In Proc. 2004 IEEE Int'l Conf. on Robotics and Automation (ICRA 2004)

Effective Sampling and Distance Metrics for 3D Rigid Body Path Planning

James J. Kuffner

The Robotics Institute
Carnegie Mellon University
5000 Forbes Ave., Pittsburgh, PA 15213, USA
email: kuffner@cs.cmu.edu

Digital Human Research Center
National Institute of Advanced
Science and Technology (AIST)
2-41-6 Aomi, Koto-ku, Tokyo, Japan 135-0064

***Abstract*— Important implementation issues in rigid body path planning are often overlooked. In particular, sampling-based motion planning algorithms typically require a *distance metric* defined on the configuration space, a *sampling function*, and a method for *interpolating sampled points*. The configuration space of a 3D rigid body is identified with the Lie group SE(3). Defining proper metrics, sampling, and interpolation techniques for SE(3) is not obvious, and can become a hidden source of failure for many planning algorithm implementations. This paper examines some of these issues and presents techniques which have been found to be effective experimentally for Rigid Body path planning.**

I. Introduction

The configuration space ($\mathcal{C}$-space) of a 3D rigid body is usually defined as the set of all possible positions and orientations of a body-fixed frame relative to a stationary world frame. This identifies the $\mathcal{C}$-space with the *Lie group* $SE(3)$, the special Euclidean group in three-dimensions. Geometric path planning problems defined on $SE(3)$ arise in a number of important application domains including mechanical assembly planning and part removability analysis, control of free-flying robots and UAVs, satellite motion, and biochemical simulations of molecular protein docking. The topology of the space induced by the rotation component of $SE(3)$ has important implications for rigid body path planning algorithms intended to efficiently represent and search this space. Unless implemented carefully, these representational details can become a hidden source of failure.

This paper investigates several important implementation issues in rigid body path planning that are often overlooked. In particular, sampling-based motion planning algorithms typically require: 1) a *distance metric* defined on the configuration space, 2) a function to *generate a sample* in the space, and 3) a method for *interpolating sampled points*. Defining proper metrics, sampling methods, and interpolation techniques for $SE(3)$ is not immediately obvious.

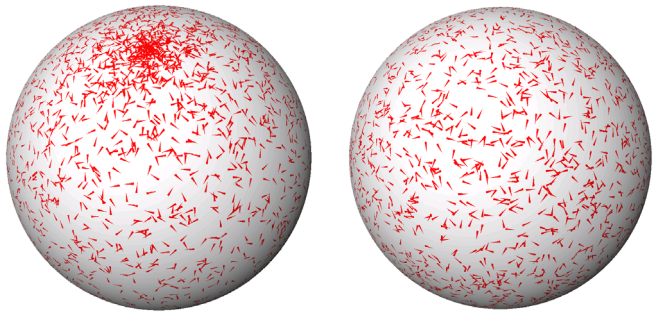
This paper explores some of these implementation issues and presents techniques which have been found to be effective experimentally for rigid body path planning. In particular, methods for generating a uniform distribution of randomly sampled rotations for both Euler angle and quaternion parameterizations are given in Section IV. Examples of distance metrics on $SE(3)$ and geodesic interpolation functions for rotations are presented in Section V and Section VI respectively. Section VII shows experimental results aimed at evaluating the computational performance and tradeoffs for different implementations, and Section VIII concludes with a summary discussion.

II. Background

The space of configurations of a 3D rigid body is usually defined as the set of all possible positions and orientations of a body-fixed frame relative to a stationary world frame. This identifies the $\mathcal{C}$-space with the *Lie group* $SE(3)$, as it has the structure of both a differentiable manifold and an algebraic group (under matrix multiplication).

Because of its importance, the characteristics of $SE(3)$ and its structural properties has been a topic of research across a number of different fields, including physics, mathematics, computer graphics, engineering and robotics. One of the fundamental results of rigid body mechanics which was proved by Chasles in the early 19th century is that any rigid body displacement can be realized by a rotation about an axis combined with a translation parallel to that axis [1]. Recall that a particular displacement can be understood as an element of $SE(3)$, while a motion is a curve on $SE(3)$. If the rotation from Chasles's theorem is performed at constant angular and translational velocity, the motions are commonly referred to as *screw motions*, and have been used as the basis for defining a number of metrics on $SE(3)$ [2]. A brief discussion of the geometry of $SE(3)$ can be found in the appendix of [3]. In the context of path planning in $SE(3)$, there has been some previous work aimed at evaluating implementation issues. Different distance metrics for planning are evaluated and discussed in [4], with further details in [5]. This work tackles the issue of relative weighting of translation and rotation components of $SE(3)$ (see Section V), though only in the context of an euler angle representation of rotation.

