Mobile Robotics - Homework 1

Kevin Robb

October 1, 2021

1

We are given four points as expressed in the object's frame of reference O, and those same points' representation in the camera's frame of reference S.

We can assume our basis for both frames is orthonormal and right-handed. Since we have preservation of distances and orientation, we can say that our transformation will be a member of the Special Euclidean group, and will be parametrized by a rotation matrix and a translation distance; finding these values will give us precisely one frames' position and orientation relative to the other.

$$SE(3) = \left\{ \begin{pmatrix} R & b \\ 0 & 1 \end{pmatrix} \middle| R \in SO(3), b \in \mathbb{R}^3 \right\} \tag{1}$$

where the group of rotations is

$$SO(3) = \left\{ R \in \mathbb{R}^{3\times 3} \;\middle|\; R^T R = I, \det(R) = 1 \right\} \tag{2}$$

This motivates our ability to use this structure to represent the pose of our object w.r.t. the camera as either an affine matrix operation (Eq. 3) or a more standard equation that keeps the rotation and translation as separate variables (Eq. 4).

$$\begin{pmatrix} \boldsymbol{v}_s \\ 1 \end{pmatrix} = \begin{pmatrix} R_{SO} & \boldsymbol{t}_{SO} \\ 0 & 1 \end{pmatrix} \begin{pmatrix} \boldsymbol{v}_o \\ 1 \end{pmatrix} \tag{3}$$

$$T_{SO}(\boldsymbol{v}_o) = R_{SO}\boldsymbol{v}_o + t_{SO} = \boldsymbol{v}_s \tag{4}$$

In order to find the orientation and position of the object's origin relative to the camera's frame, we can use either equation and the four provided pairs of points to create a system of equations. We show Eq. (4), but both will give the same system.

$$\begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \cdot \begin{bmatrix} v_{o1} \\ v_{o2} \\ v_{o3} \end{bmatrix} + \begin{bmatrix} t_1 \\ t_2 \\ t_3 \end{bmatrix} = \begin{bmatrix} v_{s1} \\ v_{s2} \\ v_{s3} \end{bmatrix} ,$$

which can be expanded to

$$\begin{cases} r_{11}v_{o1} + r_{12}vo2 + r_{13}v_{o3} + t_1 = v_{s1} \\ r_{21}v_{o1} + r_{22}vo2 + r_{23}v_{o3} + t_2 = v_{s2} \\ r_{31}v_{o1} + r_{32}vo2 + r_{33}v_{o3} + t_3 = v_{s3} \end{cases}$$

This set of three equations, used with each of our four sets of points, gives us twelve equations and twelve unknowns. We can solve for all unknowns by creating a 12×12 matrix (augmented with a 13th column) where each row represents one equation, each column corresponds to a certain variable, and each entry is the coefficient for that variable. Each group separated by a horizontal line is created with one of the four sets of points.

$$\left[\begin{array}{cccccccccccccc}
2 & 3 & -3 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & -1.3840 \\
0 & 0 & 0 & 2 & 3 & -3 & 0 & 0 & 0 & 0 & 1 & 0 & 4.5620 \\
0 & 0 & 0 & 0 & 0 & 0 & 2 & 3 & -3 & 0 & 0 & 1 & -0.1280 \\
\hline
0 & 0 & -3 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & -0.9608 \\
0 & 0 & 0 & 0 & 0 & -3 & 0 & 0 & 0 & 0 & 1 & 0 & 1.3110 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -3 & 0 & 0 & 1 & -1.6280 \\
\hline
-1 & -2 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1.3250 \\
0 & 0 & 0 & -1 & -2 & 2 & 0 & 0 & 0 & 0 & 1 & 0 & -2.3890 \\
0 & 0 & 0 & 0 & 0 & 0 & -1 & -2 & 2 & 0 & 0 & 1 & 1.7020 \\
\hline
-1 & 0 & -2 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & -1.3140 \\
0 & 0 & 0 & -1 & 0 & -2 & 0 & 0 & 0 & 0 & 1 & 0 & 0.2501 \\
0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & -2 & 0 & 0 & 1 & -0.7620
\end{array}\right]$$

