Robotics: Science and Systems 2018
Pittsburgh, PA, USA, June 26-30, 2018

Trajectory Optimization On Manifolds with Applications to $SO(3)$ and $\mathbb{R}^3 \times S^2$

Michael Watterson[1], Sikang Liu[1], Ke Sun[1], Trey Smith[2], and Vijay Kumar[1]

***Abstract*—Manifolds are used in almost all robotics applications even if they are not explicitly modeled. We propose a differential geometric approach for optimizing trajectories on a Riemannian manifold with obstacles. The optimization problem depends on a metric and collision function specific to a manifold. We then propose our Safe Corridor on Manifolds (SCM) method of computationally optimizing trajectories for robotics applications via a constrained optimization problem. Our method does not need equality constraints, which eliminates the need to project back to a feasible manifold during optimization. We then demonstrate how this algorithm works on an example problem on $SO(3)$ and a perception-aware planning example for visual-inertially guided robots navigating in 3 dimensions. Formulating field of view constraints naturally results in modeling with the manifold $\mathbb{R}^3 \times S^2$ which cannot be modeled as a Lie group.**

I. Introduction

Using differential geometry for trajectory optimization in robotics has been often used with manifolds such as the special orthogonal groups $SO(n)$ and special euclidean groups $SE(n)$. [27] used this theory to generate curves for a free-flying space robot. More general works like [1], generated trajectories for rigid bodies based on the metric proposed by [29] which ensured the associated affine connection obeys the differential structure of rigid body dynamics. [3] shows how to compute a basis of trajectories for robots using differential geometry by decoupling dimensions, but requires zero velocity in between each segment of a multi-dimensional motion.

Polynomials and rational splines are frequently used for optimization on Euclidean spaces $\mathbb{R}^n$ as part of the algorithms in [15], [32], [23], [14], [8], [18]. There are more specialized methods for modeling Bézier curves on manifolds using geometric constructions from [19] applied to rigid body motion. More recently, [20] proved properties of this construction and showed how to construct splines with this method. These are recursive definitions for Bézier curves, but the derivatives do not have the same convenient knot properties as the Euclidean case such as the end velocity being a linear function of the control points.

Trajectory optimization with obstacle avoidance is a well studied problem on $\mathbb{R}^3$. Using convex polyhedra to model free space has been done in [7] and [14] with mixed integer programming and quadratic programming respectively. [25] used a collision cost function with sequential convex programming. Others [10], [31] use spheres to model free space with convex programming to generate dynamically feasible trajectories. Some methods [23], [18], use collision data directly from an occupancy grid during the optimization process, but are not guaranteed to converge to a feasible solution in polynomial time.

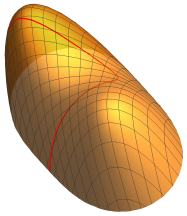

Fig. 1: Example 2-dimensional manifold embedded in $\mathbb{R}^3$ with a trajectory γ in red. The video for this paper can be found at https://youtu.be/gu8Tb7XjU0o

CHOMP [32] formulates a trajectory optimization problem on Hilbert spaces rather than Riemannian manifolds and uses variational calculus to compute gradients. They enforce collision free constraints by adding a term to their cost functional similar to barrier functions [2]. A philosophical difference between that works and ours is that we focus on manifolds which are directly parameterizable but require multiple parameterizations ex.($SE(n)$, S^n, RP^n) as opposed to parameterizations which have one chart in R^n, but have complicated constraints (ex. serial chain manipulators with task space collision avoidance). The construction of our algorithm ensures that the optimization does not have to enforce non-linear equality constraints unlike [32]. This is important because optimization iterations are represented with a minimum number of dimensions, thus stay implicitly on the manifold and do not have to be projected back. [11] formulates RRT* on a manifold using random sampling, but their construction of manifold charts requires iteratively projecting using Newton's method.

Perception aware planning [6] for vision based navigation has been explored with a variety of approaches. [4] summarizes a large variety of works from before 2010. Some relevant works from before and after then include [30] use an extension of a Probabilistic Road Map (PRM) which incorporates a information-theoretic model to improve observation of landmarks during path planning. [21] uses an observation model to generate collision free trajectories to maximize the conver-

This work was supported by a NASA Space Technology Research Fellowship, ARL grants W911NF-08-2-0004, W911NF-17-2-0181, ONR grants N00014-07-1-0829, N00014-14-1-0510, ARO grant W911NF-13-1-0350, NSF grants IIS-1426840, IIS-1138847, DARPA grants HR001151626, HR0011516850.

[1]The authors are with the GRASP Lab, University of Pennsylvania, 3330 Walnut Street, Philadelphia, PA 19104, USA. email: {wami, sikang, sunke, kumar}@seas.upenn.edu.

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

gence of a monocular visual-inertial estimator. Maintaining a level of features in view through a re-planning framework was done in [24].

The main contributions of this work are:

- A manifold trajectory optimization algorithm which optimizes without projecting to the manifold from a higher dimensional embedding, is parameterization invariant and is formulated without equality constraints.
- The addition of free-space constraints via convex sub-approximation on coordinate charts.
- An application of this method for navigation with constrained field of view visual estimation.
- An application of this method for trajectory generation on SO(3) with obstacle avoidance.

We formulate the free space constraints on a chart and not intrinsically for two reasons: first, a convex region on a chart may result in a convex optimization while describing a region which is non-convex on a manifold and second, it is easier to transform into a numerical optimization. For example, a region of S^2 larger than a hemisphere is never convex using the geodesic definition of convexity, but can be convex in stereographic coordinates.

This paper is organized with a background Section II followed by four main sections. We describe the manifold optimization and safe corridor algorithm in Section III. Then in Section IV, we describe how to use our method on the example of $\mathbb{R}^3 \times S^2$ for the application of a quadrotor with an active, gimbaled camera. In Section V, we discuss an example of how to plan trajectories on SO(3) using this method. Finally, in Section VI, we present results with experimental hardware using this trajectory generation algorithm.

II. Preliminaries

We will briefly review a few important definitions and properties of Riemannian manifolds that are critical to our algorithm. For a more detailed explanation of these concepts, please refer to [9].

