# Robotic Systems:
# Project Part 1

Christopher Waschenko & Johan Gamboa

November 3, 2025

## Contents



## List of Tables

# 1 Selected Robot

## 1.1 UR5 Robot Kinematic Model

The Universal Robots UR5 is a 6-DOF industrial manipulator designed for flexible automation tasks such as assembly, pick-and-place, and machine tending. It features a lightweight structure with a reach of approximately 85 cm and a payload capacity of 5 kg.

## 1.2 DH Table

| **Joint** | $\mathbf{a_i}$ | $\mathbf{d_i}$ | $\boldsymbol{\alpha_i}$ | $\boldsymbol{\theta_i}$ |
|---|---|---|---|---|
| 1 | 0 | 8.9459 | $\frac{\pi}{2}$ | $\theta_1$ |
| 2 | -42.5 | 0 | 0 | $\theta_2$ |
| 3 | -39.225 | 0 | 0 | $\theta_3$ |
| 4 | 0 | 10.915 | $\frac{\pi}{2}$ | $\theta_4$ |
| 5 | 0 | 9.465 | $-\frac{\pi}{2}$ | $\theta_5$ |
| 6 | 0 | 8.23 | 0 | $\theta_6$ |

Table 1: Standard DH Parameters for UR5 Robot (units in cm)

## 1.3 Joint Limits

| **Joint** | **Minimum (rad)** | **Maximum (rad)** | **Range (deg)** |
|---|---|---|---|
| 1 | $-2\pi$ | $2\pi$ | $[-360°, 360°]$ |
| 2 | $-2\pi$ | $2\pi$ | $[-360°, 360°]$ |
| 3 | $-2\pi$ | $2\pi$ | $[-360°, 360°]$ |
| 4 | $-2\pi$ | $2\pi$ | $[-360°, 360°]$ |
| 5 | $-2\pi$ | $2\pi$ | $[-360°, 360°]$ |
| 6 | $-2\pi$ | $2\pi$ | $[-360°, 360°]$ |

Table 2: UR5 Joint Limits in Radians and Degrees

# 2 Math

## 2.1 Math Definitions

- let $a, b, c$ be $x_{offset}$, $y_{offset}$, $z_{offset}$ from the robot's base frame to the center of the sphere respectively.
- let $r_s$ be the radius of the sphere.
- let $r_c$ be the radius of the circular path along the sphere's surface.
- let $d$ be the offset along the $z_1$ axis s.t. $d = \sqrt{r_{sphere}^2 - r_{circle}^2}$
- let a sphere be described as $s = (x-a)^2 + (y-b)^2 + (z-c)^2 - r_{sphere}^2$

## 2.2 Define Homogenous Transforms

$$H_1^0 = \begin{bmatrix} 1 & 0 & 0 & a \\ 0 & 1 & 0 & b \\ 0 & 0 & 1 & c \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

