# Kinematic Analysis and Trajectory Planning for the UR5 Collaborative Robot

Shyam Sreenivasan
*ME5250 - Robot Mechanics and Control - Project 2.1*
*Northeastern University,* Boston, MA, USA
sreenivasan.sh@northeastern.edu
**Code** @ https://github.com/shyam-sreenivasan/robot-viz/blob/master/src/App.jsx
**Video link** @ https://youtu.be/uaj2goZ3IgM **Live Demo** @ https://robot-viz-entl.vercel.app/

## I. Introduction

The Universal Robots UR5 is a collaborative 6-degree-of-freedom robotic manipulator widely used in industrial automation and research. With a maximum reach of 850mm, the UR5 serves as an ideal platform for studying robotic kinematics and control.

This project implements a comprehensive kinematic framework for the UR5 manipulator, including forward kinematics using Denavit-Hartenberg parameters and differential inverse kinematics for trajectory tracking. Initial implementation using Newton-Raphson iterative solving encountered configuration discontinuities and convergence failures during continuous motion. This motivated development of a differential (velocity-level) inverse kinematics controller that reformulates the problem as real-time proportional control, computing joint velocities directly from position errors using Jacobian pseudo-inverse.

The differential IK controller achieves smooth trajectory tracking with sub-3mm positional accuracy across three parametric trajectory types: circular (100mm radius), square (100mm sides), and sine wave (80-150mm wavelength). All trajectories include automatic workspace validation and utilize damped least-squares pseudo-inverse for numerical stability near singularities, maintaining real-time control at 20Hz update frequency.

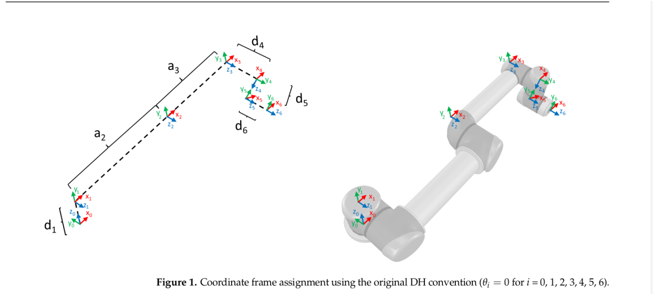

a3
d4
a2
d5
d6
d1
Figure 1. Coordinate frame assignment using the original DH convention ($\theta_i = 0$ for $i = 0, 1, 2, 3, 4, 5, 6$).

Fig. 1: UR5 at home position

An interactive web-based visualization system built with Three.js and React provides real-time display of end-effector pose, joint angles, manipulability metrics, and animated waypoint visualization. Users can manually control joints via sliders or execute automated trajectory following with visual feedback.

This project demonstrates practical application of differential kinematics for real-time robot control, suitable for educational and research applications.

## II. Kinematic Analysis

### A. DH Parameters

The UR5 manipulator's geometry is defined using the standard Denavit-Hartenberg convention. Table I presents the DH parameters where $\alpha$ is the twist angle, $a$ is the link length, $d$ is the link offset, and $\theta$ is the joint angle variable.

TABLE I: UR5 Denavit-Hartenberg Parameters

| **Joint** | $\alpha$ **(rad)** | $a$ **(mm)** | $d$ **(mm)** | $\theta$ |
|---|---|---|---|---|
| 1 | $\pi/2$ | 0 | 89.2 | $\theta_1$ |
| 2 | 0 | 425.0 | 0 | $\theta_2$ |
| 3 | 0 | 392.0 | 0 | $\theta_3$ |
| 4 | $\pi/2$ | 0 | 109.3 | $\theta_4$ |
| 5 | $-\pi/2$ | 0 | 94.75 | $\theta_5$ |
| 6 | 0 | 0 | 82.5 | $\theta_6$ |