Manifold: A manifold $\mathcal{M}$ is an n-dimensional subset of Euclidean space $\mathbb{R}^m$ with $n \leq m$ which is locally homeomorphic to $\mathbb{R}^n$.

Chart: We define a chart as a function $\varphi : \mathbb{R}^n \to \mathcal{M}$ which is a local n-dimensional parameterization of the manifold. Since the manifold is homeomorphic to $\mathbb{R}^n$, φ is locally invertible. If φ and φ^{-1} are both differentiable infinite times, we say the manifold is a smooth manifold. For the scope of this work, we will assume all manifolds to which we apply our formulation are smooth. Common examples of smooth manifolds are $\mathbb{R}^n$, $SO(n)$, $SE(n)$, S^n, and $\mathbb{RP}^n$. As a consequence of the homeomorphic property, we can find a chart $\varphi_p : \mathbb{R}^n \to \mathcal{M}$ centered at p such that a ball of radius ϵ around the origin is homeomorphic to an open subset of $\mathcal{M}$ containing p for every $\epsilon \in (0, \delta)$ for some $\delta > 0$. A chart with this property is drawn in Figure 2, this is important, because for a small enough curve $\gamma : [0, 1] \to \mathcal{M}$, we can always find a chart φ_γ and function $\eta : [0, 1] \to \mathbb{R}^n$ such that $\varphi_\gamma \circ \eta = \gamma$. It is important to note, that for a curve γ with arbitrary length, it is not always possible to find a parameterization which only spans one chart.

Vector on a Manifold: A vector on a manifold at a point p is a derivative quantity which is tangent to the manifold at p. Since our manifold is n-dimensional, the differential structure lives in a space homeomorphic to $\mathbb{R}^n$. For our purposes, it is fine to just use $V = \dot{\eta}$ as a vector, and not worry about coordinate free ways to define the vector.

Metric: A metric is a defined inner product $\langle ., . \rangle$ between vectors on a manifold. For example the magnitude of the velocity of a curve in coordinates with $V = \dot{\eta}$ is:

$$\langle V, V \rangle = \sum_{ij} g_{ij} V_i V_j \tag{1}$$

With g_{ij} being a function of the point on the manifold. For a single chart φ that is a function of the variables $x_1..x_n$, we can default to using the metric:

$$g_{ij} = \sum_l \frac{\partial \varphi_l}{\partial x_i} \frac{\partial \varphi_l}{\partial x_j} \tag{2}$$

If we want to have our metric make physical sense for a dynamical system, we can choose an appropriate g_{ij} which are compatible with the chosen manifold [29][3].

Riemannian Manifold: A manifold, along with a metric, is called a Riemannian manifold. For this paper, we assume that all manifolds are Riemannian and thus our formulation will require having a metric.

Covariant Derivative To calculate higher derivatives vectors, we use the covariant derivative ∇ which defines the derivative of vectors along each other. The acceleration of a curve on the manifold is defined as $\nabla_V V$. We can also chain the ∇ operator to get define jerk and higher derivatives.

$$\underbrace{\nabla_V \nabla_V \nabla_V ... \nabla_V V}_{\text{n } \nabla\text{'s'}} = \nabla_V^n V \tag{3}$$

In coordinates ξ, these are evaluated as:

$$\nabla_V V = \ddot{\xi}_k + \sum_{ij} \Gamma_{ij}^k \dot{\xi}_i \dot{\xi}_j \tag{4}$$

$$\nabla_V \nabla_V V = \dddot{\xi}_k + \sum_{ij} \Gamma_{ij}^k \dot{\xi}_i (\ddot{\xi}_j + \sum_{lm} \Gamma_{lm}^j \dot{\xi}_l \dot{\xi}_m) \tag{5}$$

Christoffel symbols: The Γ_{ij}^k in the above equation are called the Christoffel symbols. We can compute them from the metric with g^{ij} being the ith, jth element of the matrix G^{-1} if the matrix G elements are g_{kl}.

$$\Gamma_{ij}^k = \frac{1}{2} \sum_l g^{kl} \left(\frac{\partial g_{jl}}{\partial x_i} + \frac{\partial g_{il}}{\partial x_j} - \frac{\partial g_{ij}}{\partial x_l} \right) \tag{6}$$

Coordinate Invariant: A mathematical object in relation to a manifold is coordinate invariant if its value is the same regardless of the chart and or basis which is used to describe it. For some of the following definitions it is not obvious why the values are independent of the coordinate chart. See the more detailed reference for proofs of this critical property [9]. For example, the components in $\nabla_V V$ in Equation 4 depend on the coordinate chart φ as we implicitly choose a basis for the vector V. However, the value $\langle \nabla_V V, \nabla_V V \rangle$ is independent of the coordinate chart. We note that constraints of the form $\nabla_V V = \mathbf{0}$ are also coordinate free because there

is a linear transform L between any two sets of basis vectors and $L(\mathbf{0}) = \mathbf{0}$.

Geodesic: The shortest path between two points on a manifold can be found by solving the system of differential equations $\nabla_V V = \mathbf{0}$ with the boundary condition of the two points.

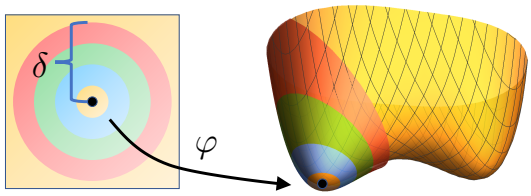

Fig. 2: A chart centered at p which is homeomorphic over a domain of a ball of radius δ.

III. Manifold Problem Formulation

With a differential geometry based formulation, we would like to have several properties: first, to be independent of coordinates; second, to be calculable; and third to be physically meaningful. While the third might seem difficult to do without a specific manifold, our formulation actually allows for it. We use a class of functionals from [29] which depend on a metric for the manifold. As in [1], these can be used to describe physical dynamics. For example, when the manifold is SE(3), the connection can be chosen to describe the rigid body translational and rotational accelerations.

