to goal pose specified by elements of SE(3) in an environment with obstacles. The required smoothness is determined by the system dynamics. We define optimal with respect to the jerk functional in Equation 12 because it is general, independent of the actuator dynamics, and minimizes the change in actuator force.

For higher order functionals, this same methodology can be applied, but the number of optimization variables would grow. For a different system, snap can be used as in [10]. If we were trying to minimize energy for our system, we could use acceleration.

Since our metric in Section III is left invariant, we can translate the trajectory without changing the jerk. Therefore, without loss of generality, we assume the trajectory starts from the identity element of SE(3).

Our approach separately plans translation and rotation trajectories, then combines them to form the overall 6-DOF trajectory. Decoupling makes sense for platforms that, like Astrobee, are nearly spherically symmetric and have similar thrust capabilities on all axes.

We require that our trajectory is not in collision with obstacles. Relying on near-spherical symmetry, we can represent collision avoidance as a constraint on only the positional part of the trajectory. As in other methods [10] [29] [27], we can replace checking collision of our robot, by a point model with expanded obstacles. We inflate the obstacles in our environment by a margin δ

$$\delta = \delta_r + \delta_n \tag{1}$$

Where δ_r is the radius of the robot, and δ_n is a conservative parameter to account for errors in obstacle position measurements and errors in robot pose estimation and control.

The permitted area for our robot is specified as a set of keep-in polyhedrons and a set of keep-out polyhedrons. The robot is required to be in at least one keep-in polyhedron and outside all keep-out polyhedrons at all times. These polyhedrons are subsets of $\mathbb{R}^3$ which can be specified as vertices, edges, and faces or as a intersection of half-spaces. We assume that we can freely switch between these two representations ([28]). The free space that the robot can traverse is the difference between the union of all the keep-in polyhedrons and the union of all the keep-out obstacles.

$$P^{\text{free}} = \bigcup_i P_i^{\text{keep-in}} - \bigcup_j P_j^{\text{keep-out}} \tag{2}$$

III. Background

A. A Note on Notation

Table I lists frequently used variables and functions. Since some variables have many indices, we choose to use both sub-scripts and super-scripts to index these variables. For brevity, when the intended range of a summation is clear from context, we will abbreviate it. For example, $\sum\limits_{i,j} a_{ij}$ would be $\sum\limits_{i=1}^{n}\sum\limits_{j=0}^{m} a_{ij}$ if i ranges between 1 and m, and if j ranges between 0 and m. The trajectory of the robot A, position of the robot R, angular velocity ω and exponential coordinates ξ are all functions of time. Most of the time we suppress the explicit time parameterization $\omega(t)$ and will just write ω. In addition, $\dot{\omega}$ represents the time derivative of the variable ω.

B. Lie Groups

SO(3) is the group of all rotations in 3 dimensional space. It is also the set of all 3×3 orthogonal matrices with positive determinant:

$$\text{SO}(3) = \{R \in \mathbb{R}^{3\times3} | R^T R = \mathcal{I}, det(R) = 1\} \tag{3}$$

The special Euclidean group SE(3) is the group composed of all rigid body transformation in 3-dimensional Euclidean space. The group product is defined as the composition of two rigid body transformation.

$$A = \begin{bmatrix} R & d \\ 0 & 1 \end{bmatrix} \quad R \in \text{SO}(3) \quad d \in \mathbb{R}^3 \tag{4}$$

If we write an element is a matrix of the form of Equation 4, the group product for both SO(3) and SE(3) can be expressed as matrix multiplication.

$$A \cdot B = AB \quad A, B \in \text{SO}(3) \text{ or SE}(3) \tag{5}$$

It is often useful to parameterize these groups by their Lie algebras. It can be shown for the linear operators $\hat{(.)}_{\text{so}(3)} : \mathbb{R}^3 \to \text{so}(3)$ and $\hat{(.)}_{\text{se}(3)} : \mathbb{R}^6 \to \text{se}(3)$ that:

$$\exp \hat{g} \in \text{SO}(3) \;\; \forall g \in \mathbb{R}^3 \tag{6}$$

$$\exp \hat{g} \in \text{SE}(3) \;\; \forall g \in \mathbb{R}^6 \tag{7}$$

We drop the group subscript on the hat operator and allow the domain of the argument to distinguish between the two.

$R \in \mathrm{SO}(3)$	Rotation of robot expressed in world frame	$\mathcal{I} \in \mathrm{SO}(3)$	Identity matrix
$d \in \mathbb{R}^3$	Position of the robot expressed in world frame	$J : (A(t) : [0, T] \to \mathrm{SE}(3)) \to \mathbb{R}$	Jerk functional
$A \in \mathrm{SE}(3)$	Robot pose expressed in world frame	$\xi \in \mathbb{R}^3$	Exponential coordinates of R
$v \in \mathbb{R}^3$	Linear velocity of the robot in world frame	$P \subset \mathbb{R}^3$	Polyhedron
$\omega \in \mathbb{R}^3$	Angular velocity of the robot in body frame		
$\exp(\cdot) : \mathrm{so}(3) \to \mathrm{SO}(3)$	Exponential map operating a member of so(3)	$\log(\cdot) : \mathrm{SO}(3) \to \mathrm{so}(3)$	Inverse of exponential map
$\exp(\cdot) : \mathrm{se}(3) \to \mathrm{SE}(3)$	Exponential map operating a member of se(3)	$\log(\cdot) : \mathrm{SE}(3) \to \mathrm{se}(3)$	Inverse of exponential map
$\hat{\cdot} : \mathbb{R}^3 \to \mathrm{so}(3)$	Hat function to so(3)	$(\cdot)^\vee : \mathrm{so}(3) \to \mathbb{R}^3$	Inverse of hat function
$\hat{\cdot} : \mathbb{R}^6 \to \mathrm{se}(3)$	Hat function to se(3)	$(\cdot)^\vee : \mathrm{se}(3) \to \mathbb{R}^6$	Inverse of hat function

TABLE I: Table of variables, functions, and operators.

