# 22AIE214 INTRODUCTION TO AI ROBOTICS

## Labsheet 3

Representing position and orientation in 3D surface

ANANTHAKRISHNAN S

AM.SC.U4AIE23019

1. **Consider a pure rotation of 90 radians expressed as a rotation matrix**

   >> R = trotx(90)

```
R =  1   0   0   0
     0   0  -1   0
     0   1   0   0
     0   0   0   1
```

```
angle_rad = 90;
theta = deg2rad(angle_rad)
R = trotx(theta)
```

---

```
theta = 1.5708

R = 4×4

     1     0     0     0
     0     0    -1     0
     0     1     0     0
     0     0     0     1
```

2. **Create an initial transformation plane, with origin (0,0)**

   plotvol([-5 10 -5 10 -5 10]);

   grid on;

   xlabel('X-axis'); ylabel('Y-axis'); zlabel('Z-axis');

   T0 = eye(4,4);

   trplot(T0, 'frame', '0', 'color', 'b', 'length', 2);

   view(3); Set 3D perspective

```
plotvol([-5 10 -5 10 -5 10]);
grid on;
T0 = eye(4)
ylabel('Y-axisI); zlabelCZ-axis');
trplot(T0, 'frame', 'color', 'length', 2);
view(3);
```

```
T0 = 4×4
     1     0     0     0
     0     1     0     0
     0     0     1     0
     0     0     0     1
```

**3. Translate the coordinate plane from (0,0,0) to (2,3,4) {Pure translation}**

X = transl(2, 3, 4)

trplot(X, 'frame', '1', 'color', 'b', 'length', 2);

view(3);

```
X = transl(2,3,4)
trplot(X, 'frame', 1, 'color', 'b', 'length', 2);
view(3);
```

```
X = 4×4
     1     0     0     2
     0     1     0     3
     0     0     1     4
     0     0     0     1
```

4. **Rotate the initial plane to 45 degree {Pure rotation}**

R = trotx(45)
trplot(R, 'frame', '2', 'color', 'r', 'length', 2);
view(3);

```
R = trotx(45)
trplot(R, 'frame', '2', 'color', 'r', 'length', 2);
view(3);
```

```
R = 4×4
     1.0000         0         0         0
          0    0.5253   -0.8509         0
          0    0.8509    0.5253         0
          0         0         0    1.0000
```

5. **Rotate the translated plane to 45 degree**

trplot(X*R, 'frame', '3', 'color', 'r', 'length', 2);
view(3);

```
trplot(X*R, 'frame', '3', 'color', 'r', 'length', 2);
view(3);
```

6. **Create a homogeneous transformation which represents a translation of (7,4) followed by a rotation of 30°**

T = transl(7, 4, 6)* troty(30);

trplot(T, 'frame', '4', 'color', 'g', 'length', 2);

view(3);

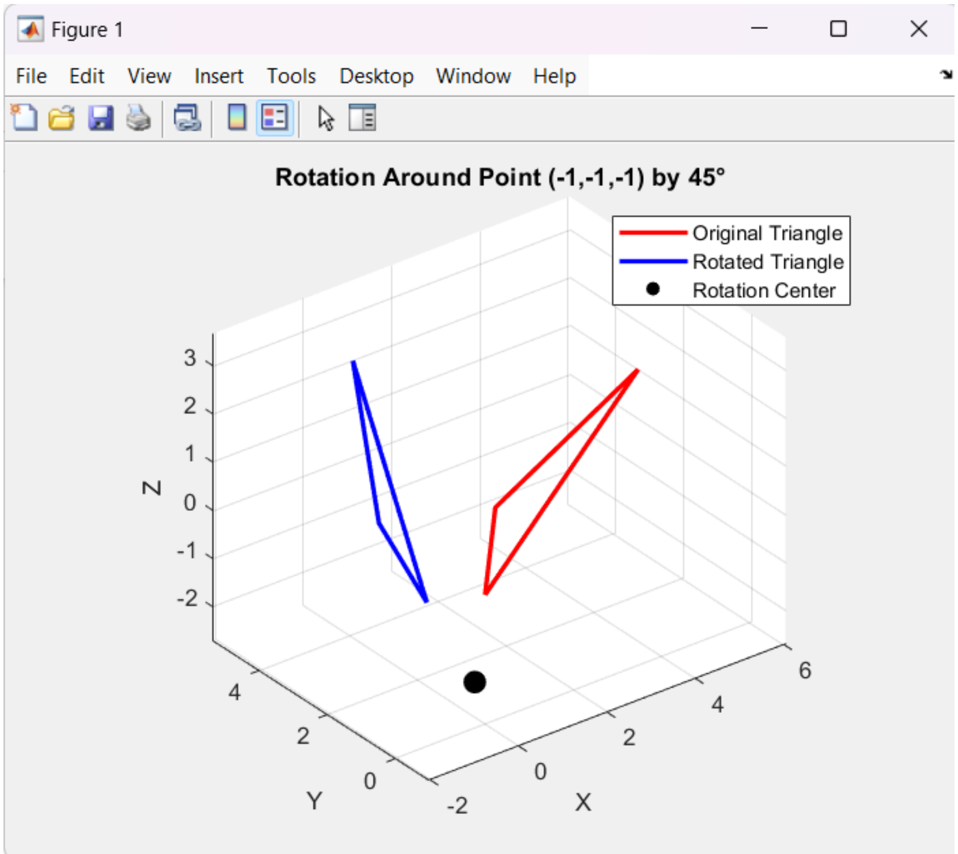
```
T = transl(7, 4, 6)* troty(30);
trplot(T, 'frame', '4', 'color', 'g', 'length', 2);
view(3);
```