Along with the background in the previous section, our formulation also assumes we have a function $C(p) : \mathcal{M} \to \{True, False\}$ which determines if a point p is in C^{free}. This collision checking does not appear in the works [3] or [1]. We can use this function, along with a coordinate chart to check the collision of a value in $\mathbb{R}^n$. Our formulation is:

$$\begin{aligned} \min_{\xi} \quad & \int_0^T \langle \nabla_V^n V, \nabla_V^n V \rangle dt \\ st. \quad & \xi(0) = \xi_N \\ & \xi(T) = \xi_F \\ & \xi \in C^{free} \\ & \dot{\xi} = V \\ & \nabla_V^k V(0) = N_k \qquad k = 1..p \\ & \nabla_V^k V(T) = F_k \qquad k = 1..p \\ & \langle \nabla_V^k V, \nabla_V^k V \rangle \leq L_k \quad k = 1..p \end{aligned} \tag{7}$$

Where L_k is a set of scalar **L**imits on the velocity, acceleration, up to the pth derivative. N_k and F_k are vector boundary conditions on the i**N**itial and **F**inal derivative constraints of the trajectory and ξ_N and ξ_F are the i**N**itial and **F**inal points on the manifold. We note that these conditions are coordinate invariant in the sense that if their value is given in one chart, they can be transformed into a different chart to give the same meaning on the manifold even if their value is different. In the case that they are homogeneous ($N_k = 0$ and $F_k = 0$) those constraints are the same in any coordinate chart as stated in the previous section. The $\xi \in C^{free}$ constraint will be detailed in the next section.

The cost functional is coordinate free and a generalization of the cost functionals used in [27] [1] to arbitrary derivatives. Alternatively it can be seen as a generalization to arbitrary manifolds of the cost functional used in [23] [14] [15] [7].

A. Algorithm Overview

The Safe Corridor on Manifolds (SCM) can be interpreted as generalization of the safe flight corridor in [14]. Instead of optimizing over a full constraint manifold, we optimize a trajectory over a piecewise convex region of the parameter space representation of a manifold. For a convex cost functional, this approximates a non-convex optimization problem with a convex one and thus can be solved quickly and efficiently with global optimality constraints during the optimization phase. On the full manifold, we note that this method is locally optimal with respect to a reference path.

The basic idea is as follows: 1 generate a collision free "optimal" path from a graph search on the manifold, then 2: form a locally convex corridor in a local parametrization of the manifold around this path, and 3: optimize a trajectory using the set of local parameters around which the convex corridor is described. This basic picture is shown in Figure 3 with a manifold and three regions of the corridor with their associated coordinate charts.

B. Path Finding on Manifolds

Collision free path finding is a much easier problem than finding collision free dynamically feasible trajectories (position + higher order constraints) especially since the search space of a dynamically feasible trajectory of an n dimensional trajectory with d dimensions of dynamics information requires search on $n \cdot d$ dimensions of space. Since graph search algorithms are $O(v \log v + e)$ runtime for v vertices and e edges, the runtime of the search will be at least $O(n^d \log n^d)$ for a space sampled with the same distance resolution. For even a small number of dimensions (ex. 5), the bottleneck of the trajectory generation algorithm is in the graph search. Thus, for tractability, it is better to search over as low dimensional space as possible. In state of the art search based methods over very high dimensions [26] using RRT methods [12][13], the search is biased towards a much lower dimensional manifold to achieve reasonable runtime performance. The worst case runtime of those algorithms was not considered in their work because the algorithms' probabilistic nature cannot guarantee convergence in finite time.

We can construct a graph on a manifold with a series of points on the manifold connected with edges which are geodesics. With such a graph, we assume the edges are non-overlapping, thus we can construct this graph on an arbitrary manifold with arbitrary points by connecting all pairs of points with geodesic edges. Whenever two geodesics intersect, we remove the longer of the two. We want the points to uniformly cover the manifold. Searching over this graph can search over the complete manifold up to a small discretization error. This is the same notion as searching over a discrete grid on $\mathbb{R}^n$ instead of the full continuous space. This graph construction can also be done in a more structured way, as we will do with

S^2, because there are easy constructions for geodesic grids. There are also versions of structured grid constructions on $SO(3)$ used in [28]. Performing graph search on this manifold grid is done using the geodesic length as the edge length and $A^\star$ heuristic. No modification is needed for $A^\star$ because graph search only needs nodes and edges and does not care how it is embedded in a manifold. In addition, the graph construction and search can be done in parallel to avoid building up large mostly-unexplored graphs in higher dimensions.

C. Safe Corridor on Manifolds

For any path $H = (h_1, \ldots, h_m)$ of points on the manifold which are sufficiently close to each other, we can find a corresponding sequence of charts with one chart φ^i centered at each h_i. RILS [14] computes a convex polyhedron around a line segment in 3D. We can extend this by finding a collision-free polyhedron on each domain of each chart, then adding them to a list of polyhedra $\mathcal{P}$. This is done by creating a voxel grid on the parameter space, then checking collision of all the voxels through the composition of $C \circ \varphi_i$. With an $\mathbb{R}^n$ occupancy grid, we perform an inflation around the line $l(0, \varphi_i^{-1}\varphi_{i+1}(0))$ using n-dimensional RILS. The n-dimensional version is the same as the 3D version, except the ellipsoids are expanded about extra minor axes during the inflation step. Finally, the set of charts is pruned if the geodesic between the center of adjacent charts is collision free and less than a length of δ.

Algorithm 1 Calculating the safe corridor on a manifold. Will return a set of charts Ψ and a set of polyhedra $\mathcal{P}$.

$(\Phi, \mathcal{P}) \leftarrow \{\}$
$H \leftarrow$ Manifold $A^\star(p_N, p_F)$
for $h_i \in H$ **do**
  Find φ^i, such that $\varphi^i(0) = h_i$
  $\Phi \leftarrow \{\Phi, \varphi^i\}$
  $\mathcal{P} \leftarrow \{\mathcal{P}, RILS((\varphi^i)^{-1} \circ \text{Geodesic}(\varphi^i(0), \varphi^{i+1}(0)))\}$
end for
for $(\varphi^i, \mathcal{P}_i) \in (\Phi, \mathcal{P})$ **do**
  if $\varphi^{i-1}(Line(0, (\varphi^{i-1})^{-1} \circ \varphi^{i+1}(0))$ is collision free and $d(\varphi^{i-1}(0)), \varphi^{i+1}(0)) < \delta$ **then**
    Prune $(\varphi^i, \mathcal{P}_i)$
  end if
end for
return $(\Phi, \mathcal{P})$

D. Optimization on manifold

All the quantities in Equation 7 are continuous values with continuous constraints. To numerically compute the optimization, we need to convert an infinite dimensional functional optimization to a finite dimensional one. In the literature, this is done by either using splines [15][27][14] [23] [7][8] or using a discrete time optimization [2]. It is well known that splines can approximate arbitrary continuous functions within a given $\epsilon > 0$ [17]. Inequality constraints are more complex than the discrete case, but require far fewer optimization variables.