If g_i is the ith component of g, the hat operators are as follows:

$$\hat{g}_{\mathrm{so}(3)} = \begin{bmatrix} 0 & -g_3 & g_2 \\ g_3 & 0 & -g_1 \\ -g_2 & g_1 & 0 \end{bmatrix} \quad \hat{g}_{\mathrm{se}(3)} = \begin{bmatrix} \hat{g}_{\mathrm{so}(3)} & g_{4:6} \\ 0 & 0 \end{bmatrix} \tag{8}$$

Although it is tempting to parametrize elements of SE(3) with respect to the Lie algebra like [20], it is more useful for us to separately consider the rotational (R(t)) and translation (d(t)) parts of the group. This parametrization will be useful for enforcing collision avoidance as well as optimizing the trajectories using a loosely coupled method. For the rotational part, we will use several representations the rotation matrix R and the standard exponential coordinates at the identity: ξ such that $R = \exp \hat{\xi}$. It is often useful to represent a rotation with respect to the integral its angular velocity.

$$R(t) = \mathcal{I} + \int_0^t R(\tau)\hat{\omega}(\tau)d\tau \tag{9}$$

This integral has no known closed form solution, except in the case where ω is of the form $n * f(t)$ for a vector n and a scalar $f(t)$. In that particular case, the solution is:

$$R(t) = \exp(\hat{n} * \int_0^t f(\tau)d\tau) \tag{10}$$

The differential form of Equation 9 is also used with $(.)^\vee$ being the inverse of $\hat{(.)}$, the angular velocity is [20]:

$$\omega = (R^T\dot{R})^\vee \tag{11}$$

C. Connections and Metrics

On a general manifold, the notion of a derivative is extended by the covariant derivative ([6]). $\nabla_V W$ represents the derivative of the vector field W along the direction of V. Like [30], we choose ∇ to be the affine connection such that the derivative is consistent with physical rigid body motion. If $V = \frac{dA}{dt}$ is the velocity vector field, the jerk of the trajectory A is:

$$Jerk = \nabla_V\nabla_V V = \begin{bmatrix} \ddot{\omega} + \frac{1}{2}\omega \times \dot{\omega} \\ \frac{d(\dot{v}+\omega\times v)}{dt} + \omega \times (\dot{v} + \omega \times v) \end{bmatrix} \tag{12}$$

Where v is simply the linear velocity $\dot{d}$ of the trajectory.

To be able to calculate a jerk functional, we also need a metric to define the inner product $\langle V, V\rangle$. We can choose any metric of the form [30]:

$$\langle V, V\rangle = \alpha||V^{1:3}||^2 + \beta||V^{4:6}||^2 \tag{13}$$

This allows us to define our cost functional J for a trajectory on SE(3), which takes a total time of T to execute.

$$J = \int_0^T \langle\nabla_V\nabla_V V, \nabla_V\nabla_V V\rangle dt \tag{14}$$

D. Comparisons

In Section VII-B, we will compare our trajectory generation to other methods for calculating the minimum jerk trajectory that are based on intrinsic techniques and are invariant of coordinate charts on SO(3) and SE(3). Here we will present the equations needed to compare our work. [1] uses an interpolation method on $\mathbb{R}^{12}$ which is then projected back to SE(3). We can compare to this method for small boundary conditions due to the restriction on the domain of the projection map.

To for this comparison, we need to map boundary conditions on ω and d to $A(t)$. That work gives:

$$\dot{A}(t) = A(t)\begin{bmatrix} \hat{\omega} & R^T\dot{d} \\ 0 & 0 \end{bmatrix} \tag{15}$$

A straight forward calculation finds:

$$\ddot{A}(t) = A\begin{bmatrix} \hat{\dot{\omega}} & R^T\ddot{d} - \hat{\omega}R^T\dot{d} \\ 0 & 0 \end{bmatrix} + \dot{A}A^{-1}\dot{A} \tag{16}$$

From [30], the necessary condition for a minimum jerk trajectory is:

$$\begin{aligned} &\omega^{(5)} + 2\omega \times \omega^{(4)} + \tfrac{5}{4}\omega \times (\omega \times \omega^{(3)}) + \tfrac{5}{2}\dot{\omega} \times \omega^3 \\ &+\tfrac{1}{4}\omega \times (\omega \times (\omega \times \ddot{\omega})) + \tfrac{3}{2}\omega \times (\dot{\omega} \times \ddot{\omega}) - (\omega \times \ddot{\omega}) \times \dot{\omega} \\ &-\tfrac{1}{4}(\omega \times \dot{\omega}) \times \ddot{\omega} - \tfrac{3}{8}\omega \times ((\omega \times \dot{\omega}) \times \dot{\omega}) \\ &-\tfrac{1}{8}(\omega \times (\omega \times \dot{\omega})) \times \dot{\omega} = 0 \\ &d^{(6)} = 0 \end{aligned} \tag{17}$$

All the methods for trajectory generation try to find a d and ω which match boundary conditions on both ends of the curves. With these differential equations, we can not solve for a closed form solution. We can compare the methods by numerically calculating the minimum jerk trajectory with some initial condition and plug the resultant initial and final boundary conditions into our method and the methods of [1] and [23].