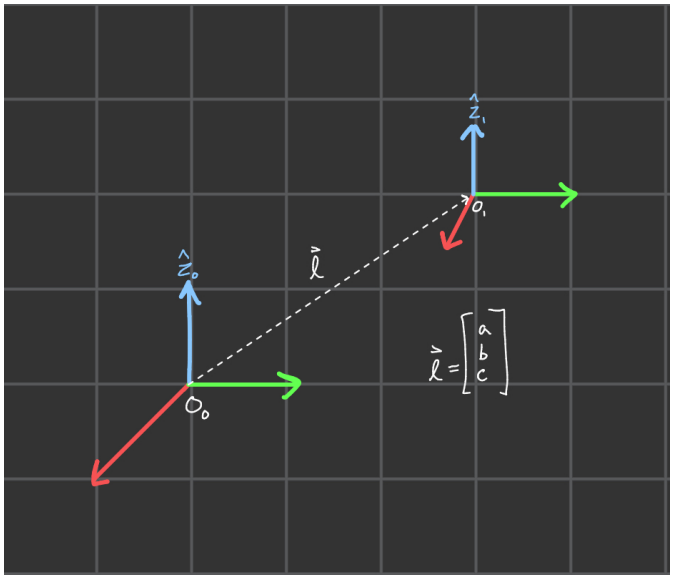


Figure 1: Translation a b and c of sphere

$$R_c = R_{x,\theta} R_{y,\phi}$$

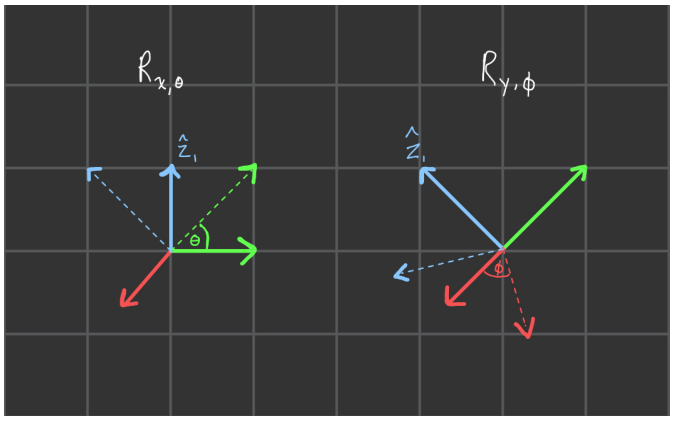


Figure 2: Composition of Rotations about the center of the sphere

$$p = \begin{bmatrix} 0 \\ 0 \\ d \end{bmatrix}$$

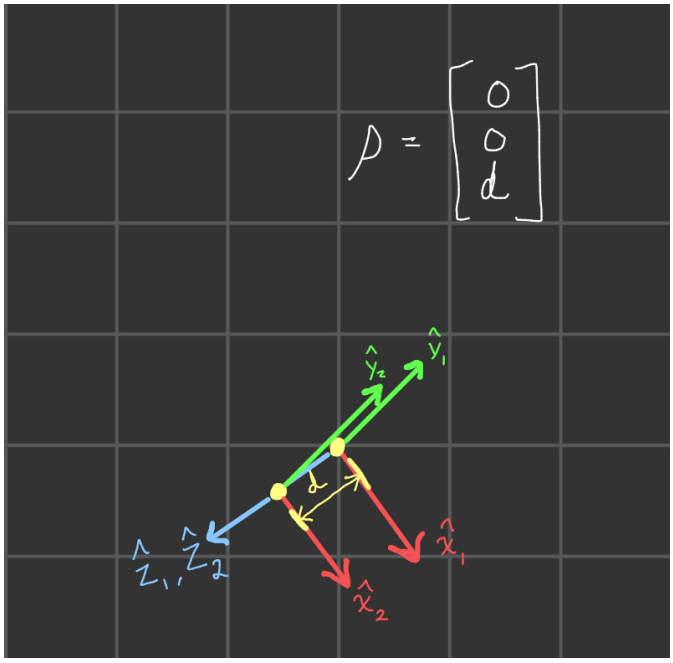


Figure 3: The offset $d$ from the rotated $z_2$ axis

$$p' = p * R_C$$

$$H_2^1 = \begin{bmatrix} R_c & p' \\ 0_{3x1} & 1 \end{bmatrix}$$

- To generate a circular path on the calculated coordinate system above, we use the parameterization of a cirular path given that are circular path has a radius of $r_c$, where $t$ is time $[0 \rightarrow 2\pi]$:

$$x_c(t) = r_c * cos(t) \qquad y_c(t) = r_c * sin(t) \qquad z_c(t) = 0$$

- This gives us our points along the circular path $p_c$.