Computing the integral in Equation 7 cannot be done in closed form for a general manifold. We can compute an approximation of the integral using Gaussian quadrature to approximate it as a sum of the functional evaluated at some t_i with weights w_i which are found in [22]. For some Riemannian manifolds with enough w_i, this will actually be exact:

$$\int_0^T f(t)dt \approx \sum_i w_i \cdot f(t_i) \tag{8}$$

We choose to represent the trajectory as a polynomial spline in the parameter space of the charts φ^i we found in Algorithm 1. Each chart has an associated polynomial $p_i(s) : (0, 1) \rightarrow \mathbb{R}^n$, which corresponds to the ith segment of the trajectory with normalized time. This is represented as a sum of basis polynomials $b_j(s)$ and coefficients $\alpha_{ij} \in \mathbb{R}^n$. To evaluate the trajectory, there is also an associated time interval for each polynomial $\Delta_i \in \mathbb{R}^+$.

$$\begin{gathered} p_i(s) = \textstyle\sum_j \alpha_{ij} b_j(s) \\ \xi_i(t) = p_i(\frac{t - t_i}{\Delta_i}) \\ (t_{i+1} = t_i + \Delta_i, t_0 = 0) \end{gathered} \tag{9}$$

The reason we keep the polynomials evaluated in the interval $[0, 1]$ as opposed to $[0, \Delta_i]$ is done for numerical computing reasons. For example, [23] found numerical issues in [15] with polynomials of degree greater than 9 which we partly attribute to this difference in modeling.

With each polynomial being on a different chart, we need to explicitly constrain the knot points to be continuous up to a certain derivative. Using the map $(\varphi^i)^{-1} \circ \varphi^{i+1}$, we can express equality in terms of the coefficients α_{ij}. The picture of these constraints are shown in Figure 3. If the polynomial degree is high enough not to be over-constrained by the required continuity, we can ensure that the constraints on the ith knot point share no variables with the $i + 1th$ knot point when using an endpoint constrained basis. Because of this, we can replace variables in expressions for the knot points using the map $(\varphi^i)^{-1} \circ \varphi^{i+1}$. Doing so, will eliminate all equality constraints on the optimization. In [23], they do this for the quadratic program case and provided a more detailed explanation of the endpoint constrained basis.

To confine continuous trajectories inside each polyhedron, we use the property that the convex hull of the control points of a Bézier curve is a conservative bound on the convex hull of the curve [8], this bound can be enforced with linear inequality constraints [21] [27]. The convex hull of the spline is a subset of the convex hull of the control points. Thus by confining the control points, we can guarantee that the trajectory will be inside the polyhedra. Since the Bézier control points are a linear transformation of any polynomial basis coefficients, this confinement can be done regardless of the choice of basis polynomials.

The full optimization is the nonlinear program, which is in general non-convex:

$$\begin{aligned} \min_{\alpha, \Delta} \quad & f(\alpha, \Delta) \\ s.t \quad & g(\alpha, \Delta) \leq 0 \end{aligned} \tag{10}$$

Where α is a subset of the basis coefficients describing the trajectory, Δ is a set of time durations corresponding to the duration of each segment of the spline. The number of free α parameters equals the number of coefficients in the spline minus the number of equality constraints in (7) and number of knot points for each dimension in Figure 3. After optimizing, we can solve for the remaining α parameters using the non-linear chart transition functions.

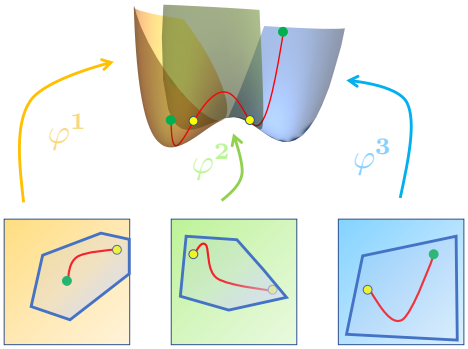

Fig. 3: Constraining equality of spline knot points with convex polyhedral constraint regions on each chart. Yellow points are equated between adjacent charts. Green points represent ξ_N and ξ_F.

E. Optimization method

We can solve the optimization problem in Equation 10, with constrained gradient descent. We implemented the primal-dual Newton's method with the centering heuristic as detailed in [2]. Changing the variable initialization did not seem to affect convergence, so all values were set to 1. The full optimization in [14] is usually not convex. In practice, it is straightforward to find approximately optimal Δ values using a heuristic proposed in [14]. With fixed Δ, optimizing only over the α values is convex for suitable manifolds. Alternatively, the two sets of variables can be alternately optimized as in [23].

We can model polynomials of both α and Δ as multi-dimensional polynomials, with integer powers.

$$f(x) = \zeta \sum_{i \in I} \left[\beta_i \prod_{j \in J_i} x_{ij}^{q_{ij}} \right] \tag{11}$$

Where I and J_i are index sets, $\zeta, \beta_i \in \mathbb{R}$ are coefficients and $q_{ij} \in \mathbb{Z}$ are the integer powers.

The cost functional and inequality constraints are in general nonlinear functions of these polynomials. In our vision based example, the cost will be a rational function of these polynomials. We assume we can calculate the gradients and Hessians of functions of polynomials using the chain rule.