Other related research efforts have focused mainly on developing kinematic metric functions: distance and Riemannian metrics. It has been proven that there exist no bi-invariant Riemannian metrics on $SE(3)$ [6], and that there are no differentiable bi-invariant distance metrics on $SE(3)$ [7]. Lin and Burdick provide an excellent summary of the details of these results in the context of frame-invariant kinematic metric functions in [8].



(a) Naïve sampling (b) Uniform sampling

Fig. 1. Naïve sampling (a) and uniform sampling (b) of $SO(3)$ using euler angles. There are a total of 5000 rotation samples, with each sample visualized as an oriented arrow on the surface of the unit sphere.

III. Representing Rotations in Three Dimensions

There are a variety of conventions for representing and parameterizing rotations in three dimensions. In this section, three of the more popular representations are discussed: *rotation matrices*, *euler angles*, and *unit quaternions*. A number of important characteristics and tradeoffs exist in terms of performance, storage efficiency, numerical

stability, and ease of use. Several of these tradeoffs are summarized in the paragraphs that follow.

A. Rotation Matrices

Defining $SE(3)$ as the set of all possible positions and orientations of a body-fixed frame relative to a stationary world frame naturally leads to a matrix representation. As a homogeneous matrix, this is usually written as:

$$M = \begin{bmatrix} R & X \\ 0 & 1 \end{bmatrix}$$

Considering just the rotation subcomponent R, valid rotations are comprised of all 3x3 orthonormal matrices with unit determinant. Although nine numbers are used to specify this matrix, there are a total of six constraints: three for keeping the columns of R to be of unit length, and three pairwise orthogonality constraints between the columns. Thus, a total of $9-6=3$ degrees of freedom exist.

Although seemingly convenient at first, matrix representations of rotations suffer from a number of problems when implemented in a finite-precision computing system. Aside from being space inefficient in terms of memory usage, matrices often suffer from numerical drift during use as a result of the underlying floating-point approximation of real numbers. For example, multiplying two rotation matrices together will result in a matrix that represents the composed rotations. However, due to floating-point error, the resulting matrix will often be *not quite* orthonormal, and the most appropriate method for *re-normalizing* the matrix is typically ill-defined.

For path planning applications, we are interested in sampling, distance metrics, and interpolation of rotations. Unfortunately, it is unclear how to easily define a function $\rho(R_1,R_2)$ that represents the "distance" between two rotation matrices, or to interpolate between two matrices R_1 and R_2 in order to generate a series of smooth intermediate rotations.

B. Euler Angles

According to Euler's rotation theorem, any orientation can be described by three successive rotations (θ,ϕ,η) about certain sets of three axes (v_1,v_2,v_3). Since rotations do not commute, the order in which rotations are applied about these axes is important. There are at least 24 standard euler angle conventions in use depending upon which axes are used and the order in which the rotations are applied. For details of these conventions explained in a robotics context, see [9].

Euler angles are compact: three angles for three rotational degrees of freedom. They also are stable numerically, relatively computationally efficient, and are considered to be more intuitive to work with and visualize than matrices. Due to their simplicity, euler angles have been used in a number of path planning implementations. Unfortunately, there are problems with using euler angles to represent rotations. Within a given euler angle convention, there are multiple sets of parameter values which can yield the same rotation, leading to a fundamental ambiguity. This ambiguity exists due to the interdependence of the rotations, which also manifests itself when two or more axes happen to align, causing a loss of a degree of freedom known as "gimbal lock". But more importantly, euler angles have serious problems in the context of path planning, namely: proper sampling, interpolation, and distance metrics. These issues are discussed in more detail throughout the remaining sections of this paper.

C. Unit Quaternions

Hamilton formulated over a century ago the mathematics with which a vector of unit magnitude with four components can be used to parameterize rotations in three dimensions. The intuition behind quaternions is apparent by considering their relationship to *axis-angle* pairs. Namely, Euler showed that any arbitrary orientation in three dimensions could be achieved by a single rotation θ about a single axis $v=(v_x,v_y,v_z)$. The corresponding unit quaternion is given by:

$$Q=(w,x,y,z)=(cos(\frac{\theta}{2}),v_x sin(\frac{\theta}{2}),v_y sin(\frac{\theta}{2}),v_z sin(\frac{\theta}{2}))$$

These four scalar numbers still have only three degrees of freedom due to the unit magnitude constraint $||Q||=1$. For further derivation and details, see [10].

