CENG483: Behavioural Robotics
Izmir Institute of Technology
Fall 2023
Instructor: Dr. Berat Alper Erol

Homework Set: 02

Ozan Gülbaş

This homework answers the problem set sequentially.

1. *Use MATLAB Robotics Toolbox and write functions for Theta (degree) = 25, 45, 65, 82:*

   (a) *ROTX(theta). This function will give a 4x4 homogenous transformation matrix that will rotate "theta" about the X-axis.*

   ```
   function output_matrix  = ROTX(theta)
       output_matrix = trotx(theta, "deg");
   end
   ```

   (b) *ROTY(theta). This function will give a 4x4 homogenous transformation matrix that will rotate "theta" about the Y-axis*

   ```
   function output_matrix  = ROTY(theta)
       output_matrix = troty(theta, "deg");
   end
   ```

   (c) *ROTZ(theta). This function will give a 4x4 homogenous transformation matrix that will rotate "theta" about the Z-axis*

   ```
   function output_matrix  = ROTZ(theta)
       output_matrix = trotz(theta, "deg");
   end
   ```

   (d) *TRANS(x,y,z). This function will give a 4x4 homogenous transformation matrix that will translate "x" units over the X-axis, "y" units over the Y-axis, and "z" units over the Z-axis*

   ```
   function output_matrix = TRANS(x, y, z)
       output_matrix = transl(x, y, z);
   end
   ```

.

The code which displays ROTX,ROTY and ROTZ for given Thetas:

```
thetas = [25, 45, 65, 82];

for i = 1 : length(thetas)
    theta = thetas(i);

    rot_matrix_X = ROTX(theta);
    rot_matrix_Y = ROTY(theta);
    rot_matrix_Z = ROTZ(theta);

    fprintf("\nROTX with theta : %d = \n", theta)
    disp(rot_matrix_X)
    fprintf("\nROTY with theta : %d = \n", theta)
    disp(rot_matrix_Y)
    fprintf("\nROTZ with theta : %d = \n", theta)
    disp(rot_matrix_Z)

    fprintf("\n=======================\n")
end
```

The output is:

```
ROTX with theta : 25 =
    1.0000         0         0         0
         0    0.9063   -0.4226         0
         0    0.4226    0.9063         0
         0         0         0    1.0000


ROTY with theta : 25 =
    0.9063         0    0.4226         0
         0    1.0000         0         0
   -0.4226         0    0.9063         0
         0         0         0    1.0000


ROTZ with theta : 25 =
    0.9063   -0.4226         0         0
    0.4226    0.9063         0         0
         0         0    1.0000         0
         0         0         0    1.0000


=======================

ROTX with theta : 45 =
    1.0000         0         0         0
         0    0.7071   -0.7071         0
```

```
         0    0.7071    0.7071         0
         0         0         0    1.0000


ROTY with theta : 45 =
    0.7071         0    0.7071         0
         0    1.0000         0         0
   -0.7071         0    0.7071         0
         0         0         0    1.0000


ROTZ with theta : 45 =
    0.7071   -0.7071         0         0
    0.7071    0.7071         0         0
         0         0    1.0000         0
         0         0         0    1.0000


=======================

ROTX with theta : 65 =
    1.0000         0         0         0
         0    0.4226   -0.9063         0
         0    0.9063    0.4226         0
         0         0         0    1.0000


ROTY with theta : 65 =
    0.4226         0    0.9063         0
         0    1.0000         0         0
   -0.9063         0    0.4226         0
         0         0         0    1.0000


ROTZ with theta : 65 =
    0.4226   -0.9063         0         0
    0.9063    0.4226         0         0
         0         0    1.0000         0
         0         0         0    1.0000


=======================

ROTX with theta : 82 =
    1.0000         0         0         0
         0    0.1392   -0.9903         0
         0    0.9903    0.1392         0
         0         0         0    1.0000
```

```
ROTY with theta : 82 =
    0.1392         0    0.9903         0
         0    1.0000         0         0
   -0.9903         0    0.1392         0
         0         0         0    1.0000


ROTZ with theta : 82 =
    0.1392   -0.9903         0         0
    0.9903    0.1392         0         0
         0         0    1.0000         0
         0         0         0    1.0000


=======================
```

2. *Use the resulting functions of previous problem to do the following operations:*

   (a) *Find a final transformation that translates a frame N 2 units in X, 3 unites in Y and -2 units in Z, i.e. (2,3,-2), and then rotate the translated frame 60 degrees over its translated Y axis.*
   Code:

```
N = TRANS(2, 3, -2);
N_rot = N * ROTY(60);
fprintf("Resulting matrix N:\n")
disp(N_rot)
```