We use this matrix to arrive at solutions for all variables by transforming it into reduced row echelon form (rref). This can be done with Gaussian Elimination by hand, but here I use the *rref()* function on my TI-84 graphing calculator. This would not be too difficult by hand since all relevant entries are small integers, but it would be long and tedious. The result is shown below.

$$\Longrightarrow \begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.70681 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -0.6123 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.35361 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.70722 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.61219 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & -0.3537 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0.5 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0.866 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0.10004 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0.24996 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0.97
\end{bmatrix}$$

Thus we have a solution to our unknowns and can write the pose translation.

$$T_{SO}(\boldsymbol{v}_o) = \begin{bmatrix} 0.70681 & -0.6123 & 0.35361 \\ 0.70722 & 0.61219 & -0.3537 \\ 0 & 0.5 & 0.866 \end{bmatrix} \boldsymbol{v}_o + \begin{bmatrix} 0.1 \\ 0.25 \\ 0.97 \end{bmatrix} = \boldsymbol{v}_s$$

Note:

Instead of creating this large matrix, we can use our four coordinates to create a 4×4 matrix for each frame, and write

$$\begin{pmatrix} \boldsymbol{v}_{s1} & \boldsymbol{v}_{s2} & \boldsymbol{v}_{s3} & \boldsymbol{v}_{s4} \\ 1 & 1 & 1 & 1 \end{pmatrix} = \begin{pmatrix} R_{SO} & \boldsymbol{t}_{SO} \\ 0 & 1 \end{pmatrix} \begin{pmatrix} \boldsymbol{v}_{o1} & \boldsymbol{v}_{o2} & \boldsymbol{v}_{o3} & \boldsymbol{v}_{o4} \\ 1 & 1 & 1 & 1 \end{pmatrix},$$

which we can solve by taking the inverse of the rightmost matrix and post-multiplying it.

$$\begin{pmatrix} \boldsymbol{v}_{s1} & \boldsymbol{v}_{s2} & \boldsymbol{v}_{s3} & \boldsymbol{v}_{s4} \\ 1 & 1 & 1 & 1 \end{pmatrix} \cdot \begin{pmatrix} \boldsymbol{v}_{o1} & \boldsymbol{v}_{o2} & \boldsymbol{v}_{o3} & \boldsymbol{v}_{o4} \\ 1 & 1 & 1 & 1 \end{pmatrix}^{-1} = \begin{pmatrix} R_{SO} & \boldsymbol{t}_{SO} \\ 0 & 1 \end{pmatrix}$$

This is certainly more elegant, but I used Matlab to verify that this gives the same result as the longer method, so I haven't rewritten my solution to the problem to use this method.

2

We have a Lie group G. $\forall g \in G$, the *left-translation map* $L_g : G \to G$ is defined as $L_g(x) = gx$. This map, determined by g, is a diffeomorphism of G. The operation combining g and x is determined by the group operation of G.

We know this map can be used to *identify* the Lie algebra Lie(G) with the set of *left-invariant vector fields on G*:

$$\begin{aligned} \phi : \text{Lie}(G) &\to \{ \text{ left-invariant vector fields on } G \, \} \\ \phi(\omega) &= V_\omega \ , \end{aligned} \tag{5}$$

where the left-invariant vector field on G is

$$V_\omega(x) = d(L_x)_e(\omega) \tag{6}$$

To each element $\omega \in \text{Lie}(G)$ in the Lie algebra, we associate the left-invariant vector field V_ω, whose value is determined by Eq. (6).

$V_\omega(x)$ at $x \in G$ is the image of ω under the derivative of the left-translation map L_x that sends the identity $e \in G$ to x.

This exercise will consider the two Lie groups $\mathbb{R}^n$ and $GL(n)$. The group operation of $\mathbb{R}^n$ is addition, and the group operation of $GL(n)$ is matrix multiplication.