Unit quaternions are relatively compact and efficient to work with. Computing with quaternions can introduce slight numerical drift due to floating-point errors. However, it is fortunately straightforward to renormalize a quaternion by simply dividing each quaternion component by the magnitude of its length, resulting once again in a 4-vector of unit length. In the context of path planning, unit quaternions are an excellent choice for representing rotations since it is relatively easy to define proper methods for sampling, interpolation, and computing a measure of distance between quaternion rotations. Several of these techniques are explained and investigated in the following sections.

IV. Sampling Issues

The majority of popular heuristic path planning algorithms for searching high-dimensional configuration spaces are *sampling-based* planners. Examples include the probabilistic roadmap (PRM) and its variations [11], [12], and single-query path planners like the RRT [13] or expansive-space planners [14], [15]. These planners build graphs or trees of connected sampled configurations that attempt to approximate the connectivity of the actual free configuration space, $\mathcal{C}_{free}$. When applied to searching the 6-dimensional space of rigid body motions, some care must be taken.

A fundamental component of sampling-based motion planners is a function to incrementally generate samples in the configuration space $\mathcal{C}$. For spaces such as an n-dimensional cube in $\Re^n$, a standard pseudo-random number generator can be used to generate samples, or for better uniformity in terms of dispersion and discrepancy, a deterministic sequence of quasi-random numbers has been shown to offer advantages [16].

Sampling the space of rigid body configurations $SE(3)$ is more complicated than sampling a cube in $\Re^n$ due to the topology of the space. Here, the choice of parameterization for the rotation component is important. Assume that we have available a pseudo-random number generator function $rand()$, which returns a floating point number on the range $[0,1)$. Sampling the translation component is straightforward, as we can simply generate independent random values along each axis and scale by the axis dimension:

$$(x,y,z)=(X_{dim}rand(),Y_{dim}rand(),Z_{dim}rand())$$

The rotation component must be handled differently. In particular, care must be taken so that the resulting distribution of samples is not *biased* to favor specific rotations. Rather, we desire our sampling function to yield a uniform distribution of rotations in the limiting case. In the context of path planning, having a uniform sampling distribution will prevent search algorithms from oversampling or undersampling large portions of the $\mathcal{C}$-space. This affects both the performance and reliability of planning algorithms.

Intuitively, picking a random rotation axis and a random angle will generate the desired distribution of rotations. We can visualize this as randomly oriented objects distributed uniformly across the surface of

a three-dimensional sphere. An equivalent intuitive way of iteratively accomplishing this effect is to successively rotate objects located at the north pole vertically by a random amount, and then rotate the axis of the north pole to a random position on the sphere. This is the inspiration behind Jim Arvo's method for generating fast random rotation matrices derived in [17].

A. Uniform Sampling of Euler Angles

A naïve attempt at uniformly sampling euler angles might try to uniformly sample each angle independently. However, this results in a distribution that is heavily biased towards "polar" regions according to the set of rotation axes. In Figure 1, sets of oriented arrows on the surfaces of two spheres are shown as visual representations of sampling distributions in $SO(3)$ of 5000 Roll-Pitch-Yaw euler angles. The first shows the result of a naïve sampling, while the second shows a uniformly distributed sampling of rotations. Notice the concentration of samples at the poles when using the naïve method. This can adversely affect the convergence of many sampling-based path planning algorithms.

In addition to the lopsided distribution, multiple angle sets may map to the same rotation over a large portion of $\mathcal{C}$. For example, if the limits are set equally on the range $[-\pi,+\pi]$, the parameterization results in a *double coverage* of the space of rotations. To prevent this, the range of one of the angles should be set to $[-\frac{\pi}{2},+\frac{\pi}{2}]$ while the other two should be set to $[-\pi,+\pi]$. Which angle should have the reduced range depends on which of the 24 euler angle conventions used.

Fortunately, there exists a simple and efficient way to generate uniform random distributions of euler angles. Figure 1(b) shows a uniform distribution of 5000 random Roll-Pitch-Yaw angles that was generated using a method based on uniform spherical sampling. The idea is to generate uniform distributions on the range $[-\frac{\pi}{2},\frac{\pi}{2})$ for both θ and η, and use the inverse cosine relationship for ϕ to avoid oversampling the polar regions. Since the $\arccos()$ maps from the domain $[-1,+1]$ to $[0,\pi]$, an adjustment should be made if necessary to map to the proper range of ϕ. An example implementation in pseudocode is given in Algorithm 1.