The norm squared in our cost functional results in the squaring of nonlinear functions with many terms. If our cost is a polynomial or rational function, expanding out polynomials in this form leads to inefficient calculations and numerical robustness issues since the maximum power q_{ij} doubles when the polynomial is squared and the number of terms we need to store is now $O(|I|^2)$. Instead, we can just store the polynomial or rational function f and square the function after the evaluation. This results in gradients and Hessians calculated as:

$$\frac{\partial f^2}{\partial x_i} = 2 \frac{\partial f}{\partial x_i} f \tag{12}$$

$$\frac{\partial f^2}{\partial x_i \partial x_j} = 2 \frac{\partial^2 f}{\partial x_i \partial x_j} f + 2 \frac{\partial f}{\partial x_i} \frac{\partial f}{\partial x_j} \tag{13}$$

We found that in practice, this small step dramatically decreased the computation time of our algorithm by orders of magnitude in both overhead computing products of polynomials and evaluation of Hessians.

IV. Field of View Constrained Trajectory Optimization Details

When planning for vision based navigation, we look at a system in which we can control the view direction of the camera on S^2. This is analogous to controlling the pan-tilt of a hardware gimbal, or in our experimental setup, emulate this in software by cropping the appropriate region of a wide angle view based on the orientation of the robot. We assume for sufficiently slow trajectories, the robot orientation is close enough to hover so the desired cropped image is within the wide angle lens. In practice, we can rescale the time of trajectory to fit this even with the aggressive motions shown the video. We choose this, and not $SO(3)$, because feature tracking is approximately symmetric with respect to rotations about the camera z axis, and modeling the viewing region of a camera as a cone has some convenient geometric properties which we can exploit with stereographic coordinates.

A. Vision Model

The standard projection equation in normalized image coordinates (p_x, p_y) for a cameras whose orientation is R and position is $[x, y, z]^T$ measured in an inertial frame is:

$$\frac{1}{\lambda} \begin{bmatrix} p_x \\ p_y \\ 1 \end{bmatrix} = R \left(\begin{bmatrix} X_i \\ Y_i \\ Z_i \end{bmatrix} - \begin{bmatrix} x \\ y \\ z \end{bmatrix} \right) \tag{14}$$

We use this equation to model whether or not a known feature in world coordinates is in the field of view of the camera. We also assume that a feature is only useful for resolving the position of the robot if it is in a known range of depths.

The optical flow is a function of a point in pixel coordinates depends on the angular velocity of the camera ω:

$$\dot{p} = \begin{bmatrix} \dot{p}_x \\ \dot{p}_y \end{bmatrix} = \begin{bmatrix} \frac{p_x \cdot \dot{z} - \dot{x}}{\lambda} \\ \frac{p_y \cdot \dot{z} - \dot{y}}{\lambda} \end{bmatrix} + \begin{bmatrix} p_x p_y & -(1+p_x)^2 & p_y \\ (1+p_y)^2 & -p_x p_y & -p_x \end{bmatrix} \omega \tag{15}$$

The front-end of visual-odometry has decreased accuracy and increased computational load with increasing optical flow. With Equation 15 and assuming known ranges on the depth of features and pixel coordinates, we can conservatively upper bound the optical flow with a linear function of the camera's linear and angular velocities. Thus we choose $\nabla_V V$ as the derivative we minimize for the S^2 part of the cost functional.

B. Field of View Formulation

For vision-based navigation, we would like a cost function which produces a trajectory that is good for the estimator. Paramount to a VINS estimator, is maintaining track of all optical features. This tracking can be lost two ways: 1 by lack of features, and 2 by an excessive rate of optical flow. We ensure 1 by keeping a minimum number of visible features in the direction of camera z, with a region constraint on S^2. The tracking of features is roughly symmetric about rotations about the camera z axis, so we consider a feature to be in the field of view if it is inside a conical region which is a circular region on S^2. To limit 2, we minimize $\nabla_V V$ or V which will minimize the angular and linear acceleration or velocity respectively as seen in Equation 15.

C. Stereographic Coordinates For S^n

Let $P \in S^n \subset \mathbb{R}^{n+1}$ be a point on the sphere. For an arbitrary matrix $R \in SO(n+1)$, we define a set of projective coordinates with Re_1 being the singular point on S^n. Let $p \in \mathbb{R}^n$ be the stereographic coordinates. If e_i is a vector which is all 0, except the ith element, which is 1, we can define a chart φ with the functions:

$$\xi = p_1 Re_2 + p_2 Re_3 + ...p_n Re_{n+1} - Re_1 \tag{16}$$

$$P = \frac{2}{||p||^2 + 1}\xi + Re_1 \tag{17}$$

The inverse φ can be shown to be:

$$P' = \frac{P}{1 - P \cdot Re_1} \tag{18}$$

$$p = (P' \cdot Re_2, P' \cdot Re_3, ..., P' \cdot Re_{n+1}) \tag{19}$$

The maps $(\varphi^2)^{-1} \circ \varphi^1 : \mathbb{R}^n \to \mathbb{R}^n$ can be computed with

$$\hat{\xi} = \frac{2}{||p||^2 + 1}(p_1 e_2 + p_2 e_3 + ... + -e_1) + e_1 \tag{20}$$

$$(\varphi^2)^{-1} \circ \varphi^1)_k(p) = \frac{e_{k+1}^T R_2^T R_1 \hat{\xi}}{1 - e_1^T R_2^T R_1 \hat{\xi}} \tag{21}$$

1) Christoffel Symbols on S^n: To be able to calculate the Christoffel symbols, we will use the standard $\mathbb{R}^{n+1}$ metric tensor which corresponds to a L_2 norm. This tensor is shown in Equation 2, and results in:

$$g_{ij} = \begin{cases} \frac{4}{(1+||p||^2)^2} & i = j \\ 0 & \text{otherwise} \end{cases} \tag{22}$$

The metric is independent of the $SO(n)$ rotation in the stereographic coordinates, which can be shown by seeing that Equation 17 is linear in R and then cancels out during the multiplication in Equation 2.

We can use Equation 6 to calculate symbols for the metric induced by $\mathbb{R}^{n+1}$ on the sphere. With stereographic coordinates, these are:

$$\Gamma_{ij}^k = \frac{2}{(1 + ||p||^2)} \cdot \begin{cases} x_k & i \neq k \text{ and } j \neq k \text{ and } i = j \\ -x_j & i = k \\ -x_i & j = k \\ 0 & \text{otherwise} \end{cases} \tag{23}$$

We note that these symbols are independent of the value of R. The proof of this is omitted for space, but the rotational symmetry of the sphere shows that this should be the case. And we can see that these symbols are torsion free ($\Gamma_{ij}^k = \Gamma_{ji}^k$).