[23] provides a method for interpolating rotation which are not minimum jerk. The proposed method assumes that the rotational trajectory is parameterized as:

$$R(t) = R_0 \exp(\hat{a}t^3 + \hat{b}t^2 + \hat{c}t) \tag{18}$$

The angular velocity can be expressed in terms of the constants a, b, and c by the relation:

$$\omega = K(at^3 + bt^2 + ct)(3at^2 + 2bt + c) \tag{19}$$

Where $K : \mathbb{R}^3 \rightarrow \mathbb{R}^{3\times3}$ is the right Jacobian of the exponential coordinates as defined by [9]. With this and the conditions on the initial and final rotations and angular velocities, we can solve for a, b, and c by setting up a linear system of equations.

The right Jacobian is given explicitly by [9]:

$$K(\xi) = I - \frac{1-\cos||\xi||}{||\xi||^2}\hat{\xi} + \frac{||\xi|| - \sin||\xi||}{||\xi||^3}\hat{\xi}^2 \tag{20}$$

IV. Optimization Method

A. Rotational Optimization

We can approximate ω as a nth order polynomial $\omega(t) \approx a_0 + a_1 t + a_2 t^2 + ...a_n t^n$ where $a_i \in \mathbb{R}^3$

Since the the zeroth derivative of this approximation is exact at both of the endpoints, the Lagrange error bound can be used to show that there exist an optimal polynomial $p^*(t)$ which deviates from ω by $Err(t)$. $Err(t)$ is bounded by:

$$||Err(t)|| \leq \frac{1}{(n+1)!}(\frac{T}{2})^{n+1} \max_t \omega^{(n+1)}(t) \tag{21}$$

Since we assume inputs to the initial and final conditions to the trajectory, we can eliminate constraints on the rotational part components by using an endpoint constrained basis. $\omega(t) = b_0 q_0(t) + b_1 q_1(t) + ...b_n q_n(t)$ for which q_i are nth order polynomials and $\omega^{(k)}(0) = b_k$ and $\omega^{(k)}(T) = b_{k+\frac{n}{2}}$ for $k = 1..\frac{n}{2}$.

The rotational jerk of the trajectory can now be found as:

$$\nabla_V \nabla_V V^{1:3} = \sum_i b_i \ddot{q}_i + \frac{1}{2}\sum_{i,j}(b_i \times b_j) q_i q_j \tag{22}$$

If we stack all of the components of each b_i and $b_i b_j$ into one large $3(\frac{n^2}{2} + \frac{3n}{2}) \times 1$ vector b, we can write the jerk as:

$$j(t,b) = E(t)Qb \tag{23}$$

Where

$$E(t) = \begin{bmatrix} e(t) & 0 & 0 \\ 0 & e(t) & 0 \\ 0 & 0 & e(t) \end{bmatrix} \tag{24}$$

$$e(t) = \begin{bmatrix} 1 & t & t^2 & ... & t^n \end{bmatrix} \tag{25}$$

And Q can be found by appropriate bookkeeping of the endpoint basis and Equation 22.

The jerk functional can be expressed in polynomial terms of all the b_i if polynomials are integrated in closed form:

$$\begin{aligned} &j(b) = \textstyle\int_0^T ||j(t,b)||^2 dt = \\ &b^T Q^T (\textstyle\int_0^T E^{\mathrm{T}}(t)E(t)dt)Qb = b^{\mathrm{T}} M b \end{aligned} \tag{26}$$

Finding the minimum value of a multivariate quartic polynomial is NP-Hard [24]. However for general polynomials it can be closely approximated by a sum-of-squares (SOS) program:

$$\begin{aligned} \max \quad & \gamma \\ s.t. \quad & j(b) - \gamma \in SOS \end{aligned} \tag{27}$$

With coefficients, this becomes an SDP. For $x = \begin{bmatrix} 1 & b_1 & b_2 & ... & b_n & b_1^2 & b_1 b_2 & ... & b_i b_j \end{bmatrix}^T$ With N being all zeros, except its top left element which is 1 and B_i being a basis for the space of matrices B for which $b^T B b = 0$

$$\begin{aligned} \max \quad & \gamma \\ s.t. \quad & M - \gamma N + \textstyle\sum \beta_i B_i \in \mathrm{SDP} \\ & \beta_i \in \mathbb{R} \end{aligned} \tag{28}$$

B. Boundary Conditions for Rotation

While this method solves for an optimal trajectory based on the end conditions of ω, it does not properly handle the final $R(T)$ condition. Without a solution to equation 9, we cannot directly optimize with this endpoint. However, instead we can compose a second minimum jerk trajectory $R^c(t)$ with our optimal trajectory $R^o(t)$

$$R(t) = R^c(t) R^o(t) \tag{29}$$

If we assume zero endpoint conditions on ω for $R^c(t)$, then the optimal trajectory will be along one axis. Therefore we can use Equation 10 to explicitly solve for $R^c(t)$ algebraically.