Algorithm 1: Pseudocode to generate uniformly-distributed random Roll-Pitch-Yaw euler angles.

Input: *none*
Result: uniform random euler angles (θ,ϕ,η)
$\theta = 2\pi * rand() - \pi$;
$\phi = \arccos(1-2*rand()) + \frac{\pi}{2}$;
if $rand() < \frac{1}{2}$ **then**
  if $\phi < \pi$ **then** $\phi = \phi + \pi$;
  else $\phi = \phi - \pi$;
end
$\eta = 2\pi * rand() - \pi$;
return (θ,ϕ,η)

B. Uniform Sampling of Unit Quaternions

Generating uniformly distributed random unit quaternions is relatively straightforward. If we have already computed a uniform random axis v and angle θ, we can use the equation in Section III-C to generate the equivalent unit quaternion. However, a direct and more efficient method of computing uniform random quaternions is derived in [18]. This method utilizes three intermediate random variables to compute four quaternion parameters that map uniformly to the unit sphere in four dimensions. This calculation produces a quaternion of unit length, so it is not necessary to renormalize the result. Performance comparisons of both Algorithm 1 and Algorithm 2 are included in Section VII.

Algorithm 2: Pseudocode to generate uniformly-distributed random unit quaternions.

Input: *none*
Result: uniform random quaternion $Q = (w,x,y,z)$
$s = rand()$;
$\sigma_1 = \sqrt{1-s}$;
$\sigma_2 = \sqrt{s}$;
$\theta_1 = 2\pi * rand()$;
$\theta_2 = 2\pi * rand()$;
$w = cos(\theta_2) * \sigma_2$;
$x = sin(\theta_1) * \sigma_1$;
$y = cos(\theta_1) * \sigma_1$;
$z = sin(\theta_2) * \sigma_2$;
return (w,x,y,z)

V. Distance Metric Issues

Many sampling-based planning algorithms require a distance metric be defined over $\mathcal{C}$ in order to give an approximate measure of the "closeness" between pairs of configurations. We define the symmetric scalar function:

$$\rho(q_0,q_1) \mapsto \Re \qquad q_0,q_1 \in \mathcal{C}$$

that returns a measure of the relative distance between the configurations q_0 and q_1. The efficiency and accuracy of the distance metric can have a large impact on the efficacy of the planning algorithm.

Intuitively, an ideal metric for path planning in $SE(3)$ would correspond to a measure of the minimum *swept-volume* in the workspace while moving a rigid object from one configuration to another. Intuitively, minimizing the swept-volume will minimize the chance of collision with obstacles, which in turn maximizes the chance of discovering collision-free paths between pairs of configurations in $\mathcal{C}$. Unfortunately, computing the *exact* swept-volume is a notoriously difficult geometric problem, and although recently developed approximate techniques have demonstrated improved efficiency [19], [20], they are currently too expensive for path planning. Instead, heuristic metrics are typically defined that generally attempt to approximate the ideal swept-volume metric. The most simple and commonly used metrics consider the $\mathcal{C}$-space as a Cartesian space and define a Euclidean metric. For example, if X and R represent the translation and rotation components of the configuration $q = (X,R) \in SE(3)$ respectively, then:

$$\rho(q_0,q_1) = w_t||X_0 - X_1|| + w_r f(R_0,R_1)$$

is a weighted metric with the translation component $||X_0 - X_q||$ using a standard Euclidean norm, and the positive scalar function $f(R_0,R_1)$ returning an approximate measure of the distance between the rotations $R_0,R_1 \in SO(3)$. The rotation distance is scaled relative to the translation distance via the weights w_t and w_r. One of the difficulties with this method is deciding proper weight values. Previous research has suggested that the relative importance of the rotation component decreases as the planning queries become harder (see [4], [5] for a discussion).

A. Euler Angle Distance Metric

For the case of using roll-pitch-yaw euler angles for rotation, we first define a distance function $\Delta(\theta_1, \theta_2)$ that returns the difference between two angles θ_1 and θ_2. When subtracting angles that "wrap-around", there two possible interpolation directions, so care must be taken to use the "shortest path" direction between the angles:

Algorithm 3: Computes "shortest path" difference between two angles (Hereafter denoted by the function $\Delta(\theta_1, \theta_2)$).

