Geometric Computations for Motion Planning

David E. Johnson

I. INTRODUCTION

Path planning done offline, that is, without an ongoing stream of sensor data to interpret, must have a computer representation of the environment and robot. Since one view of path planning is to move between a start and goal configuration without hitting the environment, a corresponding goal in the simulation is for the virtual robot model to move without collision with any of the environment models. As will be seen later, along with collision tests, distance measures between the robot model and environment model are also useful in offline planners. Therefore, it is worth discussing what is meant by collision and distance measures and learning some simple ideas for computing these on modeling primitives such as lines and circles.

A. Collision and Distance

When you think of a collision, you probably think of some portion of two objects occupying (or trying to occupy) the same location in space. Many “common-sense” definitions are more precisely defined using sets and set notation. The collision between two sets can be defined as

\[ C = \{a = b : a \in A, b \in B\} \]

where \( A \) and \( B \) are two sets representing objects in the environment and \( C \) is the collision set. If \( C \) is empty, then there is no collision.

The distance between two sets is more complicated. In general, when distance is used as a measure between two objects, what is really meant is the minimum distance between the two objects, as there are any number of distances between different parts of the objects. So,

\[ \text{minDist}(A, B) = \min\{\text{dist}(a, b) : a \in A, b \in B\}. \]

1) Metrics and Norms: However, we have not yet defined what the distance between two elements, or points, in the sets is, so this definition is not yet complete. Distance is a function defined as a metric on a set \( M \), where \( \text{dist} : M \times M \rightarrow \mathbb{R} \), and a metric space is \( (M, \text{dist}) \) which satisfies

\[ \text{dist}(a, b) = 0 \iff a = b \]
\[ \text{dist}(a, b) = \text{dist}(b, a) \]
\[ \text{dist}(a, b) + \text{dist}(b, c) \geq \text{dist}(a, c) \]

Using these, we can show that \( \text{dist}(a, b) \geq 0 \).

\[ \text{dist}(a, b) + \text{dist}(b, c) \geq \text{dist}(a, c) \]

Substituting \( a \) for \( c \)
\[ \text{dist}(a, b) + \text{dist}(b, a) \geq \text{dist}(a, a) \]
\[ 2\text{dist}(a, b) \geq \text{dist}(a, a) \]
\[ \text{dist}(a, b) \geq 0 \]

Most people think of Euclidean distance when they think of distance, defined for \( n \)-dimensional points \( a \) and \( b \) as
\[ dist(a, b) = \left( \sum_{i=1}^{n} |a_i - b_i|^2 \right)^{\frac{1}{2}}, \]

which does satisfy the definition of a distance function in a metric space. There are a number of other distance metrics possible. Some examples are the Manhattan distance

\[ dist(a, b) = \left( \sum_{i=1}^{n} |a_i - b_i| \right), \]

which corresponds to distance if you are only able to move in one dimension at a time, and the chessboard distance

\[ dist(a, b) = \lim_{p \to \infty} \left( \sum_{i=1}^{n} |a_i - b_i|^p \right)^{\frac{1}{p}}, \]

where diagonals have the same cost as axis-aligned movements. Other metrics are possible by using different values of \( p \), but they have little practical importance.

The concept of these different distance metrics is closely related to that of vector norms. A vector norm \(|a|_p\) is defined as

\[ |a|_p = \left( \sum_{i=1}^{n} |a_i|^p \right)^{\frac{1}{p}}. \]

Since the difference between two points is a vector, these are essentially identical to the distance metrics. These vector norms are also known as L-p norms, such as L-1 and L-2 norms.

2) Dot, Inner, or Scalar Product: A convenient way of computing the Euclidean distance is to find the vector between the two points and compute its L-2 norm using the dot or inner product operation on vectors. The dot product for two vectors \( a \) and \( b \) is

\[ a \cdot b = \sum_{i=1}^{n} a_i b_i \]

If a vector is dotted with itself, the squared L-2 norm results, since

\[ a \cdot a = \sum_{i=1}^{n} a_i a_i = \sum_{i=1}^{n} |a_i|^2 = |a|_2^2 \]

The dot product has several other nice properties which will be used in computing distances between geometric primitives. In the general case

\[ a \cdot b = |a||b| \cos \theta, \]

where \( \theta \) is the angle between the vectors. Therefore, the dot product of perpendicular vectors is zero and the dot product of parallel vectors the product of their norms. If \( b \) is unit length, then the dot product \( a \cdot b \) is the projection of \( a \) onto \( b \), or the length of \( a \) in the \( b \) direction. One can also think of projection as moving the head of \( a \) onto \( b \) as quickly as possible, or along the shortest path. So there is a connection between projection and computing minimum distance.
B. Geometric Computations on Modeling Primitives

This idea of projection now leads to ways of computing distances between various geometric primitives, such as points, lines, circles, and triangles. A point projected onto a primitive gives the straight-line distance to that object, or the minimum distance. For example, the closest point on a plane \( P \) to a point \( a \) is the projection of \( a \) onto the plane.