2.a

Given $v \in \mathbb{R}^n$, the corresponding left-translation map $L_v : \mathbb{R}^n \to \mathbb{R}^n$ is

$$L_v(x) = v + x$$

because the group operation of $\mathbb{R}^n$ is addition.

2.b

The derivative of this map is the standard derivative in $\mathbb{R}^n$,

$$dL_v(x) = \frac{d}{dx}v + \frac{d}{dx}x = 1$$

because $dL_v(x)$ is a single-variable function in Euclidean space.

2.c

Given a vector $\xi \in \mathrm{Lie}(G) \cong \mathbb{R}^n$ in $\mathbb{R}^n$'s Lie algebra, the left-invariant vector field is simply the identity matrix

$$V_\xi(x) = d(L_x)_e(\xi) = I_n \cdot \xi$$

This is a constant vector field. The vector associated with any point is the same as that at the identity, so the point at which we evaluate $dL_v(x)$ does not matter:

$$\forall x \in G, dL_v(x) = I \cdot x = x$$

2.d

Given a matrix $A \in GL(n)$, the corresponding left translation map is $L_A : GL(n) \to GL(n)$ is

$$L_A(X) = A \cdot X$$

where $X \in \mathbb{R}^{n\times n}$, because the group operation of $GL(n)$ is matrix multiplication.

2.e

The derivative of this map is

$$dL_A = A \cdot \dot{X}$$

We use the general expression $\dot{X}$ to mean the derivative of X.

2.f

The Lie algebra of $GL(n)$ is $\mathrm{Lie}(GL(n)) = \mathbb{R}^{n\times n}$, the set of all $n \times n$ matrices. Now given a matrix $\Omega \in \mathrm{Lie}(GL(n))$, the left-invariant vector field V_Ω on $GL(n)$ determined by Ω is

$$V_\Omega = A \cdot \Omega$$

since A is a constant matrix.

3

We are given the exponential map for the general linear group:

$$\begin{aligned} \exp : \mathbb{R}^{n\times n} &\to GL(n) \\ \exp(X) &= \sum_{k=0}^{\infty} \frac{X^k}{k!} \end{aligned} \tag{7}$$

We can see that this series looks like the form of the Taylor series of e^x centered at 0.

We've mentioned that this formula can be significantly simplified when applied to a *subgroup* $G \subseteq GL(n)$, which we will demonstrate for the orthogonal group $O(2)$.

3.a

We know Lie($O(n)$) is Skew(n), the set of n-dimensional skew-symmetric matrices defined as

$$\text{Skew}(n) = \left\{ A \in \mathbb{R}^{n\times n} \mid A^T = -A \right\}. \tag{8}$$

For $O(2)$ in particular,

$$\text{Lie}(O(2)) = \text{Skew}(2) = \left\{ \begin{pmatrix} 0 & \omega \\ -\omega & 0 \end{pmatrix} \in \mathbb{R}^{2\times 2} \;\middle|\; \omega \in \mathbb{R} \right\}. \tag{9}$$

We will take some element of this set, $\Omega = \begin{pmatrix} 0 & \omega \\ -\omega & 0 \end{pmatrix}$, and find an expression for Ω^k.

By taking increasing powers of Ω, we see that all even powers are symmetric matrices, while all odd powers are skew-symmetric matrices. We can also see a pattern in the signs which we can parametrize.

$$\Omega^k = \begin{cases} \begin{pmatrix} (-1)^{k/2}\omega^k & 0 \\ 0 & (-1)^{k/2}\omega^k \end{pmatrix} & \text{for even } k \\ \begin{pmatrix} 0 & (-1)^{(k-1)/2}\omega^k \\ -(-1)^{(k-1)/2}\omega^k & 0 \end{pmatrix} & \text{for odd } k \end{cases}$$

When examining the signs, we see what may be the Taylor expansions for $\cos x$ and $\sin x$ emerge. To help us in future steps, we can rewrite our equation for Ω^k in terms of i now and pull common factors out of the matrices.