C. Translation Optimization

When adding in the full SE(3) optimization we need to incorporate the translational elements of Equation 12 into the jerk functional of Equation 14. In addition, to avoid obstacles, we create a spline with a trajectory segment confined to be within a convex corridor [12][29][27]. The corresponding evaluation of Equation 14 results in polynomial terms which are 6th order with respect to the coefficient variables as opposed to 4th order for just the rotational part. However, the need to have many convex polynomials in the corridor, results in an intractably large amount of variables in the SDP.

To reduce the number of variables, we can reduce the optimization to a loosely coupled optimization, where we first minimize the rotational part of the state and then optimize the translational part. In the absence of collision constraints, we see the optimality conditions, given by Equation 17, are de-coupled between the two parts of the state. Also for a valid choice of the metric such that $\alpha << \beta$ would give the same trajectory if calculated with ω and v being jointly optimized.

We notice with a known ω, the last terms in the jerk of 12 are linear with respect to the translational velocity of the system. Therefore, when we express the translation components of the cost functional J with respect to a polynomial basis we get a quadratic form.

$$\int \langle \nabla_V \nabla_V V^{4:6}, \nabla_V \nabla_V V^{4:6} \rangle dt = \sum_i b_i^T Q_i b_i \tag{30}$$

Where Q_i can be calculated in closed form from integrating out the angular terms of the translational components of the jerk squared. For the purposes of ensuring collision avoidance, we choose to represent the translational part of the state a Bezier basis spline. The Bezier basis has the convenient property which that each curve segment will lie with in the convex hull of its control points and thus confining the trajectory to be inside the convex regions becomes a linear inequality constraint [12].

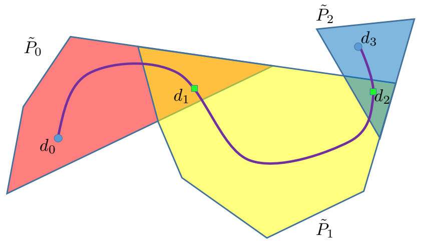

Fig. 2: Confining the positional part of the trajectory into a convex corridor defined by polyhedrons. The start and end points d_0 and d_3 are fixed, while d_1 and d_2 are only specified as continuity constraints. Implicitly, this allows d_1 and d_2 to vary within the orange and green regions respectively.

With respect the control point b_i of the polynomials representing d, this becomes the following quadratic program:

$$\begin{aligned} \min_{b} \quad & \textstyle\sum_i b_i^T H b_i \\ & Cb \leq c \\ & Sb = s \end{aligned} \tag{31}$$

In the example in Figure 2 where there are three polyhedrons, the inequality constraints come from confining the trajectory segment $d_0 d_1$ to be inside $\tilde{P}_0$, $d_1 d_2$ to be inside $\tilde{P}_1$, and $d_2 d_3$ to be inside $\tilde{P}_2$. The equality constraints are from confining d_0 and d_3 to their locations in space and imposing continuity from the 1st to the 3rd time derivative of the spline at the points d_1 and d_2.

V. Decomposition

From the given input environment as a set of polyhedrons, it has been shown that the problem of decomposing into a minimum number of polyhedrons is NP Hard. [8]. However, if we accept $O(n^2)$ polyhedrons in our decomposition, we can do the decomposition in polynomial time [8]. In practice, we do not see this many polyhedrons, especially compared to the $O(r^3)$ number in voxel grid based methods with respect to the discretization parameter r ([27]).

From a set of polyhedrons, we can find a convex corridor to confine a trajectory inside of as in [29]. We can represent each polyhedron P with matrices Λ_i and ζ_i [3]:

$$P_i = \{x \in \mathbb{R}^3 | \Lambda_i x \leq \zeta_i\} \tag{32}$$

Where each row λ_i^k of Λ_i with ζ_i^k represents a face of the ith polyhedron in the corridor.

$$P_i = \{x \in \mathbb{R}^3 | (\lambda_i^k)^T x \leq \zeta_i^k \quad \forall k\} \tag{33}$$

As in [15], we can avoid a non-convex time allocation optimization by allowing the polyhedrons to overlap slightly. We do this by removing faces of smaller polyhedrons border larger polyhedrons. From [2], we can represent this relaxation using set operations on polyhedrons. For P_i and P_{i+1} begin two adjacent polyhedrons in the corridor, we define the boarding row w_i to the pair of constraints.