D. Search on S^2

To generate a grid on the sphere, we use the well known geodesic polyhedron construction. We start with an icosahedron and sub-divide each triangular face into n^2 triangles and then project the vertices to the unit sphere. If we then take the dual of this shape by interchanging vertices with edges, we end up with a graph which almost uniformly covers the sphere and has the property that each node has 3 neighbors everywhere on the sphere.

With a graph on the sphere, we can use $A^\star$ with the geodesic length representing the arc length between adjacent nodes. To check whether or not a given node has enough features, we need to calculate how many features are within view from that point. To do this, we partition the features which are between a minimum and maximum distance from the point in $\mathbb{R}^3$ and then use the property of stereographic coordinates in subsection IV-E.

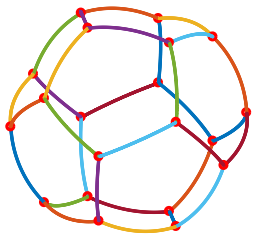

Fig. 4: Example graph of nodes (red circles) and associated edges (geodesic segments) on S^2 based off the icosahedron. Note how each node has exactly 3 edges.

E. Circles to Circles

It is known that a circle on S^2 is a circle in stereographic coordinates (Figure 5), with some center and a different radius. Thus when we model our visual constraint with a cone, we can transform all of the point into stereographic coordinates and then check to see how many of them lie within a given circle. Since this calculation needs to happen many times for the corridor creation, calculating this in the stereographic coordinates is more efficient then counting features in $\mathbb{R}^3$

We first need to find the transform between center of the circle in the parameter space and center of the circle on the sphere, expressed in the parameter space. These two quantities are actually different except at the identity element.

Exploiting the symmetry when expressing points on a sphere, we look at the point p expressed in polar coordinates $p = (r\cos(\theta), r\sin(\theta))$, the radius of the circle in coordinates is ρ with the field of view being ϕ:

$$\rho = \frac{(r^2 + 1)\sin\frac{\phi}{2}}{\cos\frac{\phi}{2} + 1 + r^2(\cos\frac{\phi}{2} - 1)} \tag{24}$$

We can also specify the center of this circle in the parameter space in polar coordinates as $p' = (r'\cos(\theta), r'\sin(\theta))$

$$r' = \frac{2r}{\cos\frac{\phi}{2} + 1 + r^2(\cos\frac{\phi}{2} - 1)} \tag{25}$$

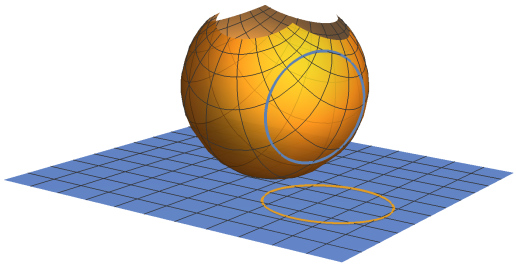

Fig. 5: A circle on a sphere S^2 will be a circle in the parameter space $\mathbb{R}^2$ when using stereographic coordinates. We use this to efficiently determine whether or not a viewpoint has enough features or not during the corridor construction.

V. Trajectory Generation on $SO(3)$

To generate trajectories on SO(3), we first need a graph. With the work of [28], we can use a uniform grid on this manifold using the Hopf fibration. This allows us to use the existing graph we described on S^2 and cross it with uniform samples on S^1. As long as the following equation holds for n_{S^1} samples on S^1 and n_{S^2} samples on S^2 , the cross product samples are close to uniform on SO(3).

$$\frac{\pi}{n_{S^1}} = \sqrt{\frac{\pi}{n_{S^2}}} \tag{26}$$

To generate charts after a path is found on $SO(3)$, we use the exponential map because it can be used for all Lie groups [9]. From a point on the path R_c, we can locally parameterize $SO(3)$ with the vector $\xi \in \mathbb{R}^3$.

$$R \in SO(3) = R_c \exp\left(\begin{bmatrix} 0 & -\xi_3 & \xi_2 \\ \xi_3 & 0 & -\xi_1 \\ -\xi_2 & \xi_1 & 0 \end{bmatrix}\right) = R_c \exp(\hat{\xi}) \tag{27}$$

The chart transition function can be found by equating two charts and rearranging:

$$\xi_1 = \log(R_1^{-1} R_2 \exp(\hat{\xi}_2))^\vee \tag{28}$$

Where $\vee$ is the inverse of the $\hat{\cdot}$ operator [5]. The derivative of this transition function is omitted for space, but guarantees continuity of angular velocity during the chart transition. This is equivalent to saying the angular velocity in the world frame is continuous. Since it has physical meaning, we can specify that the cost functional is the angular velocity. This functional has been formulated in the form of Equation 7 by [29].

In Figure 6, we show the results of this algorithm on a simple example of reorienting an object. Plotting obstacles in orientation space would clutter the view, but we can see the effect of how the trajectory needs to change to get around a blockage placed in the middle of what would otherwise be the optimal path. In the obstacle free version, our method performs slower than the methods [19] or [1], because they provide a closed form solution, but results in the same trajectory. These methods however, cannot be used in the case with obstacles.

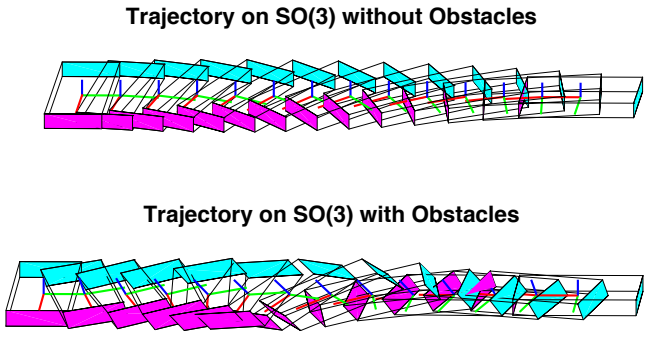

Fig. 6: Trajectories generated on $SO(3)$ using this method. Orientations are visualized as a box with a red-blue-green coordinate axis and time is visualized by translating the box to the right as it rotates. The start is from identity and the trajectory goes to a rotation which is rotated about the blue axis by 90 degrees. In the obstacle free case, the optimal trajectory rotates about this axis, which will produce the same result as [19] or [1]. In the bottom example, we place an obstacle in the middle of the obstacle free path, so the the optimized trajectory rotates out of plane around it.