$$\Omega^k = \begin{cases} \begin{pmatrix} i^k\omega^k & 0 \\ 0 & i^k\omega^k \end{pmatrix} & \text{for even } k \\ \begin{pmatrix} 0 & i^{(k-1)}\omega^k \\ -i^{(k-1)}\omega^k & 0 \end{pmatrix} & \text{for odd } k \end{cases} = \begin{cases} i^k\omega^k \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} & \text{for even } k \\ i^{(k-1)}\omega^k \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix} & \text{for odd } k \end{cases}$$

3.b

We can now fill in our expression for the exp function:

$$\exp(\Omega) = \sum_{k=0,\text{even}}^{\infty} \frac{1}{k!} i^k \omega^k \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} + \sum_{k=1,\text{odd}}^{\infty} \frac{1}{k!} i^{(k-1)} \omega^k \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}$$

We can write the Taylor series representations of $\sin x$ and $\cos x$ for comparison.

$$\sin x = \sum_{k=1,\text{odd}}^{\infty} i^{(k-1)} \cdot \frac{x^k}{k!}$$

$$\cos x = \sum_{k=0,\text{even}}^{\infty} i^{k} \cdot \frac{x^k}{k!}$$

These are clearly represented in our equation for $\exp(\Omega)$:

$$\exp(\Omega) = \cos(\omega)\begin{pmatrix}1 & 0\\ 0 & 1\end{pmatrix} + \sin(\omega)\begin{pmatrix}0 & 1\\ -1 & 0\end{pmatrix}$$

This simplified expression has a clear analog to Euler's formula, which defines the exponential in terms of sin and cos.

$$e^{i\theta} = \cos\theta + i\sin\theta \tag{10}$$

We can think of the matrix multiplying the sin term as an analog for i, while the cos term is multiplied by the identity.

The geometric interpretation of this is a rotation specified by Ω. Considering Ω as a linear map in $\mathbb{R}^2$ exactly matches the matrix for a clockwise rotation by $\frac{\pi}{2}$ and a scaling by ω.

4

The exponential map describes the *integral curves/trajectories* of a certain vector field. A specific left-invariant vector field V_ω that starts at the point $x \in G$ at time $t = 0$ has the integral curve $\gamma : \mathbb{R} \to G$ defined as

$$\gamma(t) = x\exp(t\omega). \tag{11}$$

This provides an intuition for "moving around" on the Lie group G along the "direction" determined by ω.

We will be using this equation to *interpolate* Lie group-valued data.

4.a

For a point $x \in G$ in the image of G's exponential map, we denote one of x's preimages as $\log(x)$. If G's exponential map is surjective, we know there is at least one choice of $\log(x) \in \text{Lie}(G)$ that satisfies

$$x = \exp(\log(x)) \tag{12}$$

Now for $x, y \in G$, where $\exp(G)$ is surjective, we can derive a formula for $\gamma : [0,1] \to G$ s.t. $\gamma(0) = x$ and $\gamma(1) = y$.

Since we have surjectivity, we can use the logarithm on Eq (11) to obtain a definition for ω.

$$\begin{aligned} x^{-1}\cdot\gamma(t)) &= x^{-1}\cdot x\cdot\exp(t\omega) \\ \log\left(x^{-1}\cdot\gamma(t)\right) &= \log(\exp(t\omega)) \\ \log\left(x^{-1}\cdot\gamma(t)\right) &= t\omega \end{aligned}$$

Applying the boundary conditions gives two equations.

$$\begin{aligned} &\Longrightarrow \begin{cases} \log\left(x^{-1}\cdot y\right) = \omega \\ \log\left(x^{-1}\cdot x\right) = 0 \end{cases} \\ &\Longrightarrow \omega = \log(x^{-1}\cdot y) \end{aligned}$$

We can then substitute this in for our definition of γ to see

$$\gamma(t) = x\cdot\exp\left(t\cdot\log(x^{-1}\cdot y)\right).$$

Without knowing the particular group we're acting on, there is no more simplification possible.