Input: Two angles θ_1 and θ_2
Result: "shortest path" angle difference $\delta\theta \in [-\pi, +\pi]$
$\delta\theta = \theta_2 - \theta_1$;
// normalize $\delta\theta$ *on the range* $[-\pi, +\pi)$
if $\delta\theta < -\pi$ **then** $\delta\theta = \delta\theta + 2\pi$;
else if $\delta\theta < \pi$ **then** $\delta\theta = \delta\theta - 2\pi$;
return $\delta\theta$

The sum of each of the euler angle differences gives an approximate measure of the distance between the two rotations, as in Algorithm 4:

Algorithm 4: Compute approximate euclidean distance metric between two sets of euler angles.

Input: euler angles $(\theta_1, \phi_1, \eta_1)$ and $(\theta_2, \phi_2, \eta_2)$
Result: The weighted rotation distance component ρ_r.
$\rho_r = w_r * \sqrt{\Delta(\theta_1, \theta_2)^2 + \Delta(\phi_1, \phi_2)^2 + \Delta(\eta_1, \eta_2)^2}$;
return ρ_r

However, this measure of distance between euler angles does not correctly handle multiple representations of the same rotation. In other words, it is possible that the summed difference between each of the individual angle components is large while the actual rotations they represent are very close or even identical.

B. Unit Quaternion Distance Metric

Unlike euler angles, it is possible to derive a geodesic metric for unit quaternion representations of $SO(3)$. As we shall see again in Section VI-B, the "great circle arc" on the 4D unit sphere between two unit quaternions defines a geodesic path for interpolating two rotations. This suggests a number of possible metrics. Park and Ravani have defined a bi-invariant distance metric for $SO(3)$ in [7], which has been used by Choudhury and Lynch for rolling manipulation planning [21]:

$$\rho_r = ||log(Q_1^{-1} Q_2)||$$

An alternative metric that provides a simple and convenient measure of approximate rotation distance can be defined using an inner product. Given two unit quaternions $Q_1 = (w_1, x_1, y_1, z_1)$ and $Q_2 = (w_2, x_2, y_2, z_2)$, we define the inner product λ of two quaternions as:

$$\lambda = Q_1 \cdot Q_2 = w_1 w_2 + x_1 x_2 + y_1 y_2 + z_1 z_2$$

This is the scalar inner product of two 4D unit vectors. As in 3D, the angle α formed by this pair of vectors is related to the inner product by its cosine:

$$\alpha = \arccos(Q_1 \cdot Q_2)$$

The length of the geodesic path on the 4D unit sphere is proportional to α. However, there is an important property of unit quaternions that must be considered: polar opposite points on the 4D unit sphere are identified, meaning there are exactly two unit quaternion representations for the same rotation. For example, both $Q = (w, x, y, z)$ and its opposite $-Q = (-w, -x, -y, -z)$ represent the same rotation. An intuitive explanation for this is based on the equivalent angle-axis representation: a rotation of θ about an axis v results in the same orientation as a rotation of $-\theta$ about the opposite axis $-v$.

To account for multiple representations in our definitions of unit quaternion distance metrics and interpolation schemes, we can simply test whether λ is negative, and negate one of the quaternions to obtain its equivalent alternative. Note that if $\lambda = Q_1 \cdot Q_2$, then $-\lambda = -Q_1 \cdot Q_2 = Q_1 \cdot -Q_2$. Algorithm 5 computes a scalar approximate distance measure ρ_r that returns a value on the range $[0, w_r]$, where w_r is the rotation weight.

Algorithm 5: Compute approximate distance metric between two unit quaternions

Input: Two unit quaternions Q_1 and Q_2
Result: The weighted rotation distance component ρ_r on the range $[0, w_r]$.
// compute the quaternion inner product λ
// (The result is on the range $[-1, 1]$*)*
$\lambda = Q_1 \cdot Q_2$;
$\rho_r = w_r * (1 - ||\lambda||)$;
return ρ_r

This metric obeys the triangle inequality and is fairly efficient to compute.

VI. Configuration Interpolation Issues

The problem of interpolating two configurations in $SE(3)$ shares much in common with the problem of defining metrics. We would like to have an efficient and accurate method for calculating a continuous series of intermediate transformations between two given transformations. For 3D rigid body path planning implementations that interpolate the translation and rotation components separately, let us consider the problem of interpolating two rotations in $SO(3)$.

A. Interpolation of Euler Angles

The obvious but naïve way to interpolate two rotations represented by sets of euler angles is simply to linearly interpolate each angle independently:

Algorithm 6: linear interpolation between two sets of euler angles by the fraction $f \in [0, 1]$.