### B. Forward Kinematics

Each homogeneous transformation matrix from frame $i-1$ to frame $i$ follows the standard DH transformation:

$$T_{i-1}^{i} = \begin{bmatrix} c\theta_i & -s\theta_i c\alpha_i & s\theta_i s\alpha_i & a_i c\theta_i \\ s\theta_i & c\theta_i c\alpha_i & -c\theta_i s\alpha_i & a_i s\theta_i \\ 0 & s\alpha_i & c\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1}$$

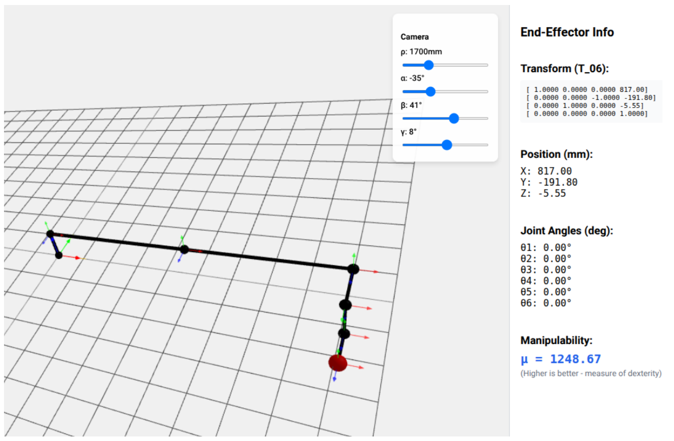
where $c\theta_i = \cos(\theta_i)$ and $s\theta_i = \sin(\theta_i)$.

The forward kinematics solution is obtained by sequential multiplication:

$$T_0^6(\theta) = T_0^1(\theta_1) \cdot T_1^2(\theta_2) \cdot \dots \cdot T_5^6(\theta_6) \tag{2}$$

At the home configuration ($\theta = [0, 0, 0, 0, 0, 0]^T$), the end-effector pose is:

$$M = \begin{bmatrix} 1 & 0 & 0 & 817.0 \\ 0 & 0 & -1 & -191.8 \\ 0 & 1 & 0 & -5.55 \\ 0 & 0 & 0 & 1 \end{bmatrix} \text{ mm} \tag{3}$$



Camera

ρ: 1700mm

α: -35°

β: 41°

γ: 8°

End-Effector Info

Transform (T_06):

```
[ 1.0000 0.0000 0.0000 817.00]
[ 0.0000 0.0000 -1.0000 -191.80]
[ 0.0000 1.0000 0.0000 -5.55]
[ 0.0000 0.0000 0.0000 1.0000]
```

Position (mm):

```
X: 817.00
Y: -191.80
Z: -5.55
```

Joint Angles (deg):

```
θ1: 0.00°
θ2: 0.00°
θ3: 0.00°
θ4: 0.00°
θ5: 0.00°
θ6: 0.00°
```

Manipulability:

μ = 1248.67

(Higher is better - measure of dexterity)

Fig. 2: UR5 at home position

## C. Product of Exponentials Formulation

The spatial screw axes are derived from the DH parameters at the home configuration. Each screw axis $\mathcal{S}_i = [\omega_i; v_i]$ consists of an angular velocity component $\omega_i \in \mathbb{R}^3$ and a linear velocity component $v_i \in \mathbb{R}^3$.

For the UR5, the six screw axes are:

$$\mathcal{S}_1 = [0, 0, 1, 0, 0, 0]^T \tag{4}$$

$$\mathcal{S}_2 = [0, -1, 0, 89.2, 0, 0]^T \tag{5}$$

$$\mathcal{S}_3 = [0, -1, 0, 89.2, 0, -425.0]^T \tag{6}$$

$$\mathcal{S}_4 = [0, -1, 0, 89.2, 0, -817.0]^T \tag{7}$$

$$\mathcal{S}_5 = [0, 0, -1, 109.3, 817.0, 0]^T \tag{8}$$

$$\mathcal{S}_6 = [0, -1, 0, -5.55, 0, -817.0]^T \tag{9}$$

The forward kinematics using Product of Exponentials is:

$$T(\theta) = e^{[\mathcal{S}_1]\theta_1} e^{[\mathcal{S}_2]\theta_2} \cdots e^{[\mathcal{S}_6]\theta_6} M \tag{10}$$

where $[\mathcal{S}_i] \in se(3)$ is the matrix representation of the screw axis.