4.b

The exponential map for $\mathbb{R}^n$ is just the identity map, $\exp : \mathbb{R}^n \to \mathbb{R}^n$, where $\exp(\xi) = \xi$. Thus the logarithm will also be the identity, so $\log(\xi) = \xi$.

We can use this fact to specialize our result from the previous part to derive a formula for a curve γ that joins x to y in $\mathbb{R}^n$. We know that the group operation in $\mathbb{R}^n$ is addition, and the inverse is simply $x^{-1} = -x$. These properties allow us to simplify our expression from part (a).

$$\begin{aligned}
\gamma(t) &= x \cdot \exp(t \cdot \log(x^{-1} \cdot y)) \\
\implies \gamma(t) &= x + \exp(t \cdot \log(-x + y)) \\
&= x + \exp(t \cdot (-x + y)) \\
&= x + t \cdot (-x + y) \\
&= (1 - t) \cdot x + t \cdot y
\end{aligned}$$

It's pretty clear to see this function creates a straight line in Euclidean space connecting the two points $x, y \in \mathbb{R}^n$. Our boundary conditions are satisfied, since $\gamma(0) = x$ and $\gamma(1) = y$.

4.c

The Lie group $SE(3)$ of 3D robot poses can be modeled as the product manifold $M = \mathbb{R}^3 \times SO(3)$. This is simply the set of pairs $(\boldsymbol{t}, R)$, where $\boldsymbol{t}$ is a translation vector and R is a rotation matrix. These together give the robot's pose. This group has the following multiplication rule:

$$(\boldsymbol{t_1}, R_1) \cdot (\boldsymbol{t_2}, R_2) = (R_1\boldsymbol{t_2} + \boldsymbol{t_1}, R_1R_2) \tag{13}$$

Given the two poses X_0 and X_1,

$$\begin{aligned}
X_0 &= \left(\begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 0.4330 & 0.1768 & 0.8839 \\ 0.2500 & 0.9186 & -0.3062 \\ -0.8660 & 0.3536 & 0.3536 \end{pmatrix} \right) \\
X_1 &= \left(\begin{pmatrix} 2 \\ 4 \\ 3 \end{pmatrix}, \begin{pmatrix} 0.7500 & -0.0474 & 0.6597 \\ 0.4330 & 0.7891 & -0.4356 \\ -0.5000 & 0.6124 & 0.6124 \end{pmatrix} \right),
\end{aligned}$$

we can write each as a single matrix which will be an element of $GL(n + 1)$.

$$\begin{aligned}
X_0 &= \begin{pmatrix} R_0 & \boldsymbol{t_0} \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} 0.4330 & 0.1768 & 0.8839 & 1 \\ 0.2500 & 0.9186 & -0.3062 & 1 \\ -0.8660 & 0.3536 & 0.3536 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \\
X_1 &= \begin{pmatrix} R_1 & \boldsymbol{t_1} \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} 0.7500 & -0.0474 & 0.6597 & 2 \\ 0.4330 & 0.7891 & -0.4356 & 4 \\ -0.5000 & 0.6124 & 0.6124 & 3 \\ 0 & 0 & 0 & 1 \end{pmatrix}
\end{aligned}$$

These matrix representations embed the multiplication rule from Eq. (13) and allow us to simply use matrix multiplication as our operation. This is easily demonstrated:

$$\begin{pmatrix} R_0 & \boldsymbol{t_0} \\ 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} R_1 & \boldsymbol{t_1} \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} R_0R_1 & R_0\boldsymbol{t_1} + \boldsymbol{t_0} \\ 0 & 1 \end{pmatrix}$$

We can now use our formula from part (a) to find the midpoint (where $t = \frac{1}{2}$) on the curve $\gamma_{SE(3)}$.

$$\gamma_{SE(3)}(1/2) = X_0 \cdot \exp\left(\frac{1}{2} \cdot \log(X_0^{-1} \cdot X_1)\right)$$