$$p_c = \begin{bmatrix} r_c cos(t) \\ r_c sin(t) \\ 0 \end{bmatrix}$$

- Next we take the negative gradient of the sphere at the point $p_c$, we normalize that value to give the end effectors z-axis:

$$\nabla s = - \begin{bmatrix} 2 \times a - 2 \times x \\ 2 \times b - 2 \times y \\ 2 \times c - 2 \times z \end{bmatrix}$$

$$\nabla \hat{s} = \frac{\nabla s}{\|\nabla \mathbf{s}\|} = \hat{z_{ef}}$$

$$\hat{z_{ef}} = \begin{bmatrix} z_i \\ z_j \\ z_k \end{bmatrix}$$

- Choose an arbitrary unit vector $\hat{k}$ such that it is not parallel to $\hat{z}_{ef}$. This provides a stable reference to generate the local tangent frame on the sphere.

- Compute the $x$-axis of the end-effector frame as the cross product:

$$\hat{x}_{ef} = \frac{\hat{z}_{ef} \times \hat{k}}{\|\hat{z}_{ef} \times \hat{k}\|} = \begin{bmatrix} x_i \\ x_j \\ x_k \end{bmatrix}.$$

- Then, compute the $y$-axis of the end-effector frame as:

$$\hat{y}_{ef} = \hat{z}_{ef} \times \hat{x}_{ef} = \begin{bmatrix} y_i \\ y_j \\ y_k \end{bmatrix}.$$

- These unit vectors $\{\hat{x}_{ef}, \hat{y}_{ef}, \hat{z}_{ef}\}$ describe our orientation aligning the end effector to the negative gradient of the sphere.

- We include them in the final homogenous transformation matrix $H_3^2$, given the rotation matrix $(R)$ and point on the circular path $(p_c)$:

$$R = \begin{bmatrix} x_i & y_i & z_i \\ x_j & y_j & z_j \\ x_k & y_k & z_k \end{bmatrix} \qquad p_c = \begin{bmatrix} r_c cos(t) \\ r_c sin(t) \\ 0 \end{bmatrix}$$

$$H_3^2 = \begin{bmatrix} x_i & y_i & z_i & r_c cos(t) \\ x_j & y_j & z_j & r_c sin(t) \\ x_k & y_k & z_k & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- We can calculate the points of the circular path wrt to the base frame using the formula:

$$H_3^0 = H_1^0 H_2^1 H_3^2$$

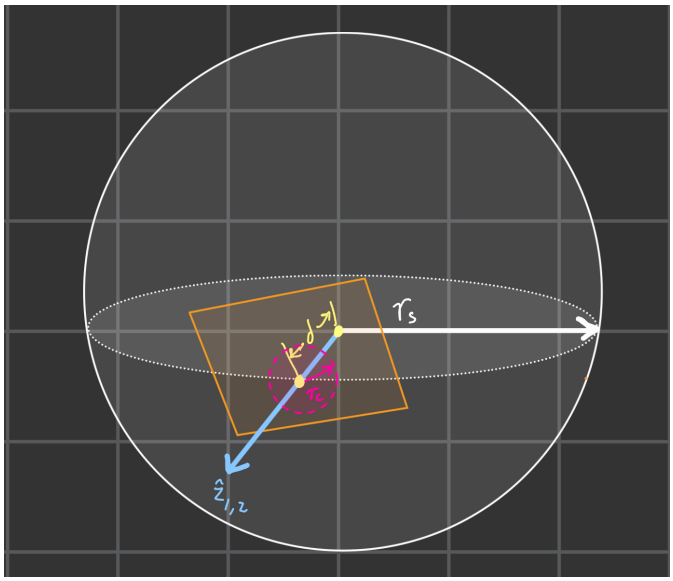

Figure 4: The circular path projected on the sphere after transformations

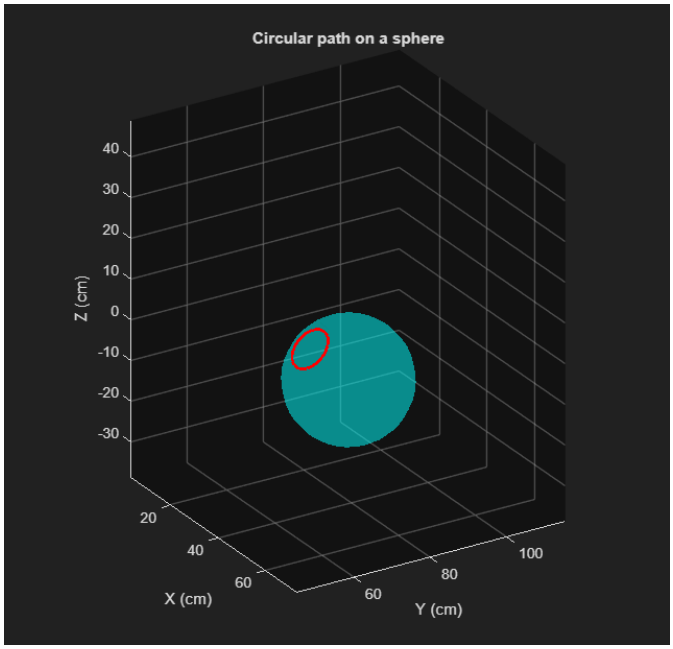


Figure 5: Matlab output of the circular path projected on the sphere after transformations

### 2.2.1 Tool Offset

The actual end-effector or tool center point (TCP) extends beyond this frame by a fixed offset along the local $z$-axis. Let the tool offset be represented as a homogeneous transformation:

$$H_T^6 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_T \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

where $d_T$ is the distance from the flange to the tool tip measured along the $z_6$-axis for our robot this is 8.23cm.

The complete forward kinematics from the robot base to the tool tip is then given by:

$$H_T^0 = H_6^0 \, H_T^6$$

This ensures that the tool tip position $p_T$ and orientation $R_T$ correspond to the actual end-effector contact point used to draw the circular trajectory on the sphere.