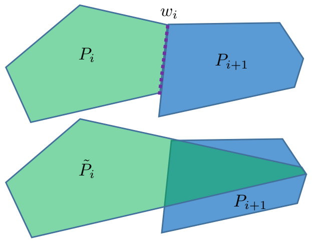

Fig. 3: Adjacent polyhedrons with P_i being expanded into P_{i+1} with face w_i being removed and replaced with the necessary face from P_{i+1}. Depending on the polyhedrons, multiple faces from P_{i+1} may be faces of $\tilde{P}_i$

$$\exists r \in \mathbb{N} \ \exists \gamma \in \mathbb{R} \left| \begin{array}{l} \lambda_i^r = -\gamma \lambda_{i+1}^{w_{i+1}} \\ \zeta_i^r = \gamma \zeta_{i+1}^{w_{i+1}} \end{array} \right. \tag{34}$$

Which then allows us to defined the relaxed polyhedron $\bar{P}_i$ which is a polyhedron without one of its faces.

$$\bar{P}_i = \{x \in \mathbb{R}^3 | (\lambda_i^k)^T x \leq \zeta_i^k \quad \forall k \neq w_i\} \tag{35}$$

Finally we can find the constraints for the polyhedron $\tilde{P}_i$ by adding the overlap from the back face with the previous polyhedron. The intersection of polyhedrons represented by matrices is done using [3] and the intersection can be done using the algorithm for the convex hull of the union of two polyhedrons ([2]). This is valid, because the union of P_i and $(P_{i+1} \cap \bar{P}_i)$ is convex if the face of P_i which borders P_{i+1} is a subset of the boarding face of P_{i+1}. We prove this in Section X.

$$\tilde{P}_i = (P_{i+1} \cap \bar{P}_i) \cup P_i \tag{36}$$

VI. Dynamics and Control

The system dynamics are those given by Euler's equations of motion for a single rigid body.

$$\frac{d}{dt}\begin{bmatrix} r \\ \xi \\ v \\ \omega \end{bmatrix} = \begin{bmatrix} v \\ K(\xi)\omega \\ mF \\ I^{-1}(M - \omega \times I\omega) \end{bmatrix} \tag{37}$$

For our system with propulsion force f_i at each jet located at position r_i with direction of force n_i, the forces and moments are mapped with the G matrix.

$$\begin{bmatrix} F \\ M \end{bmatrix} = \begin{bmatrix} n_1 & n_2 & \dots & n_{12} \\ r_1 \times n_1 & r_2 \times n_2 & \dots & r_{12} \times n_{12} \end{bmatrix} \begin{bmatrix} f_1 \\ f_2 \\ \dots \\ f_{12} \end{bmatrix} \tag{38}$$

$$\nu = \begin{bmatrix} F \\ M \end{bmatrix} = G \begin{bmatrix} f_1 \\ f_2 \\ \dots \\ f_{12} \end{bmatrix} \tag{39}$$

Since G is 6×12, we have multiple sets of jet forces which map to the same total force and moment. Like in [7], we can efficiently resolve this redundancy by formulating a quadratic program to minimize the total actuated force.

$$\begin{aligned} \arg\min_{f} \quad & \sum_i f_i^2 \\ s.t. \quad & \nu = Gf \\ & f \geq 0 \end{aligned} \tag{40}$$

This QP can be solved by simply checking all the corners of the linear constraints ([3]). Since G is a constant matrix for our platform, all matrix inverses can be pre-computed, which makes the calculation take negligible time.

We can use the same controller as [5] which has been proven to be exponential stable for small enough errors.

VII. Results

Within our trajectory generation method, we have made several simplifying assumptions. Firstly that ω is a polynomial as opposed to a sufficiently smooth continuous function. This approximation can be bounded by Equation 21 if the maximum 7th derivative of the true minimum jerk trajectory is known. However we cannot bound this for general trajectories because it is always possible to generate a trajectory from Equation 17 with an arbitrarily high initial 7th derivative. The second approximation is using an SOS optimization to find a the minimum of a multivariate polynomial, which has been shown to be a close approximation ([17]). To judge the quality of our approximation, we compare our method to a variety of existing methods for generating trajectories on SO(3) in the absence of obstacles. To the best of our knowledge, there is no other method which can generate minimum jerk trajectories defined as in Equation 14, while respecting obstacles. In addition, Equation 17, is not the necessary condition for an optimal trajectory with obstacles, therefore we compare our SE(3) trajectory generation method to others only in the case of no obstacles.

A. Implementation

We implemented our algorithms in a combination of python, C++, and MATLAB. We used the Robotics Operating System (ROS) as a communication layer between the different modules ([25]). We used the computational geometry library (CGAL) to compute the decomposition of our environment ([28]). With the Gurobi ([13]) and SeDuMi ([16]) optimizers as back-ends for the QP and SDP respectively.

B. Numerical Comparison

We computed a set of 1 second trajectories from the identity element of SO(3) with random inputs on the initial conditions of the optimal differential equations (17). The initial conditions were drawn from uniform random distributions as follows:

$$\begin{gathered} \omega, \dot{\omega}, \ddot{\omega}, \omega^{(3)}, \omega^{(4)}, \omega^{(5)} \in \\ \mathbb{U}([-\tfrac{\pi}{2}, -\tfrac{\pi}{2}, -\tfrac{\pi}{2}], [\tfrac{\pi}{2}, \tfrac{\pi}{2}, \tfrac{\pi}{2}]) \end{gathered} \tag{41}$$

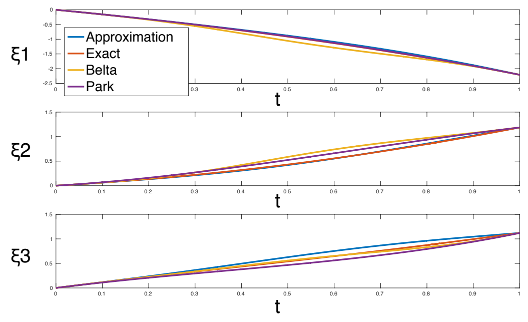

Fig. 4: Comparison of an example set of trajectories on SO(3)

Figure 9, shows an example set of trajectories on SO(3) expressed in terms of exponential coordinates. Our version of the trajectory is closer to the true minimum jerk trajectory than the other trajectory generation methods.