Input: two euler angle sets $(\theta_{start}, \phi_{start}, \eta_{start})$ and $(\theta_{end}, \phi_{end}, \eta_{end})$
Result: interpolated euler angles (θ, ϕ, η)
$\theta = \theta_{start} + f * \Delta(\theta_{start}, \theta_{end})$;
$\phi = \phi_{start} + f * \Delta(\phi_{start}, \phi_{end})$;
$\eta = \eta_{start} + f * \Delta(\eta_{start}, \eta_{end})$;
return $Normalize(\theta, \phi, \eta)$

Using the function $\Delta(\theta_1, \theta_2)$ from Algorithm 3, we compute the "shortest" path of interpolation between each of the angle components

and normalize the resulting angles on the desired range according to the euler angle convention used. Observe that this implementation suffers from the same problems arising from multiple representations that can occur when computing distances. Namely, two sets of euler angles with relatively large differences in individual angle values may actually map to very similar or identical rotations in $SO(3)$. This produces a relatively large swept-volume resulting from the spurious interpolated values, which is disadvantageous to path planning. This problem along with the difficulty in defining metrics generally makes euler angles a poor choice for representing the rotation component of $SE(3)$ in path planning applications.

B. Interpolation of Unit Quaternions

Perhaps the biggest advantage to using unit quaternions to represent configurations of $SO(3)$ is the ability to smoothly interpolate between two configurations along geodesics. As mentioned in Section V-B, the great-circle arc between two points on the surface of the 4D unit sphere represents a geodesic interpolation path between two unit quaternions. Points along this curve represent configurations in $SO(3)$ that correspond to a set of smoothly-varying intermediate rotations. One simple way to generate this set of points is to linearly interpolate two unit quaternions as points in $\Re^4$ and projecting the result onto the 4D unit sphere. As in Section V-B, care must be taken to ensure an interpolation path of minimal length by computing the alternative representation of one of the unit quaternions.

Algorithm 7: Calculates an approximate linear interpolation between two quaternions by the fraction $f \in [0,1]$ using a straight line in $\Re^4$ projected onto the unit quaternion sphere.

Input: two quaternion rotations Q_1 and Q_2
Result: interpolated quaternion $Q = (w,x,y,z)$
// compute the quaternion inner product
$\lambda = Q_1 \cdot Q_2$;
if $\lambda < 0$ **then**
  // the quaternions are pointing in opposite directions, so
  // use the equivalent alternative representation for Q_2
  $w_2 = -w_2$; $x_2 = -x_2$; $y_2 = -y_2$; $z_2 = -z_2$;
end
$w = w_1 + f * (w_2 - w_1)$;
$x = x_1 + f * (x_2 - x_1)$;
$y = y_1 + f * (y_2 - y_1)$;
$z = z_1 + f * (z_2 - z_1)$;
$Q = (w,x,y,z)$;
// normalize the result
$Q = \frac{Q}{||Q||}$;
return Q

The projection on the unit sphere in $\Re^4$ is done by simply normalizing the interpolated vector:

$$Q = \frac{Q}{||Q||} = \frac{Q}{\sqrt{w^2 + x^2 + y^2 + z^2}}$$

The drawback to this approximate scheme is that the interpolated intermediate points will not be evenly-spaced along the geodesic, especially if the two endpoint rotations are dissimilar. A better method is to use *spherical linear interpolation* or "slerp", as it is commonly called [10]. Algorithm 8 shows an implementation that first computes the nearby representation for two unit quaternions, and then calculates their inner product. If the rotations are very close (the inner product is smaller than ε), then linear interpolation from Algorithm 7 is performed. Otherwise, spherical linear interpolation factors are computed that result in evenly-distributed intermediate points along the geodesic arc. We normalize the final resulting intermediate quaternion in order to prevent numerical drift resulting from floating-point approximation errors.

Algorithm 8: interpolate quaternions using "slerp" (spherical linear interpolation)

Input: two quaternion rotations Q_1 and Q_2
Result: interpolated quaternion $Q = (w,x,y,z)$
// compute the quaternion inner product
$\lambda = Q_1 \cdot Q_2$;
if $\lambda < 0$ **then**
  // the quaternions are pointing in opposite directions, so
  // use the equivalent alternative representation for Q_2
  $w_2 = -w_2$; $x_2 = -x_2$; $y_2 = -y_2$; $z_2 = -z_2$;
  $\lambda = -\lambda$;
end
// calculate interpolation factors
if $||1 - \lambda|| < \varepsilon$ **then**
  // the quaternions are nearly parallel, so use
  // linear interpolation
  $r = 1 - f$;
  $s = f$;
else
  // calculate spherical linear interpolation factors
  $\alpha = \arccos(\lambda)$;
  $\gamma = \frac{1}{\sin\alpha}$;
  $r = \sin((1-f) * \alpha) * \gamma$;
  $s = \sin(f * \alpha) * \gamma$;
end
// set the interpolated quaternion
$w = r * w_1 + s * w_2$;
$x = r * x_1 + s * x_2$;
$y = r * y_1 + s * y_2$;
$z = r * z_1 + s * z_2$;
$Q = (w,x,y,z)$;
// normalize the result
$Q = \frac{Q}{||Q||}$;
return Q