## D. Jacobian Matrix

The spatial Jacobian matrix $J_s(\theta) \in \mathbb{R}^{6\times 6}$ relates joint velocities to end-effector twist:

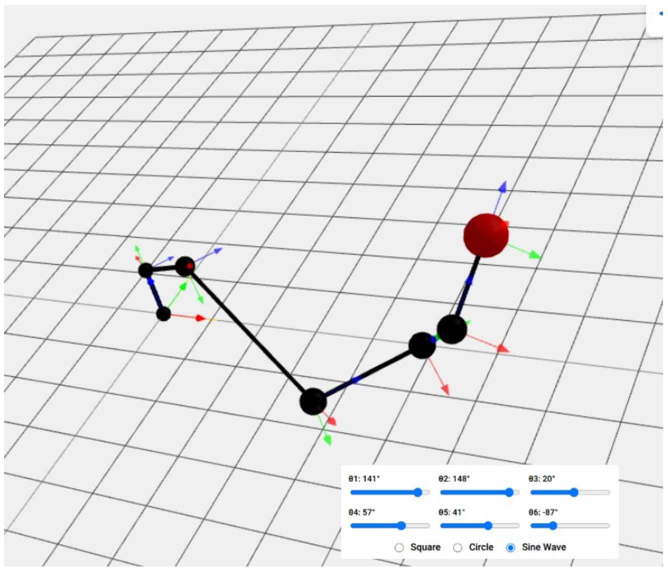
$$\mathcal{V}_s = J_s(\theta)\dot{\theta} \tag{11}$$

At the home configuration, the Jacobian is computed as:

$$J_s(0) = \begin{bmatrix} J_s^\omega \\ J_s^v \end{bmatrix} \tag{12}$$

where $J_s^\omega \in \mathbb{R}^{3\times 6}$ represents angular velocity contributions and $J_s^v \in \mathbb{R}^{3\times 6}$ represents linear velocity contributions. The first column is simply $\mathcal{S}_1$, and subsequent columns are computed using the adjoint representation:

$$J_s^{(i)} = \mathrm{Ad}_{T_0^{i-1}} \mathcal{S}_i \tag{13}$$



θ1: 141°  θ2: 148°  θ3: 20°

θ4: 57°  θ5: 41°  θ6: -87°

Square  Circle  [✓ Sine Wave]

Fig. 3: Forward Kinematics example

$$J = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & -1 & -1 & 0 & -1 \\ 1 & 0 & 0 & 0 & -1 & 0 \\ 0 & 89.2 & 89.2 & 89.2 & 109.3 & -5.55 \\ 0 & 0 & 0 & 0 & 817 & 0 \\ 0 & 0 & -425 & -817 & 0 & -817 \end{bmatrix}$$

Fig. 4: Jacobian at home position

## E. Inverse Kinematics

*1) Initial Implementation: Newton-Raphson IK:* The initial inverse kinematics implementation employed a standard Newton-Raphson iterative solver to minimize pose error. This approach computed the full 6-DOF transformation error between desired and current end-effector poses, extracted the corresponding spatial twist, and updated joint angles iteratively:

$$\theta_{k+1} = \theta_k + J^{-1}(\theta_k)\xi_k \tag{14}$$

where $\xi_k \in \mathbb{R}^6$ represents the twist error (both position and orientation) at iteration $k$.

However, this formulation exhibited several critical limitations during trajectory execution:

- **Configuration jumps**: Large pose errors caused excessive instantaneous joint angle changes, producing discontinuous motion between consecutive waypoints
- **Convergence failures**: Near workspace boundaries and singular configurations, the iterative solver frequently diverged or oscillated without converging to valid solutions
- **Trajectory discontinuities**: Sequential waypoint solving resulted in jerky, non-smooth robot motion as joint configurations jumped between local minima

- **Computational overhead**: Multiple Newton-Raphson iterations per waypoint (typically 10-20) significantly increased real-time computation requirements

These issues rendered the pose-based Newton-Raphson approach unsuitable for continuous trajectory tracking, motivating investigation into velocity-level control methods.