VI. Experimental results

For the hardware experiments, we have chosen an Ascending Technologies Pelican quadrotor with a payload of an Intel Nuc i7, stereo cameras, IMU, and an Asus Xtion Pro depth sensor. A Vicon motion capture system provides ground truth and control feedback so we can safely test trajectories in which the naive planner may cause poor estimator performance and otherwise damage the robot. The control of the robot is based off the one proposed in [15].

We have implemented the version of our algorithm on $\mathbb{R}^3 \times S^2$ in C++, with ROS handling all of the interprocess communication and python running some of the high level control commands. Dynamic feasibility of the trajectory is done using the time scaling method in [23]. The translation of the trajectory in 3 dimensions, which the yaw being the projection of the S^2 part of the trajectory are converted into real time control commands by [15]. The depth sensor was used to build up a collision occupancy grid at the same time as the vision cameras build up a map of visual features. During the map creation phase, stereo visual inertial odometery was used to provide more accurate ground truth locations of features in the environment. The region of interest planned with the S^2 part of the state, is converted into a cropped region of a wide angle lens camera image. This is to simulate a hardware controlled gimbal in software, similar to the system found in commercial platforms.

The trajectory generation done using our algorithm was performed off-line on an Intel i7 3.4 GHz processor with a single threaded implementation. The computation time is bottlenecked by the graph search and corridor generation, but

in total was less than 30s for multiple sets of start and end locations. Searching over a 5 dimensional space with $A^\star$ takes a while because of the large number of nodes in the graph and that computing the collision on the spherical part of the state requires checking all the features in the vision map at each node expansion. We believe there are ways to improve the computational performance of this step, but they are outside the scope of this work since we are not targeting on-line applications.

We used a monocular version of the Multi-State Constraint Kalman Filter (MSCKF) [16] as a benchmark for the visual-inertial odometry. We setup an environment with feature-poor obstacles as shown in the video. We compare this trajectory generation method to the one of [14], which does not take into account that the obstacles have no features on them. This is in contrast to our method, which we set to require at least 10 features in the field of view at all times. The performance over several environments is shown in Figure 7, where the distribution of our errors is to the left of that using the naive method.

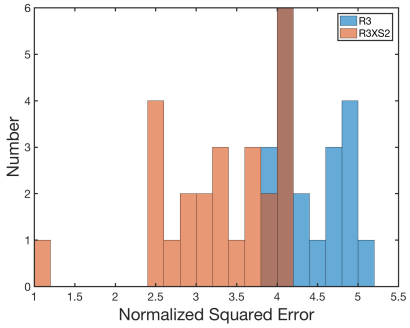

Fig. 7: Estimator performance summarized from 45 trials

When planning in the environment in the video, our algorithm found a corridor consisting of 5 polyhedrons, which are each 5 dimensional, these are hard to visualize, but we have split them into their projections on the $\mathbb{R}^3$ and S^2 part of the state separately in Figure 8.

VII. Limitations of Approach

As with any approach, there are situations where our approach is not ideal. For systems where the state, cost and constraints all live on $\mathbb{R}^n$, this method simplifies, but the added complexity of the differential geometric formulation would likely hinder implementation. Using differential geometry to formulate an optimization problem can be a potential pitfall as it adds complexity if applied to the wrong problem. Also, while all parameterizations of a manifold can work with this approach, the obstacle avoidance is dependent on the parameterization. Therefore, a different choice of parameterization can lead to poorer results.

Restriction to one homotopy class can produce sub optimal results in environments where there are narrow paths which place restrictions on dynamics. For example, a robot will choose to take a narrow shortcut over a safer wider path unless the search heuristic takes path clearance as part of its

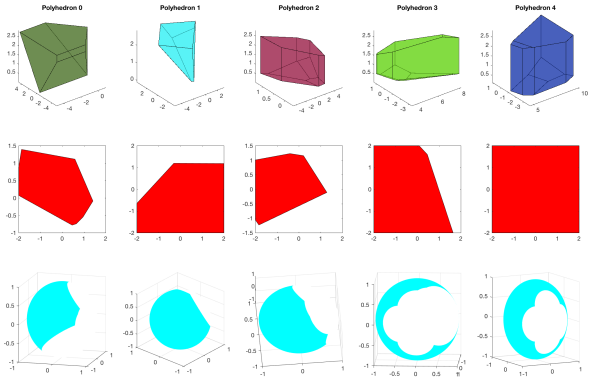

Fig. 8: Corridor sections for $\mathbb{R}^3 \times S^2$ trajectory consisting of 5 segments. The top row is the $\mathbb{R}^3$ representation of the polyhedra plotted with a 3D perspective. The bottom two rows are the S^2 parts of the trajectory. The middle row is the polyhedron in parameter space and the bottom row is that polyhedron projected onto a sphere and drawn with a 3D perspective.

cost metric. In general, this method is beneficial when there is sufficient room around the kinematically-feasible shortest path for a dynamically feasible trajectory to be generated. In situations where the kinematic planning problem is hard, the trajectory generation will have similar performance to following a kinematic path at slow speed.

When scaling to higher dimensional manifolds, the performance of this methods depends on the sparsity of the formulated optimization which is proportional to the number of faces of the polyhedra. For higher dimensions, polyhedra require more faces to approximate other bounds such as quadratic surfaces and further study is needed to understand which is a better representation. In addition, this method does not solve issues with difficulty in finding a feasible path in high dimensions which appear in applications such as navigating redundant serial chain manipulators through narrow gaps.

VIII. Conclusion

We have proposed a trajectory optimization method on Riemannian manifolds that is computed without equality constraints and that ensures the optimization never needs to project back to the manifold. Using a collision map on the manifold, we show how we can add collision constraints with convex sub-approximation on a series of coordinate charts found by our algorithm. We have then applied this method to SO(3) and an example problem in robotics with planning a robots position on $\mathbb{R}^3$ and camera view on S^2 simultaneously. We show that this method improves estimator performance of a monocular visual-inertial system by generating smooth trajectories in position and view direction.

