Problem 1: 2D Transformations (15 pts)

(i) Estimate the 2D affine transformation matrix, \( T = \begin{bmatrix} F & v \\ 0^T & 1 \end{bmatrix} \in \mathbb{R}^{3 \times 3} \), given its action on three homogeneous points:

\[
\begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \xrightarrow{T} \begin{pmatrix} 2 \\ 1 \\ 1 \end{pmatrix}, \quad \begin{pmatrix} 1 \\ -1 \\ 1 \end{pmatrix} \xrightarrow{T} \begin{pmatrix} 2 \\ 1 \\ 1 \end{pmatrix}, \quad \begin{pmatrix} -1 \\ -1 \\ 1 \end{pmatrix} \xrightarrow{T} \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix}.
\]

• Answer: You need to estimate \( F \) and \( v \) from its observed transformation of some 2D points,

\[
\begin{bmatrix} F & v \\ 0^T & 1 \end{bmatrix} \begin{pmatrix} p_k \\ 1 \end{pmatrix} = \begin{pmatrix} p'_k \\ 1 \end{pmatrix} \iff Fp_k + v = p'_k,
\]

where we will denote

\[
\begin{align*}
p_1 &= \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \rightarrow p'_1 = \begin{pmatrix} 2 \\ 1 \\ 1 \end{pmatrix}, & (1) \\
p_2 &= \begin{pmatrix} 1 \\ -1 \\ 1 \end{pmatrix} \rightarrow p'_2 = \begin{pmatrix} 2 \\ 1 \\ 1 \end{pmatrix}, & (2) \\
p_3 &= \begin{pmatrix} -1 \\ -1 \\ 1 \end{pmatrix} \rightarrow p'_3 = \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix}. & (3)
\end{align*}
\]

There are numerous ways to solve for \( F \) and \( v \), with the least easy being setting it up as a large 6x6 system of equations. Another way is to solve for \( F \) first (then compute the translation \( v \)) by subtracting one equation from the other two

\[
F(p_k - p_j) + v = (p'_k - p'_j),
\]

to obtain the matrix form,

\[
F \begin{bmatrix} (p_2 - p_1) & (p_3 - p_1) \end{bmatrix} = \begin{bmatrix} (p'_2 - p'_1) & (p'_3 - p'_1) \end{bmatrix} \iff FP = P',
\]

and then solve the 2x2 linear system for \( F \),

\[
F = P'P^{-1}.
\]

Finally you get the translation from any equation

\[
v = p'_k - Fp_k.
\]

The simplest way is to observe that the data makes this problem easy: the first two points (which differ by the \( y \) coordinate) map to the same place, \( p'_1 = p'_2 \), so the transformation \( F' \) must have a zero “\( y \) column.” For
example, in a column-oriented form you can let \( F = [t \ u] \) where \( t, u \in \mathbb{R}^2 \), and let \( p_k = (x_k, y_k)^T \), so that we need to solve for the 3 vectors \( t, u \) and \( v \):

\[
x_k t + y_k u + v = p_k', \quad k = 1, 2, 3.
\]

This leads to the system of equations

\[
\begin{align*}
t + u + v &= \begin{pmatrix} 2 \\ 1 \end{pmatrix} \\
t - u + v &= \begin{pmatrix} 2 \\ 1 \end{pmatrix} \\
-t - u + v &= \begin{pmatrix} 0 \\ 1 \end{pmatrix}.
\end{align*}
\]

Subtracting the first two equations yields

\[
u = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix},
\]

after which adding the last two equations yields

\[
v = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix},
\]

and so

\[
t = \begin{pmatrix} 1 \\ 0 \end{pmatrix},
\]

and the transformation matrix is

\[
T = \begin{bmatrix} F & v \\ 0^T & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix}
\]

(ii) What kind of transformation does this matrix represent?

- **Answer:** It applies a nonuniform scale of \((s_x, s_y) = (1, 0)\) (which projects away all \( y \) components) then translates by \((1, 1)\).

**Problem 2: Affine Transformations (10 pts)**

Show that affine transformations preserve parallel lines.