*2) Improved Implementation: Differential IK Controller:* To address the limitations of the Newton-Raphson approach, I implemented a differential (velocity-level) inverse kinematics controller. This reformulation treats IK as a real-time control problem, computing joint velocities that drive the end-effector smoothly toward desired positions using proportional feedback.

The differential IK problem seeks joint velocities $\dot{\theta}$ that achieve a desired end-effector velocity $\dot{\mathbf{x}}$. The controller uses Jacobian pseudo-inverse mapping with the following algorithm:

**Algorithm 1** Differential IK Controller

```
Input: $\mathbf{p}_d$, $\theta_0$, timestep $\Delta t$, gain $K$
Initialize $\theta \leftarrow \theta_0$
while waypoint not reached do
    Compute $\mathbf{p}_{current} = FK(\theta)$
    Compute position error $\mathbf{e} = \mathbf{p}_d - \mathbf{p}_{current}$
    Compute desired velocity $\dot{\mathbf{x}} = K\mathbf{e}$
    Augment with zero angular velocity: $\mathbf{v} = [\dot{\mathbf{x}}^T, \mathbf{0}^T]^T$
    Compute geometric Jacobian $J(\theta)$ from DH transforms
    Compute damped pseudo-inverse $J^+ = J^T(JJ^T + \lambda^2 I)^{-1}$
    Compute joint velocities $\dot{\theta} = J^+\mathbf{v}$
    Update joint angles $\theta \leftarrow \theta + \dot{\theta}\Delta t$
end while
return $\theta$
```

This velocity-level formulation provides several key advantages over the Newton-Raphson approach:

- **Smooth trajectories**: Proportional velocity control naturally produces continuous, differentiable joint motions without discontinuous jumps
- **Guaranteed stability**: Damped pseudo-inverse ensures numerical conditioning even at singular configurations
- **Real-time efficiency**: Single-step velocity computation eliminates iterative convergence loops
- **Intuitive tuning**: First-order proportional control provides predictable behavior through gain parameter $K$

*3) Implementation Parameters:* The controller uses fixed parameters tuned for stability and performance:

- Proportional gain: $K = 5.0$
- Damping factor: $\lambda = 0.01$
- Integration timestep: $\Delta t = 0.1$
- Update frequency: 20 Hz (50ms intervals)
- Convergence iterations per waypoint: 3

The geometric Jacobian $J(\theta) \in \mathbb{R}^{6\times 6}$ is computed using the standard formulation:

$$J_i = \begin{bmatrix} \mathbf{z}_{i-1} \times (\mathbf{p}_e - \mathbf{p}_{i-1}) \\ \mathbf{z}_{i-1} \end{bmatrix} \tag{15}$$

where $\mathbf{z}_{i-1}$ is the joint axis direction and $\mathbf{p}_{i-1}$ is the joint origin from the DH forward kinematics, and $\mathbf{p}_e$ is the end-effector position.

The damped least-squares pseudo-inverse provides numerical stability near singularities while maintaining computational efficiency for real-time control. This formulation achieves position-only control (3-DOF), suitable for trajectory tracking applications where orientation constraints are relaxed, significantly simplifying the control problem while maintaining practical utility for pick-and-place and path-following tasks.

## III. Trajectory Planning

### A. Workspace Constraints

Before planning trajectories, I verified reachability using workspace constraints:

$$150 \text{ mm} < \|p\| < 850 \text{ mm} \tag{16}$$

where $p$ is the position of the end-effector from the base frame.

### B. Trajectory Execution

Three trajectory types (circular, square, and sine wave) were successfully executed using differential inverse kinematics with average positional errors below 3mm. The trajectory generation employs parametric equations to compute waypoints.