References

[1] Calin Belta and Vijay Kumar. Euclidean metrics for motion generation on SE (3). *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, 2002.

[2] Alberto Bemporad and Manfred Morari. Predictive Control of Constrained Hybrid Systems. In Frank Allgwer and Alex Zheng, editors, *Nonlinear Model Predictive Control*. Birkhuser Basel, 2000.

[3] Francesco Bullo and Kevin M. Lynch. Kinematic controllability and decoupled trajectory planning for underactuated mechanical systems. May 2001.

[4] Shengyong Chen, Youfu Li, and Ngai Ming Kwok. Active vision in robotic systems: A survey of recent developments. *ISRR 2011*, September 2011.

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

[6] Gabriele Costante, Christian Forster, Jeffrey Delmerico, Paolo Valigi, and Davide Scaramuzza. Perception-aware path planning. *arXiv*, 2016. URL https://arxiv.org/abs/1605.04151.

[7] Robin Deits and Russ Tedrake. Efficient Mixed-Integer Planning for UAVs in Cluttered Environments. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 42–49, 2015.

[8] Melvin E. Flores. *Real-time trajectory generation for constrained nonlinear dynamical systems using non-uniform rational B-spline basis functions*. PhD thesis, 2007.

[9] Jean Gallier and Jocelyn Quaintance. *Notes on differential geometry and Lie groups*. Springer, 2017. URL http://www.seas.upenn.edu/~jean/diffgeom.pdf.

[10] Fei Gao and Shaojie Shen. Online quadrotor trajectory generation and autonomous navigation on point clouds. In *SSRR*. IEEE, 2016.

[11] Lonard Jaillet and Josep M. Porta. Asymptotically-optimal path planning on manifolds. *Robotics: Science and Systems VIII*, page 145, 2013.

[12] Sertac Karaman, Matthew R. Walter, Alejandro Perez, Emilio Frazzoli, and Seth Teller. Anytime motion planning using the RRT*. In *ICRA*. IEEE, 2011.

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

[14] Sikang Liu, Michael Watterson, Kartik Mohta, Ke Sun, Subhrajit Bhattacharya, Camillo J. Taylor, and Vijay Kumar. Planning Dynamically Feasible Trajectories for Quadrotors Using Safe Flight Corridors in 3-D Complex Environments. *IEEE Robotics and Automation Letters*, 2 (3):1688–1695, 2017.

[15] D. Mellinger and V. Kumar. Minimum Snap Trajectory Generation and Control for Quadrotors. In *ICRA*, May 2011.

[16] A.I. Mourikis and S.I. Roumeliotis. A Multi-State Constraint Kalman Filter for Vision-aided Inertial Navigation. In *ICRA 2007*, April 2007.

[17] Jorge Nocedal and Stephen J. Wright. *Numerical optimization*. Springer series in operations research. Springer, 2006.

[18] Helen Oleynikova, Michael Burri, Zachary Taylor, Juan Nieto, Roland Siegwart, and Enric Galceran. Continuous-time trajectory optimization for online UAV replanning. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5332–5339. IEEE, 2016.

[19] F. C. Park and B. Ravani. Bezier Curves on Riemannian Manifolds and Lie Groups witli Kinematics Applications. 1995.

[20] Tomasz Popiel and Lyle Noakes. Bzier curves and C2 interpolation in Riemannian manifolds. *Journal of Approximation Theory*, October 2007.

[21] James A. Preiss, Karol Hausman, Gaurav S. Sukhatme, and Stephan Weiss. Trajectory Optimization for Self-Calibration and Navigation. In *Robotics: Science and Systems (RSS)*, 2017.

[22] William H. Press, editor. *Numerical recipes in C: the art of scientific computing*. Cambridge University Press, 1992.

[23] Charles Richter, Adam Bry, and Nicholas Roy. Polynomial trajectory planning for aggressive quadrotor flight in dense indoor environments. In *The International Symposium on Robotics Research (ISRR)*, 2013.

[24] Seyed Abbas Sadat, Kyle Chutskoff, Damir Jungic, Jens Wawerla, and Richard Vaughan. Feature-rich path planning for robust navigation of MAVs with mono-SLAM. In *ICRA*. IEEE, 2014.

[25] John Schulman, Yan Duan, Jonathan Ho, Alex Lee, Ibrahim Awwal, Henry Bradlow, Jia Pan, Sachin Patil, Ken Goldberg, and Pieter Abbeel. Motion planning with sequential convex optimization and convex collision checking. *IJRR*, 2014.

[26] Alexander Shkolnik and Russ Tedrake. Path planning in 1000+ dimensions using a task-space Voronoi bias. In *2009 IEEE International Conference on Robotics and Automation*, pages 2061–2067. IEEE, 2009.

[27] Michael Watterson, Trey Smith, and Vijay Kumar. Smooth trajectory generation on SE (3) for a free flying space robot. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5459–5466. IEEE, 2016.

[28] Anna Yershova, Swati Jain, Steven M. Lavalle, and Julie C. Mitchell. Generating uniform incremental grids on SO (3) using the Hopf fibration. *IJRR*, 2010.

[29] M. Zefran, C. Croke, and V. Kumar. Choice of Riemannian Metrics for Rigid Body Kinematics. In *ASME Design Technical Conference*, August 1996.

[30] G. Zhang, S. Ferrari, and M. Qian. An Information Roadmap Method for Robotic Sensor Path Planning. *Journal of Intelligent and Robotic Systems*, September 2009.

[31] Zhijie Zhu, Edward Schmerling, and Marco Pavone. A convex optimization approach to smooth trajectories for motion planning with car-like robots. In *CDC*. IEEE, 2015.

[32] Matthew Zucker, Nathan Ratliff, Anca Dragan, Mihail Pivtoraiko, Matthew Klingensmith, Christopher Dellin, J. Andrew (Drew) Bagnell, and Siddhartha Srinivasa. CHOMP: Covariant Hamiltonian Optimization for Motion Planning. *International Journal of Robotics Research*, May 2013.