Kinematics pick and place project write up

This project uses a simulated Kuka KR210 6 degree of freedom manipulator to pick up a can from a shelf and drop it into a bin next to the manipulator.

I have created the inverse kinematics code in such a way that it will make sure that each joint in the arm has to maintain the correct angle in order to pick the can and drop it in the correct location.

Joint Based Transformation Matrices

Each and every joint has their own transformation matrix , which were described below. Once DH parameters are derived those values were used in the below matrix.

```
TM = [[        cos(θ),       -sin(θ),       0,         a],
      [ sin(θ)*cos(α), cos(θ)*cos(α), -sin(α), -sin(α)*d],
      [ sin(θ)*sin(α), cos(θ)*sin(α),  cos(α),  cos(α)*d],
      [             0,             0,       0,         1]]
```

Using the transformation matrix formula above, here are the joint transformation matrices for the arm:

```
Joint 1: [[ cos(θ1), -sin(θ1),  0,     0],
          [ sin(θ1),  cos(θ1),  0,     0],
          [       0,        0,  1,  0.75],
          [       0,        0,  0,     1]]
Joint 2: [[ sin(θ2),  cos(θ2),  0,  0.35],
          [       0,        0,  1,     0],
          [ cos(θ2), -sin(θ2),  0,     0],
          [       0,        0,  0,     1]]
Joint 3: [[ cos(θ3), -sin(θ3),  0,  1.25],
          [ sin(θ3),  cos(θ3),  0,     0],
          [       0,        0,  1,     0],
          [       0,        0,  0,     1]]
Joint 4: [[ cos(θ4), -sin(θ4),  0, -0.054],
          [       0,        0,  1,    1.5],
```

```
          [−sin(θ4), −cos(θ4),  0,      0],
          [       0,        0,  0,      1]]
Joint 5: [[ cos(θ5), −sin(θ5),  0,      0],
          [       0,        0, −1,      0],
          [ sin(θ5),  cos(θ5),  0,      0],
          [       0,        0,  0,      1]]
Joint 6: [[ cos(θ6), −sin(θ6),  0,      0],
          [       0,        0,  1,      0],
          [−sin(θ6), −cos(θ6),  0,      0],
          [       0,        0,  0,      1]]
```

Below is the transformation matrix for arm gripper , which holds the can.

```
Gripper: [[       1,        0,  0,      0],
          [       0,        1,  0,      0],
          [       0,        0,  1,  0.303],
          [       0,        0,  0,      1]]
```

Inverse Kinematic Orientation

To derive the inverse kinematic orientation, I have taken the rotation matrix for 4,5,6 joints and calculated the rulers angles.

spherical wrist's rotation matrix can be calculated by using:

```
R3_6 = R0_3.T * Rrpy * R_corr
```

Where:

- `R0_3.T` is the transposition of the rotation matrix from joints 1, 2 and 3, requiring the angles derived in the inverse kinematic position calculations
- `Rrpy` is the rotation matrix of the gripper's current roll, pitch and yaw
- `R_corr` is the rotation matrix of the gripper correction matrix that rotates the gripper around the Z axis by 180 degrees and around the Y axis by -90 degrees

With this rotation matrix, it is possible to derive the Euler angles.

Euler Angles

I used a tf transformations function called `euler_from_matrix` that takes in a numpy rotation matrix and Euler axis sequence, and returns the three Euler angles (alpha, beta and gamma).

The rotation matrix I provided used the Euler definition of `XYZ`, which is a Tait-Bryan angle combination. With the alpha, beta and gamma angles, I mapped them to theta 4, theta 5 and theta 6, respectively.

However, theta 4 and theta 5 required these additional calculations:

```
theta4 = np.pi/2 + theta4
theta5 = np.pi/2 - theta5
```

Final output :