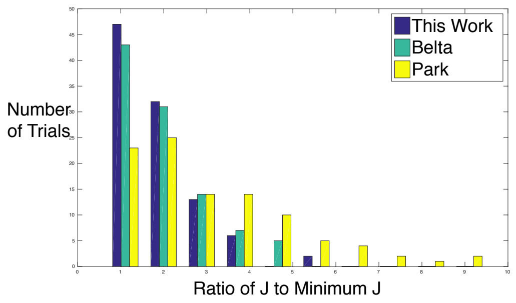

Fig. 5: Comparison of the J of 100 simulated trajecories. $(\alpha = 1, \beta = 0)$

We then computed J using first order numerical integration for each method. We found that all approximation methods varied in accuracy depending on the particular minimum jerk trajectory. To get a representative measure of the quality of the methods, we took 100 samples from equation 41 and have plotted the ratio of J of the calculated trajectory to J of the minimum jerk trajectory in Figure 9 as a measure of the sub-optimality of all the methods. To compare just the rotational part of the J, we have set the metric parameters to be $\alpha = 1$ and $\beta = 0$. A distribution which is closer to being concentrated at 1 is more accurate than a distribution which is more spread out. This distribution for our method is closer to 1 than the other methods as summarized in Table II where we calculate the percent of the trials under the same cutoffs in Figure 7.

Ratio	This Work	B&K [1]	P&R [23]
2	71	61	4
3	89	86	39
4	98	94	58
5	98	99	86
6	100	100	91

TABLE II: Percent of SO(3) trials under thresholds for ratio of $J/J_{\min}$

Thus from a random trajectory on SO(3), our method is more likely to produce a trajectory which is closer to optimal.

C. Trajectories in SE(3)

In the case in which the environment is all of $\mathbb{R}^3$, we can compare our method to [1], but not [23] because the latter only gives a trajectory generation method for the rotation group. We used the same method as in the previous subsection to generate random trajectories on SE(3) with the same range of ω. Since we can calculate the translational part of Equation 17 in closed form, we sampled from initial and final derivatives of d as:

$$d, \dot{d}, \ddot{d}, d^{(3)} \in \mathbb{U}([-1,-1,-1],[1,1,1]) \tag{42}$$

From Figure 6, we found that our method preformed much better than the other method, but not as well as the closed form solution. We can also compare the translational part of the jerk function of these three trajectories in Figure 7 by setting $\alpha = 0$ and $\beta = 1$, which shows a great improvement over the projection method, as also summarized in Table III.

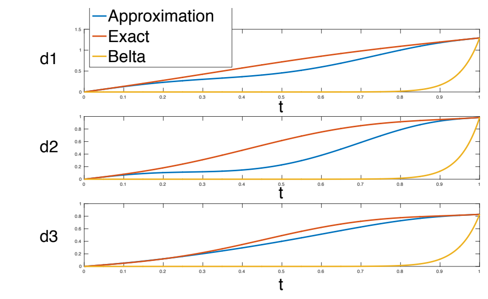

Fig. 6: Comparison of an example set of trajectories on SE(3)

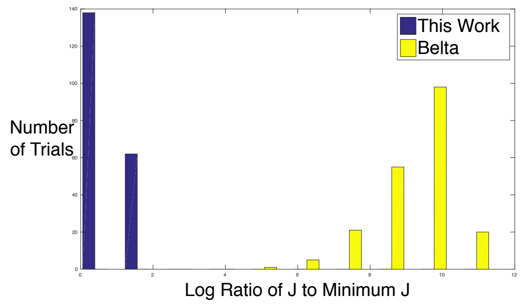

Fig. 7: Comparison of the J of 200 simulated trajectories. ($\alpha = 0, \beta = 1$)

Log Ratio	This Work	B&K [1]
1	70	0
2	100	0
7	100	5
9	100	35
10	100	79
11	100	99.5

TABLE III: Percent of SE(3) trials under thresholds for log ratio of $J/J_{\min}$

We generated trajectories in several environments with obstacles. In Figure 8, we show an example trajectory generated through the space. The robot starts with an angular velocity about its x-axis and finishes with an angular velocity about its y-axis. The trajectory is shown in cyan and successfully avoids the obstacles.