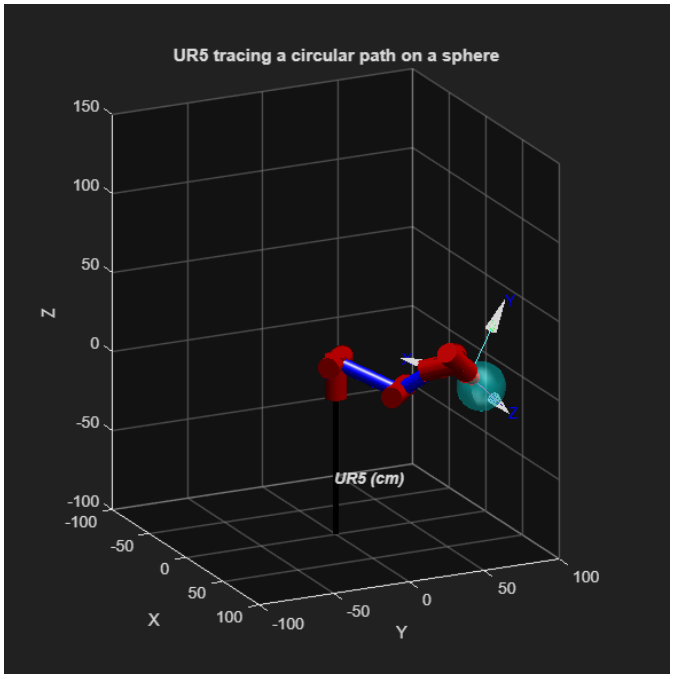


Figure 6: Matlab output of the robot interacting with the sphere

# 3 Code

```
% Course Project Part 1
% Selected Commercial Robot: Universal Robots' UR5 6DOF Robot

% UR5 Robot (DH Parameters in cm, Standard Convention)
% ----------------------------------------------------
%  Joint# |   a [cm]   |   d [cm]   |  alpha [rad]  |  theta [rad]
%    1    |   0.0      |   8.9459   |   pi/2        |   q1
%    2    |  -42.5000  |   0.0      |   0           |   q2
%    3    |  -39.2250  |   0.0      |   0           |   q3
%    4    |   0.0      |  10.9150   |   pi/2        |   q4
%    5    |   0.0      |   9.4650   |  -pi/2        |   q5
%    6    |   0.0      |   8.2300   |   0           |   q6
% ----------------------------------------------------
clear; clearvars; close all; % Clear all stale data
% ------------------------------------------------------
%        Simulation Variables
% ------------------------------------------------------
N = 200; % Number of frames captured
t = linspace(0, 2*pi, N); % linspace of time axis
%-------------------------------------------------------

% ------------------------------------------------------
%        Define Links According to DH Table
% ------------------------------------------------------
L1 = Link([0     8.9459   0.0     pi/2  0], 'standard');
L2 = Link([0     0.0     -42.5000  0     0], 'standard');
L3 = Link([0     0.0     -39.2250  0     0], 'standard');
L4 = Link([0    10.9150   0.0     pi/2  0], 'standard');
L5 = Link([0     9.4650   0.0    -pi/2  0], 'standard');
L6 = Link([0     8.2300   0.0      0     0], 'standard');
%-------------------------------------------------------

%-------------------------------------------------------
%        Define Robot as UR5 with Links (L1 - L6)
%-------------------------------------------------------
UR5 = SerialLink([L1 L2 L3 L4 L5 L6], 'name', 'UR5 (cm)');
%-------------------------------------------------------

%-------------------------------------------------------
%        Defining Joint Limits
%-------------------------------------------------------
joint_min = deg2rad([-360 -360 -360 -360 -360 -360]);
joint_max = deg2rad([ 360  360  360  360  360  360]);
UR5.qlim = [joint_min' joint_max'];
%-------------------------------------------------------

%-------------------------------------------------------
%        Define Sphere Variables
%-------------------------------------------------------
syms x y z
a = 55; % a is the x axis offset
b = 70; % b is the y axis offset
c = 0; % c is the z axis offset
radius_sphere = 15; % radius of the sphere
radius_circle = 5; % radius of the circular path
theta = pi/4; % Angle of rotation of plane about x axis
phi = 0;  % Angle of rotaion of plane about y axis
%-------------------------------------------------------
```

```
%-------------------------------------------------------
%        Calculate Offset for Plane
%        given radius of sphere and circular path
%-------------------------------------------------------
offset = sqrt(radius_sphere^2 - radius_circle^2);
%-------------------------------------------------------

%-------------------------------------------------------
%        Equation of a Sphere
%-------------------------------------------------------
s = (x-a)^2 + (y-b)^2 + (z - c)^2 - radius_sphere^2 ;
%-------------------------------------------------------

%-------------------------------------------------------
%       Define Negative Gradient of Sphere Equation
%-------------------------------------------------------
neg_grad = -gradient(s, [x,y,z]);
neg_grad_func = matlabFunction(neg_grad, 'Vars', [x,y,z]);
%-------------------------------------------------------

%-------------------------------------------------------
%          Calculate X, Y, and Z coordinates
%          using equation of a circle
%-------------------------------------------------------
x_2 = radius_circle * cos(t);