We solve this using the regular rules for matrix inversion and multiplication, since we are in $GL(n)$, and we evaluate the logarithm and exponential using the appropriate matrix functions in Matlab.

```
X0 = [0.4330, 0.1768, 0.8839, 1;
        0.2500, 0.9186, -0.3062, 1;
        -0.8660, 0.3536, 0.3536, 0;
        0, 0, 0, 1]
X1 = [0.7500, -0.0474, 0.6597, 2;
        0.4330, 0.7891, -0.4356, 4;
        -0.5000, 0.6124, 0.6124, 3;
        0, 0, 0, 1]
X_mid = X0 * expm(0.5 * logm(inv(X0) * X1))
```

This script gives the resulting midpoint pose as the matrix

$$X_{1/2} = \begin{pmatrix} 0.6124 & 0.0795 & 0.7866 & 1.6710 \\ 0.3535 & 0.8624 & -0.3624 & 2.5988 \\ -0.7071 & 0.5000 & 0.5000 & 1.3443 \\ 0 & 0 & 0 & 1.0000 \end{pmatrix}$$

which has the associated pose

$$\gamma_{SE(3)}(1/2) = \left(\begin{pmatrix} 1.6710 \\ 2.5988 \\ 1.3443 \end{pmatrix}, \begin{pmatrix} 0.6124 & 0.0795 & 0.7866 \\ 0.3535 & 0.8624 & -0.3624 \\ -0.7071 & 0.5000 & 0.5000 \end{pmatrix} \right)$$

4.d

Using the Lie groups $\mathbb{R}^3$ under vector addition and $SO(3)$ under matrix multiplication, we can construct the *product* Lie group $P = \mathbb{R}^3 \times SO(3)$. Elements of this group are similarly of the form $(\boldsymbol{t}, R)$, and have the multiplication law

$$(\boldsymbol{t_1}, R_1) \cdot (\boldsymbol{t_2}, R_2) = (\boldsymbol{t_1} + \boldsymbol{t_2}, R_1 R_2) \tag{14}$$

Thus $SE(3)$ and P have the same *manifold* structure but different *group* structures.

We can use our formula from part (a) to compute the midpoint of the same points from part (c) in P rather than $SE(3)$. Here $\gamma_P : [0, 1] \to P$.

With this multiplication rule, we can see that each element of the pose is affected separately via their respective group operation; the translation components are vectors in $\mathbb{R}^3$ and the rotation components are matrices in $GL(3)$. Thus we can perform every operation on the pose separately to each component.

$$\begin{aligned}
\gamma_P(1/2) &= X_0 \cdot \exp\left(\frac{1}{2} \cdot \log(X_0^{-1} \cdot X_1)\right) \\
&= (\boldsymbol{t_0}, R_0) \cdot \exp\left(\frac{1}{2} \cdot \log((\boldsymbol{t_0}, R_0)^{-1} \cdot (\boldsymbol{t_1}, R_1))\right) \\
&= (\boldsymbol{t_0}, R_0) \cdot \exp\left(\frac{1}{2} \cdot \log((-\boldsymbol{t_0}, R_0^{-1}) \cdot (\boldsymbol{t_1}, R_1))\right) \\
&= (\boldsymbol{t_0}, R_0) \cdot \exp\left(\frac{1}{2} \cdot \log((-\boldsymbol{t_0} + \boldsymbol{t_1}, R_0^{-1} \cdot R_1))\right) \\
&= (\boldsymbol{t_0}, R_0) \cdot \exp\left(\frac{1}{2} \cdot (-\boldsymbol{t_0} + \boldsymbol{t_1}, \log(R_0^{-1} \cdot R_1))\right) \\
&= (\boldsymbol{t_0}, R_0) \cdot \exp\left(\frac{1}{2}(-\boldsymbol{t_0} + \boldsymbol{t_1}), \frac{1}{2}\log(R_0^{-1} \cdot R_1)\right) \\
&= (\boldsymbol{t_0}, R_0) \cdot \left(\frac{1}{2}(-\boldsymbol{t_0} + \boldsymbol{t_1}), \exp\left(\frac{1}{2}\log(R_0^{-1} \cdot R_1)\right)\right) \\
&= \left(\boldsymbol{t_0} + \frac{1}{2}(-\boldsymbol{t_0} + \boldsymbol{t_1}), R_0 \cdot \exp\left(\frac{1}{2}\log(R_0^{-1} \cdot R_1)\right)\right) \\
&= \left(\frac{1}{2}(\boldsymbol{t_0} + \boldsymbol{t_1}), R_0 \cdot \exp\left(\frac{1}{2}\log(R_0^{-1} \cdot R_1)\right)\right)
\end{aligned}$$