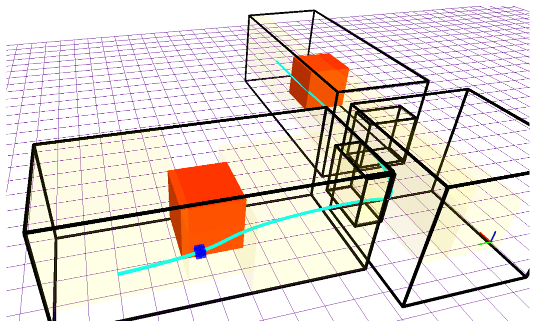

Fig. 8: SE(3) Trajectory With Obstacles: The red blocks are obstacles, and the black lines are the edges of bounding rectangular prisms. The yellow tinted boxes is the computer corridor which the trajectory is confined within. The robot is shown in dark blue, and the trajectory is plotted in cyan

Since it is difficult to interpret a full SE(3) trajectory plotted out, video provides a better visualization of these trajectories. The video for this environment and others are available at: `https://youtu.be/K3VGIHU_Wv0`.

VIII. Conclusion

We have presented a new approximation method for generating minimum jerk trajectories on both SO(3) and SE(3). We have applied this method to a space robot which is symmetrical about rotations, but has obstacles in its 3D environment which it has to avoid. Our method approximates the angular velocity of the trajectory with a polynomial basis to be able to solve for the a minimum jerk trajectory using positive semidefinite programming techniques. Once a minimum jerk trajectory is found for orientation, we can solve for the translational trajectory using a quadratic program. We use a computational geometric decomposition technique to decompose the non-convex environment into a set of convex polyhedrons. These convex polyhedrons are expanded while ensuring that they do not intersect with obstacles, and then are converted into inequality constraints for the QP.

We compare our using a numerical evaluation of the minimum jerk trajectory. We find that our method performs better than these two other methods on SO(3) and SE(3) with a sub-optimality factor of at most 4 in 98% of the trials in SO(3) and at most 2 in 100% of the SE(3) trials. We also showed an example of our trajectory generation method on SE(3) with obstacles and implemented a controller for the dynamics of our system using a PD controller on the manifold in a simulated environment.

IX. Acknowledgments

We graciously thank the support of ARO grant W911NF-13-1-0350, ONR grant N00014-07-1-0829, and ARL grant W911NF-08-2-0004. This work was supported by a NASA Space Technology Research Fellowship.

References

[1] C. Belta and V. Kumar, “An SVD-Based Project Method for Interpolation on SE(3),” *IEEE Trans. Robotics and Automation*, vol. 18, no. 3, pp. 334–345, Jun. 2002.

[2] A. Bemporad, K. Fukuda, and F. D. Torrisi, “Convexity Recognition of the Union of Polyhedra,” *Comput. Geom. Theory Appl.*, vol. 18, no. 3, pp. 141–154, Apr. 2001.

[3] D. Bertsimas and J. Tsitsiklis, *Introduction to Linear Optimization*, 1st ed. Athena Scientific, 1997.

[4] X. Beudaert, S. Lavernhe, and C. Tournier, “Direct trajectory interpolation on the surface using an open CNC,” *The International Journal of Advanced Manufacturing Technology*, vol. 75, no. 1-4, pp. 535–546, 2014.

[5] F. Bullo and R. M. Murray, “Proportional derivative (PD) control on the Euclidean group,” in *ecc*, Rome, Italy, Jun. 1995, pp. 1091–1097.

[6] F. Bullo and A. D. Lewis, *Geometric Control of Mechanical Systems*, ser. Texts in Applied Mathematics. New York-Heidelberg-Berlin: Springer Verlag, 2004, vol. 49.

[7] M. A. R. C. Ott and G. Hirzinger, *Posture and balance control for biped robots based on contact force optimization*. IEEE Intl Conf. on Humanoid Robots (Humanoids), Oct. 2011.

[8] B. Chazelle, “Convex partitions of polyhedra: a lower bound and worst-case optimal algorithm,” *SIAM Journal on Computing*, vol. 13, no. 3, pp. 488–507, 1984.

[9] G. S. Chirikjian, *Stochastic Models, Information Theory, and Lie Groups, Volume 2: Analytic Methods and Modern Applications*. Springer Science & Business Media, 2011, vol. 2.

[10] R. Deits and R. Tedrake, “Efficient Mixed-Integer Planning for UAVs in Cluttered Environments,” *ICRA*, 2015.

[11] A. Fejzi, “Development of Control and Autonomy Algorithms for Docking to Complex Tumbling Satellites,” Ph.D. dissertation, MIT Dept. of Aeronautics and Astronautics, 2008.

[12] M. E. Flores, “Real-time trajectory generation for constrained nonlinear dynamical systems using non-uniform rational B-spline basis functions,” Ph.D. dissertation, California Institute of Technology, 2007.

[13] I. Gurobi Optimization, *Gurobi Optimizer Reference Manual*, 2015. [Online]. Available: http://www.gurobi.com

[14] F. S. Hover, R. M. Eustice, A. Kim, B. Englot, H. Johannsson, M. Kaess, and J. J. Leonard, “Advanced perception, navigation and planning for autonomous in-water ship hull inspection,” *The International Journal of Robotics Research*, vol. 31, no. 12, pp. 1445–1464, Oct. 2012.

[15] Jing Chen, Kunyue Su, and Shaojie Shen, “Real-time safe trajectory generation forquadrotor flight in cluttered environments,” Zhuhai, China, Aug. 2015.

[16] Jos F. Sturm, “Using SeDuMi 1.02, a MATLAB toolbox for Optimization over Symmetric Cones,” *Optimization Methods and Software*, pp. 625–653, 1999.