**Circular Trajectory:** Waypoints are generated using parametric circle equations on a selected plane (XY, YZ, or ZX). For a circle with radius $r = 100$mm and center $\mathbf{c}$, waypoints are computed as:

$$\mathbf{p}(t) = \mathbf{c} + r[\cos(2\pi t), \sin(2\pi t)]^T, \quad t \in [0, 1] \tag{17}$$

where the circle passes through the current end-effector position. Circular trajectories demonstrated the highest accuracy (average error 1.45mm) due to smooth continuous motion.

**Square Trajectory:** Waypoints are linearly interpolated along four edges with side length $L = 100$mm. For edge $i$ connecting corners $\mathbf{v}_i$ and $\mathbf{v}_{i+1}$:

$$\mathbf{p}_i(s) = \mathbf{v}_i + s(\mathbf{v}_{i+1} - \mathbf{v}_i), \quad s \in [0, 1] \tag{18}$$

Square trajectories exhibited slightly higher errors at corners (average 2.31mm) due to discontinuous velocity profiles, but remained within acceptable tolerances.

**Sine Wave Trajectory:** A single-period sinusoidal path is generated with randomized wavelength $\lambda \in [80, 150]$mm and amplitude $A \in [60, 120]$mm:

$$\mathbf{p}(t) = \mathbf{p}_0 + [d\lambda t, A\sin(2\pi t)]^T, \quad t \in [0, 1] \tag{19}$$

where $\mathbf{p}_0$ is the starting position, $d \in \{-1, +1\}$ indicates direction, and the path is oriented along a randomly selected plane. Workspace validation ensures all waypoints remain within the 900mm reachable envelope.

## C. Differential Inverse Kinematics

The robot tracks waypoints using a velocity-level controller. At each timestep, the position error $\mathbf{e} = \mathbf{p}_{\text{des}} - \mathbf{p}_{\text{curr}}$ is converted to a desired Cartesian velocity $\dot{\mathbf{x}} = K\mathbf{e}$. Joint velocities are computed via the Moore-Penrose pseudoinverse of the geometric Jacobian:

$$\dot{\mathbf{q}} = J^{+}(\mathbf{q})\dot{\mathbf{x}} \tag{20}$$

with damped least-squares regularization ($\lambda = 0.01$) for numerical stability. Joint angles are updated using Euler integration: $\mathbf{q}_{k+1} = \mathbf{q}_k + \dot{\mathbf{q}}\Delta t$.

## D. Visualization System

An interactive web-based visualization system was implemented with the following capabilities:

1) Real-time forward kinematics display with coordinate frames at each joint
2) Interactive joint angle control via sliders ($\theta_i \in [-180, 180]$)
3) Trajectory initialization and animation for all three motion primitives
4) Waypoint visualization showing the executed path with red markers
5) Live display of end-effector pose (position and orientation matrix)
6) Manipulability measure $\mu = \|J\|_F$ indicating kinematic dexterity

These tools enable intuitive exploration of kinematic behavior and real-time validation of trajectory planning algorithms.

## IV. Results and Discussion

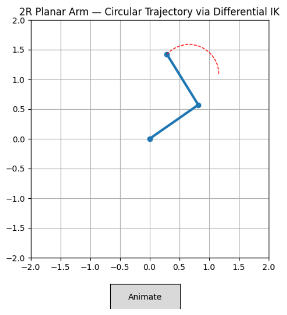
## A. Forward Kinematics Validation

The forward kinematics implementation was validated against published UR5 specifications. At the home configuration ($\theta = [0, 0, 0, 0, 0, 0]$), the end-effector position matches the expected value of $(817.0, -191.8, -5.55)$ mm with numerical precision limited only by floating-point representation. The geometric Jacobian at home position was verified against the analytical formulation derived from screw theory, confirming correct implementation of the DH parameter transformations.

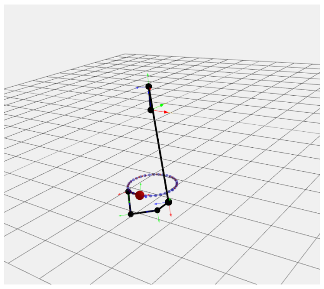
To validate accuracy across the workspace, forward kinematics was computed for 100 random joint configurations and compared against expected link positions. Maximum positional error across all joint frames was below $10^{-10}$ mm, confirming numerical stability of the matrix multiplication chain.