The translation component is easy to compute, but the rotation component will require more work. We use Matlab to perform the matrix exponential and logarithm functions.

```matlab
% instantiate poses (T0,R0) and (T1,R1).
T0 = [1; 1; 0]
R0 = [0.4330, 0.1768, 0.8839; 0.2500, 0.9186, -0.3062; -0.8660, 0.3536, 0.3536]
T1 = [2; 4; 3]
R1 = [0.7500, -0.0474, 0.6597; 0.4330, 0.7891, -0.4356; -0.5000, 0.6124, 0.6124]
% calculate the components of the midpoint's pose.
t_mid = 0.5 * (T0 + T1)
R_mid = R0 * expm(0.5 * logm(inv(R0) * R1))
```

$$\gamma_P(1/2) = \left(\begin{pmatrix} 1.5 \\ 2.5 \\ 1.5 \end{pmatrix}, \begin{pmatrix} 0.6124 & 0.0795 & 0.7866 \\ 0.3535 & 0.8624 & -0.3624 \\ -0.7071 & 0.5000 & 0.5000 \end{pmatrix}\right)$$

4.e

We compare the translational components of the curves $\gamma_{SE(3)}$ and γ_P from parts (c) and (d) over two intervals, $t \in [0, 1]$ and $t \in [0, 30]$, using Matlab.

```
% range for time. uncomment only one of them.
times = 0:0.01:1
%times = 0:0.1:30

% instantiate arrays for component values.
X_se3 = []; Y_se3 = []; Z_se3 = []
X_p = []; Y_p = []; Z_p = []

% loop through values of t and plot points (x,y,z)=t_se3.
for t = 1:length(times)
    % get translation vector for gamma_se(3).
    G_se3 = X0 * expm(times(t) * logm(inv(X0) * X1))
    % extract the 4th column, and remove the 4th row.
    t_se3 = G_se3(1:3, 4);
    % assign component values to appropriate lists.
    X_se3(end+1) = t_se3(1,1)
    Y_se3(end+1) = t_se3(2,1)
    Z_se3(end+1) = t_se3(3,1)

    % get translation vector for gamma_p.
    t_p = ((1 - times(t)) * T0) + (times(t) * T1)
    % assign component values to appropriate lists.
    X_p(end+1) = t_p(1,1)
    Y_p(end+1) = t_p(2,1)
    Z_p(end+1) = t_p(3,1)
end

plot3(X_se3,Y_se3,Z_se3,'-.r',X_p,Y_p,Z_p,'-.b')
```

The plots produced by this script are shown in Figures 1 and 2. We can see that the map γ_P (shown in blue on the plots), with its standard multiplication rule, gives a simple straight line connecting the two points. When this line is extrapolated out to $t = 30$, it just continues on its linear path. The map $\gamma_{SE(3)}$ (shown in red), with its translation twisted in its multiplication rule, follows a curved path connecting the two points. When this path is extrapolated, we see it follows a spiral/corkscrew pattern.

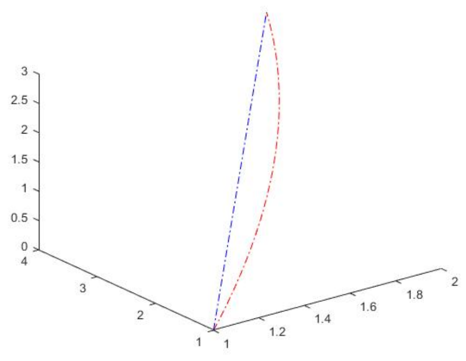

Figure 1: Both paths interpolated in the range $t \in [0, 1]$.