[17] J. B. Lasserre, “A Sum of Squares Approximation of Nonnegative Polynomials,” *SIAM Review*, vol. 49, no. 4, pp. 651–669, 2007.

[18] X. Markenscoff, L. Ni, and C. H. Papadimitriou, “The geometry of grasping,” *The International Journal of Robotics Research*, vol. 9, no. 1, pp. 61–74, 1990.

[19] N. Martinson, “Obstacle avoidance guidance and control algorithm for spacecraft maneuvers,” in *Proc. AIAA GN&C Conf.*, 2009.

[20] R. M. Murray, Z. Li, and S. S. Sastry, *A mathematical introduction to robotic manipulation*. CRC press, 1994.

[21] Y. Nesterov, “Squared functional systems and optimization problems,” in *High performance optimization*. Springer US, 2000, pp. 405–440.

[22] M. O’Connor, B. Tweddle, J. Katz, A. Saenz-Otero, D. Miller, and K. Makowka, “Visual-Inertial Estimation and Control for Inspection of a Tumbling Spacecraft: Experimental Results from the International Space Station.” American Institute of Aeronautics and Astronautics, Aug. 2012.

[23] F. C. Park and B. Ravani, “Smooth invariant interpolation of rotations,” *ACM Transactions on Graphics (TOG)*, vol. 16, no. 3, pp. 277–295, 1997.

[24] P. A. Parrilo, “Structured semidefinite programs and semialgebraic geometry methods in robustness and optimization,” Ph.D. dissertation, Citeseer, 2000.

[25] M. Quigley, K. Conley, B. P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, “ROS: an open-source Robot Operating System,” in *ICRA Workshop on Open Source Software*, 2009.

[26] A. Richards, T. Schouwenaars, J. P. How, and E. Feron, “Spacecraft Trajectory Planning with Avoidance Constraints Using Mixed-Integer Linear Programming,” *J. Guid. Control Dynam.*, vol. 25, no. 4, pp. 755–764, Aug. 2002.

[27] Sikang Liu, Michael Watterson, Sarah Tang, and Vijay Kumar, “High Speed Navigation For Quadrotors With Limited Onboard Sensing,” 2016.

[28] The CGAL Project, *CGAL User and Reference Manual*, 4th ed. CGAL Editorial Board, 2015. [Online]. Available: http://doc.cgal.org/4.7/Manual/packages.html

[29] M. Watterson and V. Kumar, “Safe receding horizon control for aggressive MAV flight with limited range sensing,” in *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*. IEEE, 2015, pp. 3235–3240.

[30] M. Zefran, V. Kumar, and C. Croke, “On the generation of smooth three-dimensional rigid body motions,” *IEEE Trans. Robotics and Automation*, 1995.

X. Appendix: Convex Envelope Proof

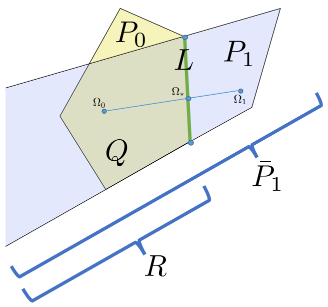

Fig. 9: Polyderal Relaxation

We need show that the union of the two convex pieces discussed in section V is convex in order to use the algorithm in [2]. Figure 9 shows the construction of these two polyhedrons in 2D. Since the proof of this is the same for arbitrary dimensions, we will assume all of these polyhedrons are in $\mathbb{R}^n$. P_0 and P_1 are both closed polyhedrons whose intersection is the $\mathbb{R}^{n-1}$ dimensional hyper face L. Let the polyhedron $Q = \bar{P}_1 \cap P_0$ and thus we will show that $P_1 \cup Q$ is convex.

We note that P_0 and P_1 are convex polyhedron. Q is the intersection of convex sets and is thus convex. Since $\bar{P}_1$ is defined in terms of a intersection of halfspaces, it is also convex. In addition $R = (\bar{P}_1 - P_1) \cup L$ is also convex because it can be written as the intersection of $\bar{P}_1$ with the halfspace opposite of the one removed for P_1. Since P_0 is disjoint from P_1 except for at the hyper face L, from counting the pieces, $Q \subset R$.

If we now assume that $P_1 \cup Q$ is not convex, there exists two points Ω_0 and Ω_1 in this set such that a least one point on the line segment $\overline{\Omega_0\Omega_1}$ is not in the set $P_1 \cup Q$. Since both P_0 and Q are convex, Ω_0 and Ω_1 cannot both lie in P_0 or both lie in Q. Thus we can flip the labeling of these two points so that Ω_0 is in P_0 and Ω_1 is in P_1. Since $\bar{P}_1$ is convex with $\Omega_0 \in R$ and $\Omega_1 \in P_1$, the line $\overline{\Omega_0\Omega_1}$ must intersect L. We call this point of intersection Ω_*. All of the points on $\overline{\Omega_0\Omega_*}$ are in Q because Q is convex, and all the points on $\overline{\Omega_*\Omega_1}$ lie in P_1 because P_1 is convex. Thus all the points on $\overline{\Omega_0\Omega_1}$ lie in $P_1 \cup Q$ and we have contradicted the assumption that $P_1 \cup Q$ is non-convex.