   Output:

```
Resulting matrix N:
    0.5000         0    0.8660    2.0000
         0    1.0000         0    3.0000
   -0.8660         0    0.5000   -2.0000
         0         0         0    1.0000
```

   (b) *Find a final transformation rotates a frame O 60 degrees over its translated Y axis and then translates the rotated frame (2,-2, 2).*
   Code:

```
O = ROTY(60);
O_trans = O * TRANS(2, -2, 2);
fprintf("Resulting matrix O:\n")
disp(O_trans)
```

   Output:

```
Resulting matrix O:
    0.5000         0    0.8660    2.7321
         0    1.0000         0   -2.0000
   -0.8660         0    0.5000   -0.7321
         0         0         0    1.0000
```

3. *Create a 2D rotation matrix. Visualize the rotation using* ***trplot2****. Use it to transform a vector. Invert it and multiply it by the original matrix; what is the result? Reverse the order of multiplication; what is the result? What is the determinant of the matrix and its inverse?*

   Code:

```
% Create a 2D rotation matrix
    theta = 60;
    T = [
        cos(theta), -sin(theta);
        sin(theta), cos(theta);
    ];
% Visualise the rotation with trplot2
    figure(1)
    trplot2(T, "frame", "A")

% Use it to transform a vector
    V = [1; 1];
    V_trans = T * V;
    fprintf("Transform of vector V with T:\n")
    disp(V_trans)

% Invert it and multiply it with the original matrix
    T_inv = inv(T);
    result1 = T_inv * T;
    fprintf("Result 1:\n")
    disp(result1)

% Reverse the order of multipication
    result2 = T * T_inv;
    fprintf("Result 2:\n")
    disp(result2)

% Determinant of the matrix and its inverse
    det_T = det(T);
    det_T_inv = det(T_inv);
    fprintf("Determinant of T:\n")
    disp(det_T)
    fprintf("Determinant of T inverse:\n")
    disp(det_T_inv)
```

   Output:

```
Transform of vector V with T:
   -0.6476
   -1.2572

Result 1:
```

```
    1.0000    0.0000
         0    1.0000

Result 2:
    1.0000    0.0000
         0    1.0000

Determinant of T:
     1

Determinant of T inverse:
     1
```

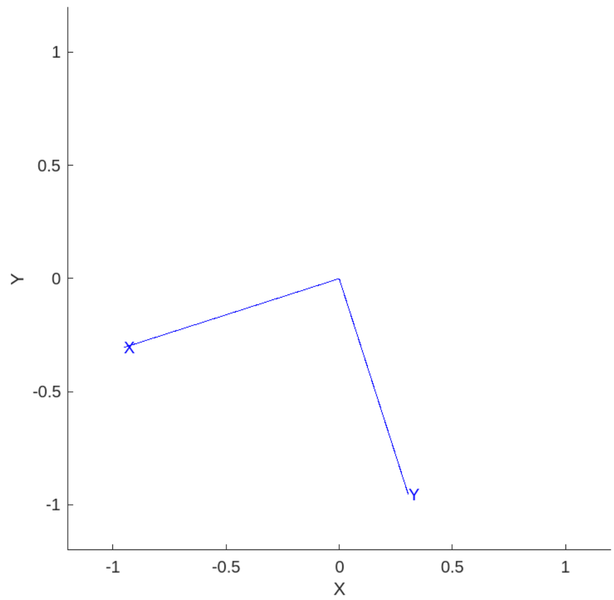

Figure 1: Visualization of the rotation with *trplot2*.

4. *Create a 3D rotation matrix. Visualize the rotation using **trplot** or **tranimate**. Use it to transform a vector. Invert it and multiply it by the original matrix; what is the result? Reverse the order of multiplication; what is the result? What is the determinant of the matrix and its inverse?*

   Code:

```
% Create a 3D rotation matrix (wrt Y axis)
    theta = 60;
    R_y = [
        cos(theta), 0, sin(theta);
        0, 1, 0;
        -sin(theta), 0, cos(theta);
    ];

% Visualise the rotation with trplot
    figure(2)
    trplot(R_y, "frame", "A")
    view(10,10)

% Transform a vector
    V = [1; 1; 1];
    V_trans = R_y * V;
    fprintf("Transform of vector V with R_y:\n")
    disp(V_trans)

% Invert it and multiply it with the original matrix
    R_y_inv = inv(R_y);
    result1 = R_y_inv * R_y;
    fprintf("Result 1:\n")
    disp(result1)

% Reverse the order of multiplication
    result2 = R_y * R_y_inv;
    fprintf("Result 2:\n")
    disp(result2)

% Determinant of R_y and R_y inverse
    det_R_y = det(R_y);
    det_R_y_inv = det(R_y_inv);
    fprintf("Determinant of R_y:\n")
    disp(det_R_y)
    fprintf("Determinan of R_y inverse:\n")
    disp(det_R_y_inv)
```