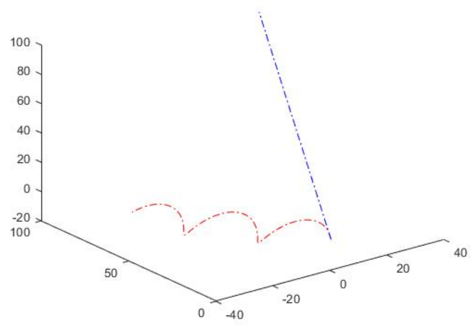

Figure 2: Both paths extrapolated to the range $t \in [0, 30]$.

5

This exercise involves estimating the position of a mobile robot from noisy sensor data using probability theory. We know that Bayes' theorem is

$$p(A|B) = \frac{p(B|A) \cdot p(A)}{p(B)} \tag{15}$$

and a conditional probability is defined as

$$p(A|B) = \frac{p(A \cap B)}{p(B)}. \tag{16}$$

The conditional form of Bayes' rule is

$$p(X|Y, Z) = \frac{p(Y|X, Z) \cdot p(X|Z)}{p(Y|Z)}. \tag{17}$$

We can also use integrals to marginalize a joint distribution with the rule

$$p(x) = \int p(x|y) \cdot p(y) \cdot \mathrm{d}\, y. \tag{18}$$

5.a

The initial estimate of the robot's position $X_0 \in \mathbb{R}^2$ is expressed as a probability density function $p(X_0)$. It drives its motors with some known motor commands u. We have access to a *motion model* describing how its subsequent position $X_1 \in \mathbb{R}^2$ depends upon X_0 and u; this is expressed as a conditional probability density $p(X_1|X_0, u)$.

We can derive $p(X_1|u)$, a formula for the conditional probability density of the robot's next position given the motor commands. We will use the fact that the initial position is independent of the motor commands, so $p(X_0|u) = p(X_0)$.

We can rearrange Eq. (16) to see

$$p(A, B) = p(A|B) \cdot p(B) = p(B|A) \cdot p(A). \tag{19}$$

From this rule, we specialize for our case:

$$p(X_1, X_0|u) = p(X_1|X_0, u) \cdot p(X_0|u) = p(X_1|X_0, u) \cdot p(X_0)$$

where we have the motion model $p(X_1|X_0, u)$ and the prior $p(X_0)$, so we have an expression for the joint distribution of x_0 and x_1.

We can then marginalize this using Eq. (18) to throw out x_0 in the pairs (x_0, x_1) and obtain the desired expression for $p(x_1|u)$,

$$p(X_1|u) = \int p(X_1|X_0, u) \cdot p(X_0) \cdot \mathrm{d}\, X_0.$$

5.b

Without reinforcement, the uncertainty will only increase the longer our model runs. We counteract this uncertainty by taking sensor measurements Y. We have a *sensor model*, $p(Y|X)$, that describes how a measurement Y depends on the true position X.

Given $p(X_1|u)$, our *prior* belief about the robot's position after applying the motor commands, and the sensor model $p(Y_1|X_1)$, we can derive a formula for the *posterior* distribution $p(X_1|Y_1, u)$. We know both the measurement Y_1 and the motor commands u.

$$p(X_1|Y_1, u) = \frac{p(Y_1|X_1, u) \cdot p(X_1|u)}{p(Y_1|u)}$$

where we know the sensor model $p(Y_1|X_1)$ and the prior $p(X_1|u)$ (derived in the previous part). We know that $p(Y_1|u)$ can be represented as an integral containing terms that we do know.

$$\begin{aligned} p(Y_1|u) &= \int p(Y_1|X_1, u) \cdot p(X_1|u) \cdot \mathrm{d}\, X_1 \\ &= \int p(Y_1|X_1) \cdot p(X_1|u) \cdot \mathrm{d}\, X_1 \end{aligned}$$