y_2 = radius_circle * sin(t);
z_2 = zeros(size(t));
%-------------------------------------------------------
```

```
%-----------------------------------------------------------
% Translate Trajectory Coordinates in Base Frame of Robot
%-----------------------------------------------------------

%-------------------
%       Define The offset
%       base frame -> center of sphere
%-------------------
H_1_0 = [eye(3) [a;b;c]; 0 0 0 1];
%-------------------
%       Calculate Rotations about x-axis and y-axis
%-------------------
r_x_t = [1 0 0; 0 cos(theta) -sin(theta); 0 sin(theta) cos(theta)];
r_y_p = [cos(phi) 0 -sin(phi); 0 1 0; sin(phi) 0 cos(phi)];
r_c = r_x_t * r_y_p;
%-------------------
%       Rotate center by R_x R_y
%-------------------
offset_axis = [0;0;offset];
circle_center = r_c * offset_axis;
%-------------------
%       Define The Circular Plane
%       wrt center of spher (H_2_1)
%-------------------
H_2_1 = [r_c circle_center; 0 0 0 1];  % 4x4
%-------------------
%       Construct Homogenous Transform of circular path
%-------------------
H_3_2 = [x_2; y_2; z_2; ones(1,length(t))];  % 4xN homogeneous
%-------------------
%       Calculate Homogenous Transform of frame 3 wrt frame 0
%-------------------
H_3_0 = H_1_0 * H_2_1 * H_3_2;  % 4xN
%-------------------
%       Extract coordinates
%-------------------
x_c = H_3_0(1,:);
y_c = H_3_0(2,:);
z_c = H_3_0(3,:);
%-------------------
%       Apply Tool offset transform
%-------------------
tool_offset = [0 0 8.23];      % tool offset along local z
T_tool = transl(tool_offset);  % 4x4 homogeneous matrix
UR5.tool = T_tool;
%-----------------------------------------------------------
```

```
% ----------------------
%       Prepare IK loop
% ----------------------
q0 = zeros(1,6);    % initial guess
q_all = zeros(N,6);
%-----------------------------------------------------------

% ----------------------
%       IK loop
% ----------------------
for i = 1:N
    % Position of desired point
    pos = [x_c(i); y_c(i); z_c(i)];

    % Compute local surface normal
    n = neg_grad_func(x_c(i), y_c(i), z_c(i));
    n = n / norm(n);  % normalize

    % Define orientation such that z-axis = n
    % You need to choose orthogonal x/y axes
    up = [0;0;1];
    if abs(dot(up,n)) > 0.9  % avoid singularity
        up = [0;1;0];
    end
    x_axis = cross(up,n); x_axis = x_axis/norm(x_axis);
    y_axis = cross(n,x_axis);
    R = [x_axis y_axis n];

    % Construct desired homogeneous transform
    T_desired = [R pos; 0 0 0 1];

    % Solve IK
    q_sol = UR5.ikcon(SE3(T_desired), q0);

    % Check joint limits
    q_sol = max(min(q_sol, joint_max), joint_min);

    % Store and update guess
    q_all(i,:) = q_sol;
    q0 = q_sol;
end
```

```
%-------------------------------------------------------
%       Plot results
%-------------------------------------------------------
[X,Y,Z] = sphere(20);
figure;
hold on; grid on;
axis equal;
xlim([-100 100]); ylim([-100 100]); zlim([-100 150]);
xlabel('X (cm)'); ylabel('Y (cm)'); zlabel('Z (cm)');
title('UR5 tracing a circular path on a sphere');

% Plot sphere
surf(radius_sphere*X + a, radius_sphere*Y + b, radius_sphere*Z + c, ...
     'FaceColor','cyan','EdgeColor','none','FaceAlpha',0.3);

% Plot desired circle
plot3(x_c, y_c, z_c, 'r-', 'LineWidth', 2);

% Plot normals
for k = 1:round(N/8):N
    g = neg_grad_func(x_c(k), y_c(k), z_c(k));
    g = g / norm(g);
    quiver3(x_c(k), y_c(k), z_c(k), g(1), g(2), g(3), 0.8, 'r', 'LineWidth', 1.2);
end

% Precompute FK trajectory
for i = 1:N
    T = UR5.fkine(q_all(i,:));
    ee_traj(i,:) = transl(T);
end
plot3(ee_traj(:,1), ee_traj(:,2), ee_traj(:,3), 'm--','LineWidth',2);

% Animate
for i = 1:N
    UR5.plot(q_all(i,:),'fps', 120);
    pause(0.03);
end

camlight; lighting gouraud;
```

# 4 Partner Contributions

| Item | Christopher Waschenko | Johan Gamboa |
|---|---|---|
| Report | X | |
| Math | X | X |
| Code | X | X |