1) Rotate a triangle placed at A(0,0,0), B(1,1,1) and C(5,2,2) by an angle 45 with respect to point P(-1,-1). Plot the points.

```
clc; clear; close all;
% Define the original triangle vertices
A = [0 0 0];
B = [1 1 1];
C = [5 2 2];
% Define the rotation center
P = [-1 -1 -1];
% Define the rotation angle (in degrees) and convert to radians
theta = 45;
theta_rad = deg2rad(theta);
% Rotation matrix about the Z-axis
R = trotz(theta_rad);
% Convert points to homogeneous coordinates
A_h = [A 1]';
B_h = [B 1]';
C_h = [C 1]';
% Define translation transformations
T1 = transl(-P(1), -P(2), -P(3)); % Translate P to the origin
T2 = transl(P(1), P(2), P(3)); % Translate back after rotation
% Apply rotation
A_rot = T2 * R * T1 * A_h;
```

```
B_rot = T2 * R * T1 * B_h;
C_rot = T2 * R * T1 * C_h;
% Extract rotated points from homogeneous coordinates
A_rot = A_rot(1:3)';
B_rot = B_rot(1:3)';
C_rot = C_rot(1:3)';
% Plot the original and rotated triangles
figure;
hold on; grid on; axis equal;
xlabel('X'); ylabel('Y'); zlabel('Z');
% Plot original triangle in red
plot3([A(1) B(1) C(1) A(1)], [A(2) B(2) C(2) A(2)], [A(3) B(3) C(3) A(3)], 'r-',
'LineWidth', 2);
% Plot rotated triangle in blue
plot3([A_rot(1) B_rot(1) C_rot(1) A_rot(1)], ...
[A_rot(2) B_rot(2) C_rot(2) A_rot(2)], ...
[A_rot(3) B_rot(3) C_rot(3) A_rot(3)], 'b-', 'LineWidth', 2);
% Mark the rotation center
scatter3(P(1), P(2), P(3), 100, 'k', 'filled');
% Add legend and title
legend('Original Triangle', 'Rotated Triangle', 'Rotation Center');
title('Rotation Around Point (-1,-1,-1) by 45°');
view(3);
```

Output:

