Planning with Attitude

Brian Jackson[1] and Zachary Manchester[1]

I. INTRODUCTION

Many robotic systems—including quadrotors, airplanes, satellites, underwater vehicles, and quadrupeds—can perform arbitrarily large three-dimensional translations and rotations as part of their normal operation. While simply representing the attitude of these sytems is nontrivial, generating and tracking motion plans for floating-base systems is an even more challenging problem.

Many approaches have been taken to address the problem of motion planning and control on the space of rigid body motions [**Kobilarov2011**, **Saccon2013**, **watterson2020control**]; most of these rely heavily on ideas and notation from differential geometry. Despite some impressive results, many of these ideas have not been widely adopted by the robotics community, and many practitioners continue to ignore the group structure of rotations. One issue preventing wider adoption is that accounting for this structure requires low-level changes to the underlying optimization algorithms, which is difficult or impossible when relying on existing off-the-shelf solvers.

To address this lack of solver support, we formulate a straightforward, generic method for adapting existing Newton and gradient-based algorithms to correctly account for the group structure of rotations. Based entirely on basic mathematical tools from linear algebra and calculus, our method is computationally efficient and should be both accessible and familiar to most roboticists. We apply this method to the ALTRO solver [**howell2019altro**] and demonstrate its performance on a constrained trajectory optimization problem.

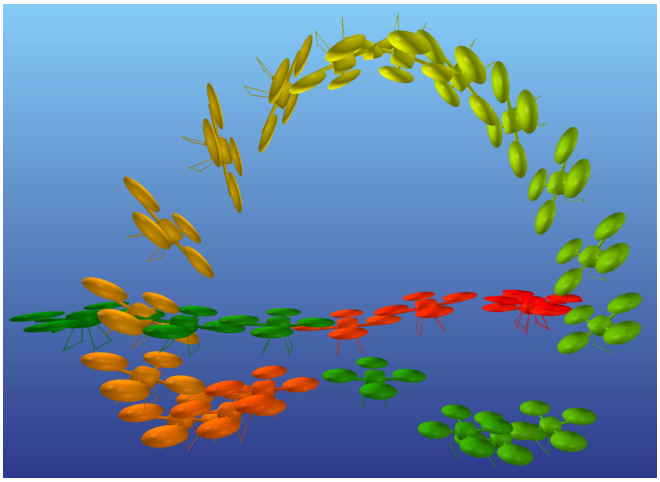

Fig. 1. Snapshots of quadrotor flip trajectory solved using a modified version of ALTRO that correctly accounts for the group structure of rotations. These modifications make it easier to find trajectories that undergo large changes in attitude, such as the loop-de-loop shown here.

II. NOTATION AND CONVENSIONS

We begin by defining some useful conventions and notation. Attitude is defined as the rotation from the robot's body frame to a global inertial frame. We also define gradients to be row vectors, that is, for $f(x) : \mathbb{R}^n \to \mathbb{R}$, $\frac{\partial f}{\partial x} \in \mathbb{R}^{1\times n}$.

We represent—following the Hamilton convention—a quaternion $\mathbf{q} \in \mathbb{H}$ as a standard vector $q \in \mathbb{R}^4 := [q_s \;\; q_v^T]^T$ where $q_s \in \mathbb{R}$ and $q_v \in \mathbb{R}^3$ are the scalar and vector part of the quaternion, respectively. We denote the composition of two quaternions as $\mathbf{q}_3 = \mathbf{q}_2 \otimes \mathbf{q}_1$. We use $[x]^\times$ to denote the skew-symmetric matrix such that $[x]^\times y = x \times y$.

III. QUATERNION DIFFERENTIAL CALCULUS

Most modern methods for planning and control require derivatives with respect to the state vector. We present a simple but powerful method for taking derivatives of quaternion-valued state vectors. The extension of this method to more general state vectors that contain Cartesian products of $SO(3)$, $SE(3)$, and $\mathbb{R}^n$ is straightforward.

The key idea of this section is that differential quaternions live in three-dimensional space instead of the four-dimensional space of quaternions; this idea should match our intuition given rotations are inherently three-dimensional and differential rotations should live in the same space as angular velocities, i.e. $\mathbb{R}^3$.

In practice, we have found Rodrigues Parameters to be a very effective representation for three-dimensional differential rotations since they are computetionally efficient and do not inherit the sign ambiguity of quaternions.

The mapping between a vector of Rodrigues parameters $\phi \in \mathbb{R}^3$ and a unit quaternion q is known as the Cayley map:

$$q = \varphi(\phi) = \frac{1}{\sqrt{1 + \|\phi\|^2}} \begin{bmatrix} 1 \\ \phi \end{bmatrix}. \tag{1}$$

We will also make use of the inverse Cayley map:

$$\phi = \varphi^{-1}(q) = \frac{q_v}{q_s}. \tag{2}$$

A. Jacobian of Vector-Valued Functions

When taking derivatives with respect to quaternions, we must take into account both the composition rule and the

[1]Robotics Institute, Carnegie Mellon University, 5000 Forbes Ave, Pittsburgh, PA, USA

nonlinear mapping between the space of unit quaternions and our chosen three-parameter error representation.