To make the connection between projection and minimum distance more explicit, consider the problem in a slightly different form. A plane can be defined as going through a set of vertices \( V_1, V_2, V_3 \), which create an internal coordinate system with axis vectors \( (V_2 - V_1, V_3 - V_1, N) \). This system then defines the plane in parametric form, \( T(u,v) \), where

\[
P(u,v) = V_1 + u(V_2 - V_1) + v(V_3 - V_1)
\]

(1)

The distance, \( D \), between a point \( a \) and every point on the plane is then

\[
D(u,v) = |a - P(u,v)|
\]

(2)

The closest point on the plane is the minimum of Equation 2. Minima, and really all extrema, occur at common zeros of the partial derivative of an equation. Since the distance, as expressed above, involves finding vector magnitude with a square root, a common trick is to use the squared distance instead. The squared distance shares roots with the Euclidean distance and has a simplified system of partial derivatives. Therefore, using the correspondence between the squared L-2 norm and dot product,

\[
D^2(u,v) = |a - P(u,v)|^2
\]

(3)

\[
= (a - P(u,v)) \cdot (a - P(u,v)).
\]

(4)

The minimum distance occurs at simultaneous zeros of \( F \), the system of partial derivatives of \( D^2(u,v) \). In the following equation, partials are denoted by a subscripted parameter. The partials are found by using the chain rule on \( D^2(u,v) \).

\[
F = \begin{bmatrix} D_u^2(u,v) \\ D_v^2(u,v) \end{bmatrix} = \begin{bmatrix} 2(a - P(u,v)) \cdot -P_u(u,v) \\ 2(a - P(u,v)) \cdot -P_v(u,v) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}.
\]

(5)

This probably does not seem like a very natural way to find the distance to a plane, and it would be computationally inefficient to use it directly. However, it does provide justification for the geometric projection operation used earlier. Looking at the system of partials, \( F \), it describes the conditions that need to be met where there is a minima in distance. The condition is that the vector between \( a \) and the proposed solution point on the plane, \( P(u,v) \), must be orthogonal to both the surface tangents at \( P(u,v) \). This is because the dot product between that vector and each tangent must equal zero for \( F \) to be a root. An equivalent way of stating these constraints is that the vector between \( a \) and the proposed solution point must be parallel to the normal, since the normal is orthogonal to both surface tangents. Therefore, the projected “straight-line distance” concept is equivalent to the idea of projecting along the normal to find the closest point and this is an idea that is very easily translated into algorithmic form.

1) Distance from a Point to a Plane: A plane \( P \) can be defined implicitly with a point \( P_a \) and a unit normal \( P_N \), as in

\[
P : (a - P_a) \cdot P_N = 0.
\]

All points \( a \) that satisfy this equation are part of the plane. Note that when \( a \) lies on \( P \), then the vector \( a - P_a \) is orthogonal to the normal, so the dot product is zero. This plane equation is sometimes written as

\[
P : a \cdot P_N + d = 0,
\]

where \( d = -P_a \cdot P_N \), which can be precomputed.

The distance from a point \( a \) to the plane can be found by evaluating the plane equation. Note that the plane equation forms the vector \( a - P_a \), and finds its projection onto the unit normal, thus giving its length in the normal direction. Since projecting along the normal gives the closest point on a surface, this projection is the minimum distance.
However, this is not quite correct. The point \( a \) may lie on either side of the plane, in which case the equation for distance can go negative, whereas distance should always be positive. The signed distance between two objects is useful in these cases and has other uses in computer simulation, like distinguishing between contact and penetration.

The closest point on the plane is the original point \( a \) moved along the normal by its distance to the plane. So the closest point \( cp \) is

\[
cp = a - ((a - P_a) \cdot P_N) P_N.
\]

2) Distance from a Point to a Circle: Given a two-dimensional point \( a \) and a circle \( C \) in the plane with its center \( C_c \) and radius \( C_r \). Then \( C \) is

\[
C : |a - C_c| - C_r = 0
\]

The distance between \( a \) and \( C \) is just \( a \) plugged into the equation for the circle. Again, this can produce negative distances when \( a \) is inside the circle. Note that the normal at any point on a circle is the vector from that point to the circle center, so this fulfills the projection along the normal paradigm for closest point.

3) Distance from a Point to a Line: A line \( L \) in parametric form is defined by two points \( P_0 \) and \( P_1 \)

\[
L(t) = P_0 + t(P_1 - P_0)
\]

The vector along the line \((P_1 - P_0)\) is usefully written as \( v_L \). The distance from a point \( a \) to the line \( L(t) \) can be found by projecting \( a \) onto the line, then computing the distance between the closest point projection and \( a \). So the closest point \( cp \) on the line is

\[
cp = P_0 + \left( (a - P_0) \cdot \frac{v_L}{|v_L|} \right) \frac{v_L}{|v_L|}
\]

and the distance is \( |a - cp| \).

If you just want the distance and not the closest point, there is a clever trick using the properties of cross products showing that

\[
dist(a, L) = \frac{v_L \times (a - P_0)}{|v_L|}.
\]

4) Distance from a Point to a Line Segment: A line segment is a bounded line. The logic for distance from a point to a line segment is similar to that for a line, but the closest point must lie within the bounded line segment. This can be checked by seeing if the projection of \( a \) onto \( v_L \) lies between zero and \( |v_L| \).

5) Circle-Circle Overlap: Collision tests are often easier than distance measures, but for simple primitives they are often equal in complexity. For two circles \( C_0 \) and \( C_1 \), they are separated if their centers are further apart than the sum of their radii. So,

\[
collide(C_0, C_1) = |C_{0c} - C_{1c}| \leq (C_{0r} + C_{1r}).
\]

6) Line Segment-Line Segment Intersection: When line segments are generally positioned, then they intersect when the endpoints of each are on opposite sides of the line formed by the other line segment. One way is to find the vectors between the closest points on the line and the endpoints and to see if the vectors point in the same or different directions. Another approach is to use the cross product between the line vector and a vector from a line point to an endpoint, which either points out or into the plane depending on the direction. Care must be taken when dealing with degenerate cases where an endpoint lies on the other line.