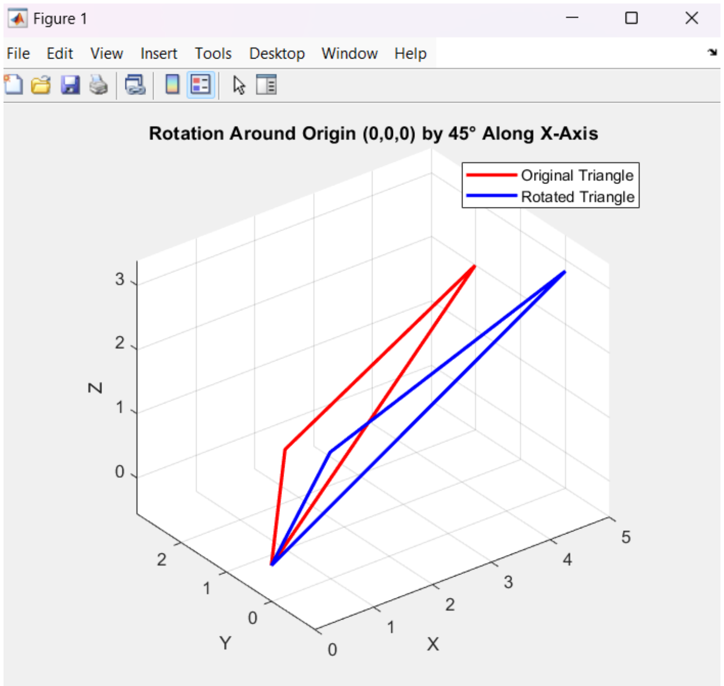



2) Rotate a triangle placed at A(0,0,0), B(1,1,1) and C(5,2,2) by an angle 45 with respect to origin. Plot the points.

```
clc; clear; close all;
A = [0 0 0];B = [1 1 1];C = [5 2 2];
theta = 45;theta_rad = deg2rad(theta);
R = trotx(theta_rad);
A_h = [A 1]';  B_h = [B 1]';  C_h = [C 1]';
A_rot = R * A_h;
B_rot = R * B_h;
C_rot = R * C_h;
A_rot = A_rot(1:3)';
B_rot = B_rot(1:3)';
C_rot = C_rot(1:3)';
figure;hold on; grid on; axis equal;
xlabel("X"); ylabel("Y"); zlabel("Z");
plot3([A(1) B(1) C(1) A(1)], [A(2) B(2) C(2) A(2)], [A(3) B(3) C(3) A(3)], "r-",
"LineWidth", 2);
plot3([A_rot(1) B_rot(1) C_rot(1) A_rot(1)], ...
[A_rot(2) B_rot(2) C_rot(2) A_rot(2)], ...
[A_rot(3) B_rot(3) C_rot(3) A_rot(3)], "b-", "LineWidth", 2);
```

```
legend("Original Triangle", "Rotated Triangle");
title("Rotation Around Origin (0,0,0) by 45° Along X-Axis");view(3);
```

Output:




3) What is Gimbal lock . Explain in 10 sentences. Experiment with the triple angle in Matlab robotics toolbox. Explore roll pitch and yaw motions.
[Command, screenshot of default model and the screenshot of gimbal lock model.]

1. Gimbal lock occurs when two rotational axes in a three-axis gimbal system align, reducing the system's degrees of freedom from three to two.
2. This results in the loss of one independent axis, making certain rotations impossible and causing unpredictable motion.
3. It happens in systems using **Euler angles** (roll, pitch, yaw), where sequential rotations can lead to alignment of two axes.

4. In aviation and robotics, gimbal lock can cause loss of control over an object’s orientation.

5. The **Apollo 11 spacecraft** had to account for gimbal lock to maintain stable navigation.

6. The main cause of gimbal lock is a **90-degree rotation** about one axis, which makes another axis redundant.

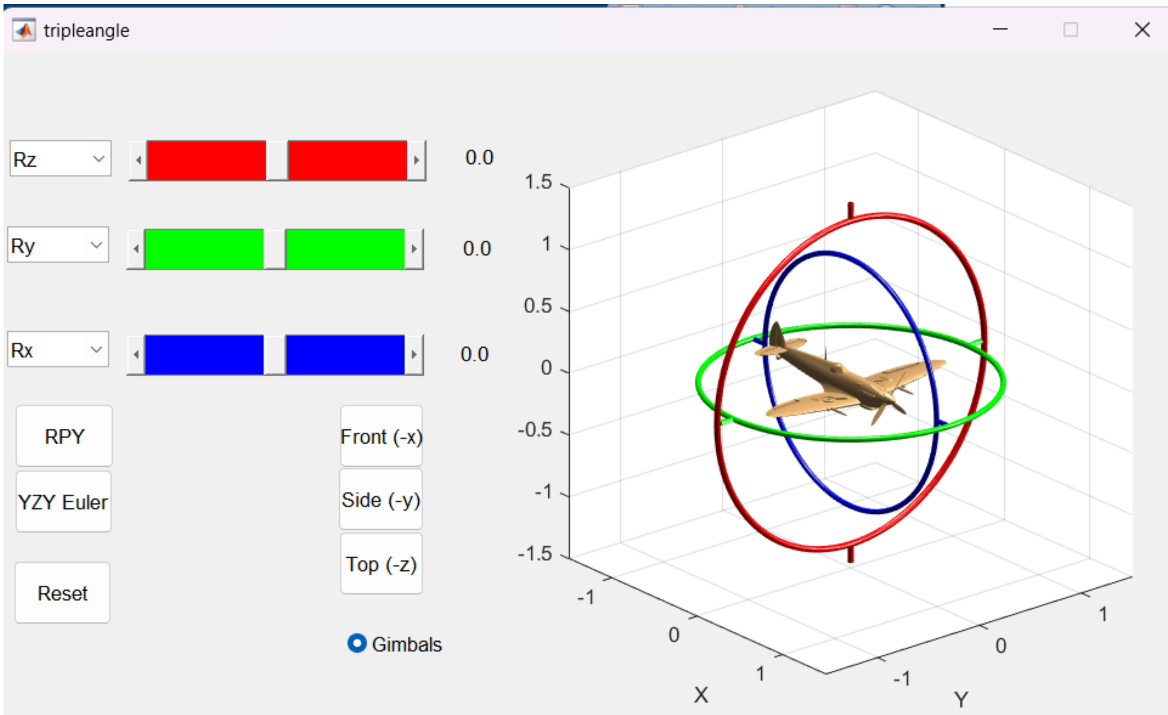
7. A common solution is to use **quaternions**, which provide smooth and continuous rotation without singularities.