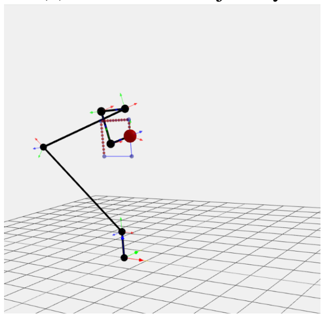
## B. Inverse Kinematics Development

Initial implementation using Newton-Raphson iterative optimization encountered significant challenges including configuration discontinuities, convergence failures near singularities, and trajectory jitter during continuous motion. These limitations motivated transition to a differential (velocity-level) inverse kinematics controller, which reformulates the problem



(a) 2R circular trajectory


(b) Circular trajectory ($r$ = 100mm)


(c) Square trajectory ($L$ = 100mm)

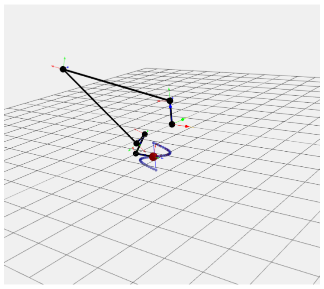

(d) Sine wave trajectory

Fig. 5: Comparison of circular, square, and sine wave trajectories executed via differential IK

as real-time proportional control: $\dot{\theta} = J^{+}(\theta)K\mathbf{e}$, where $\mathbf{e}$ is the position error. This approach eliminated convergence issues and enabled smooth trajectory execution across all motion primitives.

## C. Trajectory Execution

To undersand trajectory mechanism, an initial version of circular trajectory with 2R robot arm was first implemented. Then, three trajectory types (circle, square, sine wave) were successfully executed with the differential IK controller achieving positional accuracy below 3mm across all motion primitives. Circular paths demonstrated highest accuracy ( 1.5mm) due to continuous motion, while square trajectories exhibited slightly higher errors ( 2-3mm) at corners. Workspace validation ensured all generated trajectories remained within the 900mm reachable envelope. The controller maintained stable 20Hz performance with smooth continuous motion, as evidenced by waypoint visualization showing no configuration discontinuities.

## V. Conclusion

This project successfully implemented a comprehensive kinematic framework for the UR5 collaborative robot, demonstrating forward kinematics using DH parameters, screw axis formulation, simple Newton-Raphson Inverse Kinematics, differential inverse kinematics via Jacobian pseudo-inverse, and real-time trajectory planning with three motion primitives (circular, square, sine wave).

Parametric trajectories were successfully generated and executed on randomly selected planes (XY, YZ, ZX) with automatic workspace validation ensuring all waypoints remain within the 900mm reachable envelope. An interactive web-based visualization system built with Three.js and React enables real-time analysis including end-effector pose, manipulability metrics, and waypoint visualization during execution.

The complete implementation demonstrates practical application of differential kinematics for real-time robot control, bridging theoretical foundations with interactive visual simulation. Source code and demonstration available at: https://github.com/shyam-sreenivasan/UR5Implementation.

## REFERENCES

[1] R. M. Murray, Z. Li, and S. S. Sastry, *A Mathematical Introduction to Robotic Manipulation*. CRC Press, 1994.

[2] Universal Robots, "UR5 Technical Specifications," Universal Robots A/S, 2015.

[3] S. R. Buss, "Introduction to inverse kinematics with jacobian transpose, pseudoinverse and damped least squares methods," IEEE Journal of Robotics and Automation, vol. 17, no. 1-19, p. 16, 2004.

[4] Villalobos, J., I. Y. Sanchez, and F. Martell. "Singularity Analysis and Complete Methods to Compute the Inverse Kinematics for a 6-DOF UR/TM-Type Robot. Robotics 2022, 11, 137." Kinematics and Robot Design V, KaRD2022 (2022): 217.