Let $\phi \in \mathbb{R}^3$ be a differential rotation applied to a function with unit quaternion inputs $y = h(q) : \mathbb{S}^3 \to \mathbb{R}^p$, such that

$$y + \delta y = h(q \otimes \varphi(\phi)) \approx h(q) + \nabla h(q)\phi. \tag{3}$$

We can calculate the Jacobian $\nabla h(q) \in \mathbb{R}^{p\times 3}$ by differentiating (3) with respect to ϕ, evaluated at $\phi = 0$:

$$\nabla h(q) = \frac{\partial h}{\partial q}\begin{bmatrix} -q_v^T \\ q_s I_3 + [q_v]_\times \end{bmatrix} := \frac{\partial h}{\partial q} G(q) \tag{4}$$

where $G(q) \in \mathbb{R}^{4\times 3}$ is referred to as the *attitude Jacobian*, which essentially becomes a "conversion factor" allowing us to apply results from standard vector calculus to the space of unit quaternions. This form is particularly useful in practice since $\partial h/\partial q \in \mathbb{R}^{p\times 4}$ can be obtained using finite difference or automatic differentiation. As an aside, although we have used Rodrigues parameters, $G(q)$ is actually the same (up to a constant scaling factor) for any choice of three-parameter attitude representation.

B. Other derivatives

Using similar techniques, we find useful expressions for the Hessian of a scalar-valued function (i.e. $p = 1$):

$$\nabla^2 h(q) = G(q)^T \frac{\partial^2 h}{\partial q^2} G(q) + I_3 \frac{\partial h}{\partial q} q, \tag{5}$$

and the Jacobian of a quaternion-valued function $q' = f(q) : \mathbb{S}^3 \to \mathbb{S}^3$:

$$\nabla f(q) = G(q')^T \frac{\partial f}{\partial q} G(q). \tag{6}$$

IV. Fast Constrained Trajectory Optimization with Attitude States

A. Algorithmic Modifications

Here we briefly outline the changes to the ALTRO solver [**howell2019altro**] to efficiently solve trajectory optimization problems for systems with 3D rotations in the state vector.

The linearization of the nonlinear discrete dynamics function $x_{k+1} = f(x_k, u_k)$ is "corrected" using 6:

$$A = E(f(x,u))^T \frac{\partial f}{\partial x} E(x); \quad B = E(f(x,u))^T \frac{\partial f}{\partial u}. \tag{7}$$

Here we define the *state attitude Jacobian* $E(x)$ to be a block-diagonal matrix where the block is an identity matrix for any vector-valued state and $G(q)$ for any quaternion in the state vector. Using (4), (5), (6) we can derive similar modifications to the expansions of the cost and constraint functions. We refer the reader to the original ALTRO paper [**howell2019altro**] for futher details on the algorithm.

These adjustments effectively modify the optimization so that it is performed on differential rotations (or Rodrigues Parameters) instead of the space of unit quaternions. In practice, this is analagous to the Multiplicative Extended Kalman Filter [**markley2014fundamentals**] in the state estimation community.

The iLQR algorithm derives a locally-optimal feedback policy of the form $\delta u = K\delta x + d$ during the "backward pass" of each iteration, which is then used to simulate the system forward during the "forward pass". Instead of computing δx using simple substraction as we would in the original algorithm, we now compute the error using the inverse Cayley map (2) for the quaternion states. The rest of the forward pass is unmodified.

B. Examples

To demonstrate the effectiveness of the proposed approach, we tested the modified version of ALTRO by solving for a quadrotor flip trajectory (see Figure II). Four soft "waypoint" costs were placed along the trajectory, encouraging the quadrotor to follow a loop. The trajectory was initialized with hover controls and a state trajectory that smoothly rotated the quadrotor one full rotation about the x-axis while moving to the goal. The modified version of ALTRO solved the problem in 140 ms and 22 iterations, while the original version that naïvely treated the quaternion as a vector in $\mathbb{R}^4$ solved in 420 ms and 58 iterations. The original version failed to provide a nice, smooth loop and instead reversed direction halfway through the loop.

It's also worthwhile to note that this problem cannot be solved as-is with any three-parameter representation because of the singularities that occur at 90, 180, and 360 degrees for Euler angles, Rodrigues parameters, and MRP's, respectively.

V. Conclusion

We have proposed a general, accessible, and computationally efficient approach to doing planning and control for systems with one or more unit quaternions in their state vectors. The approach allows for straightforward adaptation of many gradient and Newton-based methods for optimal control and motion planning, which we demonstrated on the ALTRO solver. By exploiting the unique structure of both the trajectory optimization problem and the rotation group, the newly modified version of ALTRO will likely be able to solve more challenging problems with performance. Future work will extend these ideas to Newton-based methods such as direct collocation and other domains outside of trajectory optimization.