VII. Experimental Results

Based on the techniques and sample code presented in the previous sections, experiments were conducted to evaluate the relative performance in terms of computation speed and planning efficiency. Results were calculated using the GNU *gcc* compiler with level 2 optimization running in VMware on a 1.1 GHz Pentium III with 512MB or RAM. Table I compares the performance of different representations of rotation. Numerous trials where conducted $N = 1,000,000$ to collect the average absolute time (in seconds) required to calculate random sampling functions, distance metrics, and interpolation functions. In addition to this raw performance evaluation, experiments were conducted to determine the effects of representation on the success and performance of a randomized path planning strategy based on Rapidly-exploring Random Trees (RRTs) [13]. This planner has been used to efficiently solve several well-known motion planning benchmarks, such as the "Alpha 1.0 puzzle" (Figure 2) and "Flange 1.0 problem" (Figure 3 - *left*), which are

TABLE I
PERFORMANCE COMPARISON OF ROTATION REPRESENTATIONS

Rotation Rep.	Sample	$\|\|L\|\|_2$ Dist.	$(\|\|L\|\|_2)^2$	Interp.
Euler angles	1.70	0.40	0.22	0.40
Quat. (w/ lerp)	2.05	0.28	0.15	0.54
Quat. (w/ slerp)	2.15	0.28	0.14	1.03

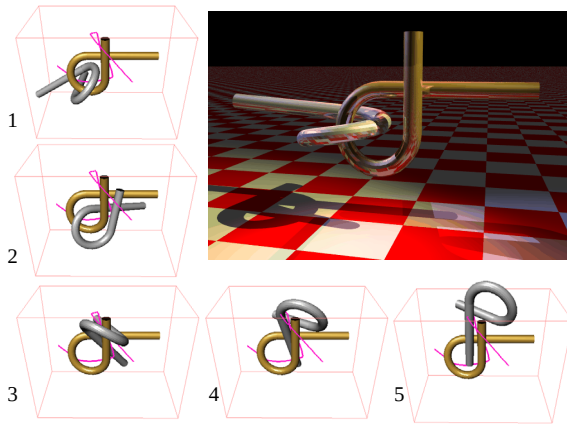


Fig. 2. Solving the "Alpha Puzzle" 1.0 Benchmark.

considered difficult due to the presence of narrow passages in the configuration space [4]. Based on my own numerous trials and experiments using different metric and interpolation schemes, two general recommendations can be made: 1) If you expect to find paths through narrow passages in C, using a weighted quaternion distance metric $||L||_2$ with slerp interpolation of samples yields the best performance for RRT-based path planners. Roughly a 45% speedup in average calculation time can be expected over using euler angles to represent rotation depending on the difficulty of the problem. For the alpha 1.0 puzzle, only the quaternion-based implementations were able to consistently solve the query. 2) If the majority of the queries you will face involve relatively large areas of open space in C_{free}, then the performance advantages of using quaternions and slerp interpolation over other, less-costly alternatives are still apparent in terms of the average number of samples needed to solve a particular query, but less apparent in terms of the overall average running time.

VIII. SUMMARY AND DISCUSSION

Geometric path planning problems defined on $SE(3)$ arise in a number of important application domains. Rigid body path planning algorithms intended to efficiently represent and search this space need to take into account the structure and topology of $SE(3)$, and implement carefully the sampling, distance metrics, and interpolation functions. Utilizing unit quaternions to represent the rotation component has been found to be both efficient and effective for path planning, and is recommended over other alternatives such as euler angles. Future work includes gathering additional experimental performance data for path planning implementations across a wider range of queries and obstacle arrangements, and investigating alternative quaternion distance metrics in the hopes of achieving a deeper understanding of performance tradeoffs in terms of computation costs, average overall planning time, and the overall rate of success in finding a path.