8. In **3D animations and game development**, gimbal lock can lead to erratic movements of objects or characters.

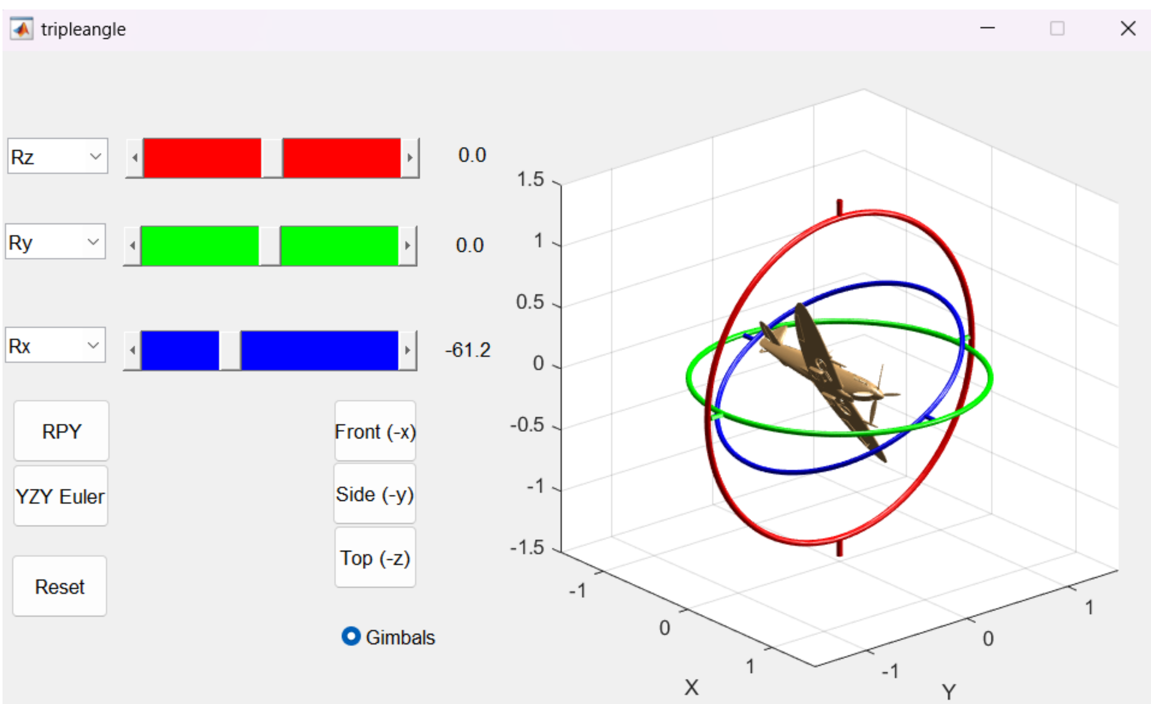
9. Robotic systems often use **rotation matrices** or **quaternions** instead of Euler angles to avoid this issue.

10. Engineers and programmers often implement alternative rotation methods, such as axis-angle representation or dual quaternions, to prevent gimbal lock issues.

