Coordinate Transforms in Robotics

Rotation Matrices and Properties

1. Calculate the matrices representing +90°, +180°, and −90° 2D rotations.
2. Prove that $R(\theta)^{-1} = R(-\theta)$ using linear algebra and geometric identities.
3. Prove that $R(\theta_1)R(\theta_2) = R(\theta_1 + \theta_2)$ using geometric identities.
4. Give examples of 3D rotation matrices for which $R_1 R_2 \neq R_2 R_1$.
5. Prove that vector length is preserved under rotation: $\|R(\theta)\mathbf{x}\| = \|\mathbf{x}\|$.
6. Use the above fact to prove that distance is preserved under rigid transforms.

Length-Preserving Transforms and Reflections

7. Prove that all length-preserving transforms in 2D are either rotations or rotations followed by a mirroring transform:

$$T(\mathbf{x}) = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} \mathbf{x}$$

Rigid Transformations

8. Produce the transform (i.e., rotation matrix R and translation t) that rotates any point $\mathbf{x}$ by 45° about the point $(1, 0)$.
9. What is the general form of a rotation of angle θ about a center point $\mathbf{c}$?

Coordinate Frames and Transformations

10. A mobile robot has a coordinate convention where X is forward and Y is to the left. It has a sweeping laser sensor at point $(0.7, 0.05)$ in the robot's X, Y frame, aligned so that 0° points in the X direction. At time t, the sensor detects an obstacle at 1m at the 15° reading. At $t + 1$, the robot moves 0.5m forward, 0.1m to the left, and rotates 20° counterclockwise. Assuming the obstacle doesn't move, at which angle and distance will the sensor detect the obstacle?
11. A 3D camera produces points (x, y, z) where $+x$ points to the right, $+y$ upward, and $+z$ forward. Is this a right-handed or left-handed coordinate system?
12. Suppose we wish to transform points from the camera so that:
    - The camera origin is at $(1, 0, 2)$ in the world frame.
    - The camera's forward points in the world X direction.
    - Up in the image maps to world Z.
    - Left in the image maps to world Y.

Provide a transformation from camera to world points.

13. Prove:

$$T^{-1}(R, t) = T(R^T, -R^T t)$$

Semantics of Points and Directions

15. The midpoint between positions P and Q is given by $M = 0.5 \cdot (P + Q)$, but this appears to violate the assumption that addition and scalar multiplication of positions are not meaningful. How can this equation be justified using meaningful operations?

Coordinate Management System Implementation

16. Implement a 2D coordinate management system in your programming language of choice. Each position must be annotated with a frame. The system should:
    - Allow creation of named frames and positions.
    - Retrieve coordinates of a position in any frame.
    - Define a `Point` structure with:
        - `coords`: coordinates in its frame.
        - `frame`: name of the reference frame.
    - Define a `Directional` structure similarly.
    - Implement `Point.to(frame)` and `Directional.to(frame)` methods.
17. Extend the system to enforce that geometric operations occur between objects in the same frame:
    - `Point - Point = Directional`
    - `Point + Directional = Point`
    - `Directional +/- Directional = Directional`
    - `Directional * scalar = Directional`
18. Extend the system to include units for `Point`, `Directional`, and `Frame` structures. Supported units: 'm', 'mm', 'cm', 'km'.

Skeleton Code

```
class Frame:
    # TODO: what here?
    pass

named_frames = dict()

class Point:
    def __init__(self, coords, frame):
        self.coords = coords
        self.frame = frame
    def to(self, newframe):
        """Returns a Point expressing the same point in space,
        but represented in the new frame"""
        # TODO: what here?
        return Point(self.coords, newframe)
```

```
class Directional:
    def __init__(self, coords, frame):
        self.coords = coords
        self.frame = frame
    def to(self, newframe):
        """Returns a Directional expressing the same direction in space,
        but represented in the new frame"""
        # TODO: what here?
        return Point(self.coords, newframe)
```