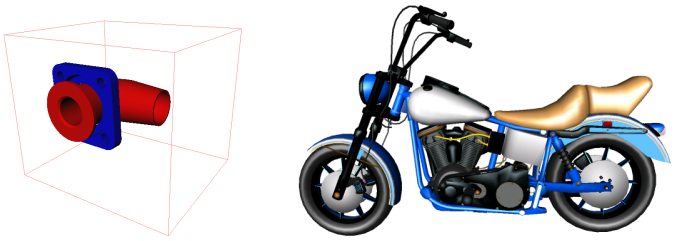


Fig. 3. *Left:* The "Flange" Benchmark. *Right:* Part removability test for assembly analysis of a detailed motorcycle and engine model.

ACKNOWLEDGMENT

I thank Steve LaValle and Al Rizzi who provided helpful insights and pointers to related research. This research was partially supported by NSF grants ECS-0325383, ECS-0326095, and ANI-0224419.

REFERENCES

[1] J. McCarthy, *An Introduction to Theoretical Kinematics*. MIT Press, 1990.

[2] G. Zefran, M. Kumar, and V. Croke, "Metrics and connections for rigid body kinematics," *Int. J. of Robotics Research*, 1998.

[3] R. Murray and S. Sastry, "Nonholonomic motion planning: Steering using sinusoids," *IEEE Trans. on Automatic Control*, vol. 38, no. 5, pp. 700–716, 1993.

[4] N. Amato, O. Bayazit, L. Dale, C. Jones, and D. Vallejo, "Choosing good distance metrics and local planners for probabilistic roadmap methods," *IEEE Trans. Robot. & Autom.*, vol. 16, no. 4, pp. 442–447, Aug. 2000.

[5] N. M. Amato, O. B. Bayazit, L. K. Dale, C. Jones, and D. Vallejo, "Choosing good distance metrics and local planners for probabilistic roadmap methods," Texas A & M Univ, Tech. Rep. TR98-010, 14, 1998.

[6] J. Loncaric, "Geometric analysis of compliant mechanisms in robotics," Ph.D. dissertation, Harvard Univ., 1985.

[7] F. Park and B. Ravani, "Smooth invariant interpolation of rotations," *ACM Trans. on Graphics*, vol. 16, no. 3, pp. 277–295, July 1997.

[8] Q. Lin and J. Burdick, "Objective and frame-invariant kinematic metric functions for rigid bodies," *Int. J. of Robotics Research*, 1997.

[9] J. J. Craig, *Introduction to Robotics : Mechanics and Control*. Addison-Wesley, 1989.

[10] K. Shoemake, "Animating rotation with quaternion curves," in *Proc. of SIGGRAPH '85*, 1985, pp. 245–254.

[11] L. Kavraki, P. Svestka, J. Latombe, and M. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Trans. Robot. & Autom.*, vol. 12, no. 4, pp. 566–580, 1996.

[12] V. Boor, M. Overmars, and A. van der Stappen, "The Gaussian sampling strategy for probabilistic roadmap planners," in *Proc. of IEEE Int. Conf. Robotics and Automation*, Detroit, MI, 1999.

[13] J. Kuffner and S. LaValle, "RRT-Connect: An efficient approach to single-query path planning," in *Proc. IEEE Int'l Conf. on Robotics and Automation (ICRA'2000)*, San Francisco, CA, Apr. 2000, pp. 995–1001.

[14] D. Hsu, J.-C. Latombe, and R. Motwani, "Path planning in expansive configuration spaces," *Int. J. Comput. Geom. & Appl.*, vol. 9, no. 4-5, pp. 495–512, 1997.

[15] G. Sanchez and J. Latombe, "A single-query bi-directional probabilistic roadmap planner with lazy collision checking," in *Robotics Research: The Tenth Int. Symp. on Robotics Research (ISRR'01)*. Springer, 2001, pp. 403–417.

[16] S. M. LaValle and M. S. Branicky, "On the relationship between classical grid search and probabilistic roadmaps," in *Proc. Workshop on the Algorithmic Foundations of Robotics*, Dec. 2002.

[17] J. Arvo, *Fast Random Rotation Matrices*, ser. Graphics Gems III. Academic Press, 1992, ch. ., pp. 117–120.

[18] K. Shoemake, *Uniform Random Rotations*, ser. Graphics Gems III. Academic Press, 1992, ch. ., pp. 124–132.

[19] P. Xavier, "Fast swept-volume distance for robust collision detection," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 1997, pp. 1162–1169.

[20] Y. J. Kim, G. Varadhan, M. C. Lin, and D. Manocha, "Fast swept volume approximation of complex polyhedral models," in *Proc. SM03*, 2003.

[21] P. Choudhury and K. Lynch, "Rolling manipulation with a single control," in *Conf. on Control Applications*, 2001.