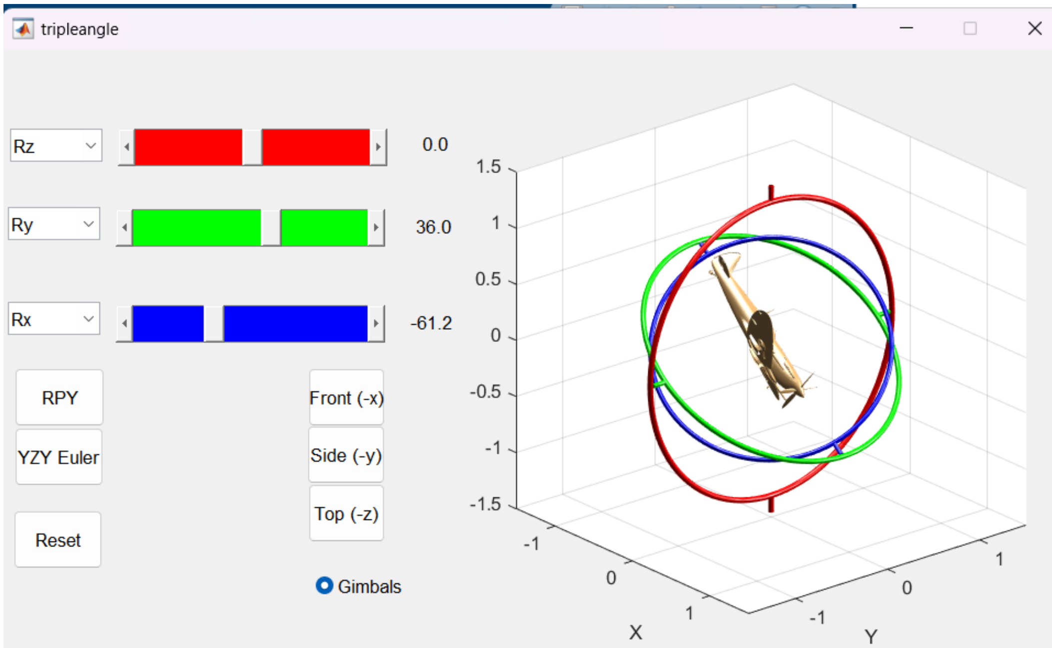
Default:


tripleangle
Rz
0.0
Ry
0.0
Rx
0.0
RPY
YZY Euler
Reset
Front (-x)
Side (-y)
Top (-z)
Gimbals
X
Y

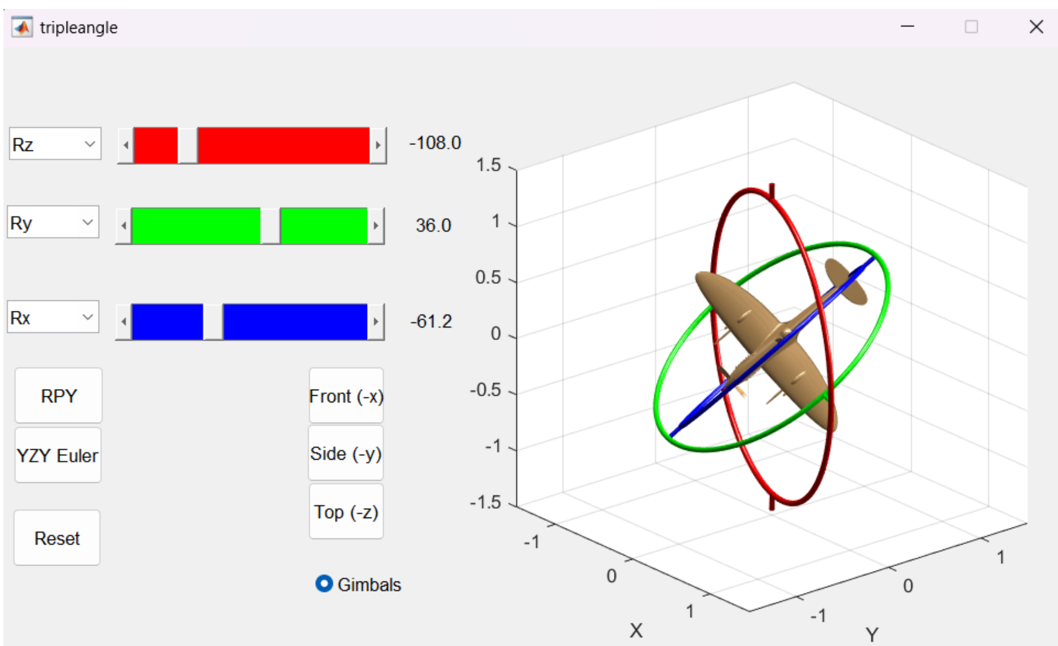
Roll:


tripleangle
Rz
0.0
Ry
0.0
Rx
-61.2
RPY
YZY Euler
Reset
Front (-x)
Side (-y)
Top (-z)
Gimbals
X
Y

Yaw:


tripleangle
Rz
0.0
Ry
36.0
Rx
-61.2
RPY
YZY Euler
Reset
Front (-x)
Side (-y)
Top (-z)
Gimbals
X
Y

Pitch:


tripleangle
Rz
-108.0
Ry
36.0
Rx
-61.2
RPY
YZY Euler
Reset
Front (-x)
Side (-y)
Top (-z)
Gimbals
X
Y