   Output:

```
Transform of vector V with R_y:
   -1.2572
```

```
    1.0000
   -0.6476

Result 1:
    1.0000         0   -0.0000
         0    1.0000         0
         0         0    1.0000

Result 2:
    1.0000         0   -0.0000
         0    1.0000         0
         0         0    1.0000

Determinant of R_y:
     1

Determinan of R_y inverse:
     1
```

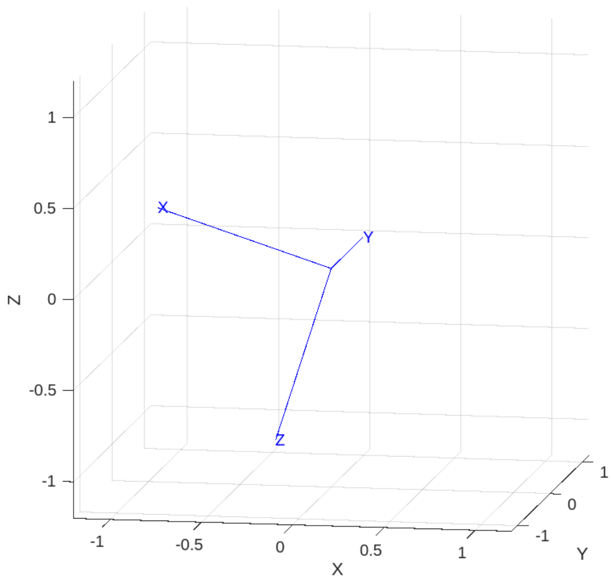

Figure 2: Visualization of the rotation with *trplot*.

5. *Generate the sequence of plots shown in Fig. 2.12.*

   Code:

```
    base_pose = roty(0);

% First sequence
    rot1 = ROTX(90);
    rot2 = rot1 * ROTY(90);

% Second sequence
    rot3 = ROTY(90);
    rot4 = rot3 * ROTX(90);

    figure(3)
    subplot(2, 3, 1)
    trplot(base_pose)
    xaxis(-0.2, 1.2);yaxis(-0.2, 1.2); zlim([-0.2, 1.2])
    view(10, 10)

    subplot(2, 3, 2)
    trplot(rot1)
    xaxis(-0.2, 1.2);yaxis(-1.2, 0.2); zlim([-0.2, 1.2])
    view(10, 10)

    subplot(2, 3, 3)
    trplot(rot2)
    xaxis(-0.2, 1.2);yaxis(-0.2, 1.2); zlim([-0.2, 1.2])
    view(10, 10)

    subplot(2, 3, 4)
    trplot(base_pose)
    xaxis(-0.2, 1.2);yaxis(-0.2, 1.2); zlim([-0.2, 1.2])
    view(10, 10)

    subplot(2, 3, 5)
    trplot(rot3)
    xaxis(-0.2, 1.2);yaxis(-0.2, 1.2); zlim([-1.2, 0.2])
    view(10, 10)

    subplot(2, 3, 6)
    trplot(rot4)
    xaxis(-0.2, 1.2);yaxis(-1.2, 0.2); zlim([-1.2, 0.2])
    view(10, 10)
```