*(Hint: Recall the explicit parameterization of a line.)*

- **Answer:** Consider two parallel lines whose points can be explicitly represented as

\[
p_k + tu, \quad k = 1, 2,
\]

where each is parameterized by some \( t \in \mathbb{R} \), and both point in the same \( u \) direction (by virtue of being parallel) but have different base points \( p_k \). Applying an affine transformation \( \begin{bmatrix} F & v \\ 0^T & 1 \end{bmatrix} \) to such points produces new line points

\[
F(p_k + tu) + v, = (Fp_k + v) + tFu, = p_k' + tu'.
\]

However, since both transformed lines point in the same direction \( u' = Fu \), they are still parallel.
Problem 3: Quaternions (15 pts)

Rotate the point \( p = (1, 1, 1) \) using the rotation specified by the quaternion \( q = \langle d; u \rangle = (1; 1, 1, 1) \).

- **Answer:** In short, the point is unchanged since it lies on the rotation axis. You can show this by using the definition of a unit quaternion. First, you need to make a unit quaternion and apply the definition

\[
\hat{q} = \frac{q}{\|q\|} = \frac{q}{\sqrt{4}} = \langle \frac{1}{2}; \frac{1}{2}, \frac{1}{2}, \frac{1}{2} \rangle = \langle \cos \frac{\theta}{2}; \sin \frac{\theta}{2} v \rangle
\]

so that the unit axis of rotation, \( v \), is

\[
\sin \frac{\theta}{2} v = \frac{1}{2}(1, 1, 1) \implies v = \frac{1}{\sqrt{3}}(1, 1, 1).
\]

Clearly since \( p = (1, 1, 1) \) lies on the rotation axis, the rotated version will not change under rotation, \( p' = p \). Arguing thus was sufficient to receive full credit.

Alternately many people proceeded directly to apply the quaternion multiplication formulas, which would also work. We can represent the point using a quaternion vector, which before and after rotation is

\[
\tilde{p} = \langle 0; p \rangle, \quad \text{and} \quad \tilde{p}' = \langle 0; p' \rangle.
\]

Then the quaternion rotation formula is (using the unit quaternion version for convenience)

\[
\tilde{p}' = \hat{q} \tilde{p} \hat{q}^* = \left( \frac{1}{2} < 1; 1, 1, 1 > \right) < 0; p > \left( \frac{1}{2} < 1; -1, -1, -1 > \right) \quad (7)
\]

\[
= \frac{1}{4} < 1; 1, 1, 1 > < 0; 1, 1, 1 > < 1; -1, -1, -1 > \quad (8)
\]

\[
= \frac{1}{4} < 1; 1, 1, 1 > < 0; 1, 1, 1 > < 1; -1, -1, -1 > \quad (9)
\]

then using the formula for quaternion multiplication

\[
<d; u> <d'; u'> = <dd' - u \cdot u'; du' + d'u + u \times u'>
\]

we obtain

\[
\tilde{p}' = \frac{1}{4} < 1; 1, 1, 1 > < 0; 1, 1, 1 > < 1; -1, -1, -1 > \quad (10)
\]

\[
= \frac{1}{4} < 1; -3, 1, 1 > < 1; -1, -1, -1 > \quad (11)
\]

\[
= \frac{1}{4} < 0; 4, 4, 4 > \quad (12)
\]

\[
= < 0; 1, 1, 1 > \quad (13)
\]

\[
= < 0; p > \quad (14)
\]

Problem 4: SLERP (10 pts)

When interpolating with SLERP between two unit quaternions, \( x \) and \( y \), we use:

\[
\text{SLERP}(x, y, \alpha), \text{if } x \cdot y > 0, \text{and } \text{SLERP}(x, -y, \alpha) \text{ otherwise.}
\]

(i) Why is this method better than just \( \text{SLERP}(x, y, \alpha) \)? What is the difference between \(+y\) and \(-y\) here?
• **Answer:** Recall that scaling a quaternion by a nonzero constant does not change the rotation it represents, and therefore $\pm y$ both represent the same physical rotation. The only difference is the size of the interpolation performed, which is related to the angle $\Omega$ between $x$ and $y$, given by $\Omega = \cos^{-1}(x \cdot y)$. The two interpolation paths differ by which way they travel around the great circle, and the formula just takes the shorter path—as many students illustrated with a figure.

(ii) When interpolating unit normal vectors, $n_1$ and $n_2$, for lighting calculations, should we also use $\text{Slerp}(n_1, n_2, \alpha)$, if $n_1 \cdot n_2 > 0$, and $\text{Slerp}(n_1, -n_2, \alpha)$ otherwise?

• **Answer:** No, because, unlike quaternions for which $\pm q$ correspond to the same physical rotations, $\pm n$ correspond to different normals for shading/lighting calculations. Using this formula would lead to interpolated normals which vary insufficiently when the two normals are at angles greater than $90^\circ$. 

4