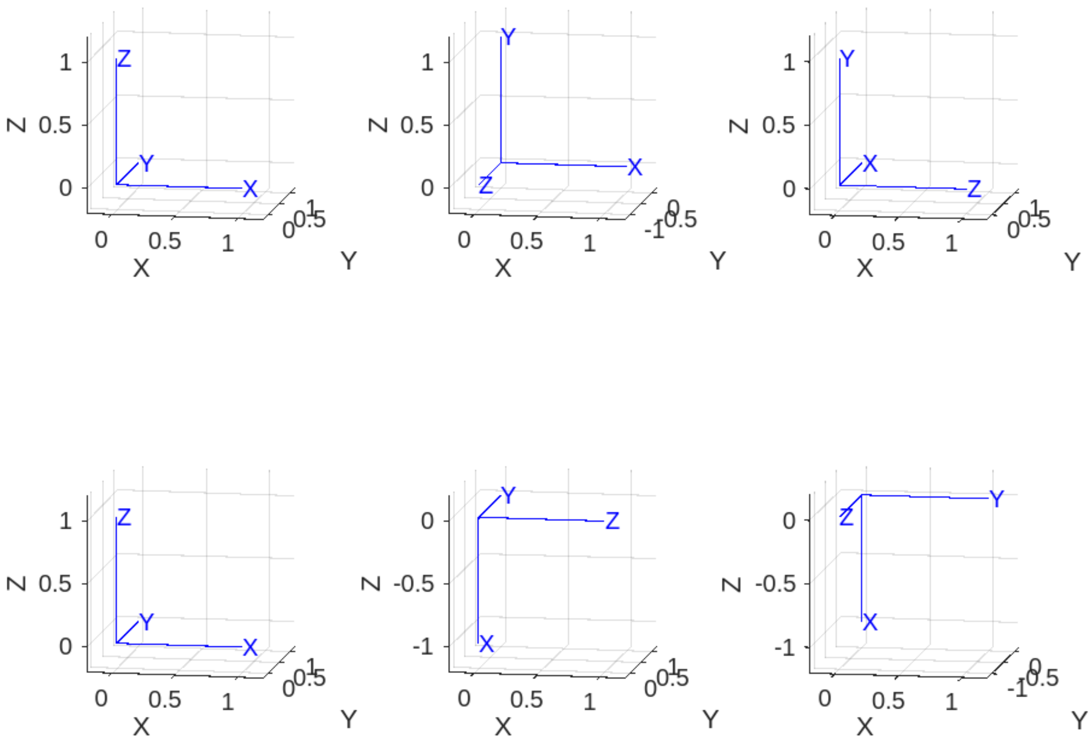

Figure 3: Recreation of Fig. 2.12 in the textbook.

6. *For the 3-dimensional rotation about the vector [2, 3, 4] by 0.5 rad compute an SO(3) rotation matrix using: the matrix exponential functions* ***expm*** *and* ***trexp****,* ***Rodrigues' rotation formula*** *(code this yourself), and the Toolbox function* ***angvec2tr****. Compute the equivalent unit quaternion.*

   First, the Rodrigues's forluma implemented like this:

```
function output_matrix = Rodrigues(theta, V)
    I = [
    1, 0, 0;
    0, 1, 0;
    0, 0, 1;
    ];

    skew_matrix = skew(V) ;

    output_matrix = I + sin(theta) * skew_matrix
        + (1 - cos(theta)) * skew_matrix^2;
end
```

   Code:

```
% A 3D rotation matrix (wtr Y axis)
    rad = 0.5;
    theta = rad2deg(rad);
    R_y = [
        cos(theta), 0, sin(theta);
        0, 1, 0;
        -sin(theta), 0, cos(theta);
    ];

% Define vector
    V = [2; 3; 4];
    V_normalized = V / norm(V);

% Rotation with expm()
    rot_expm = expm(skew(V_normalized) * rad);
    fprintf("Rotation matrix with expm():\n")
    disp(rot_expm)

% Rotation with trexp()
    rot_trexp = trexp(skew(V_normalized) * rad);
    fprintf("Rotation matrix with trexp():\n")
    disp(rot_trexp)

% Rotation with Rodrigues's formula
    rot_rodriguez = Rodrigues(rad, V_normalized);
    fprintf("Rotation matrix with Rodrigues's formula:\n")
    disp(rot_rodriguez)
```

```
% Rotation with angvec2tr()
    rot_toolbox = angvec2tr(rad, V_normalized);
    fprintf("Rotation matrix with angvec2tr\n")
    disp(rot_toolbox)

% Equivalent unit quaternion
    quaternion = UnitQuaternion(rot_expm);
    fprintf("Equivalent unit quaternion:\n")
    disp(quaternion)
```

Output:

```
Rotation matrix with expm():
    0.8945   -0.3308    0.3009
    0.3814    0.9156   -0.1274
   -0.2333    0.2287    0.9451

Rotation matrix with trexp():
    0.8945   -0.3308    0.3009
    0.3814    0.9156   -0.1274
   -0.2333    0.2287    0.9451

Rotation matrix with Rodrigues's formula:
    0.8945   -0.3308    0.3009
    0.3814    0.9156   -0.1274
   -0.2333    0.2287    0.9451

Rotation matrix with angvec2tr:
    0.8945   -0.3308    0.3009         0
    0.3814    0.9156   -0.1274         0
   -0.2333    0.2287    0.9451         0
         0         0         0    1.0000

Equivalent unit quaternion:
  UnitQuaternion with properties:

    s: 0.9689
    v: [0.0919 0.1378 0.1838]
```