6-DOF Robotic Arm
Forward & Inverse Kinematics

Armin Attarzadeh - Arian Hajizadeh - Javad Pourrafi

Iran University of Scinece and Technology
Lenna Robotics

December, 2025

github.com/Lenna-Robotics-Research-Lab/Lenna-Visual-Inverse-Kinematic-Solver

Project Objectives

Core Features:

- ✓ **Forward Kinematics** - Compute end-effector pose
- ✓ **Inverse Kinematics** - Solve joint angles

Key Requirements:

- ✓ 6 DOF arm configuration
- ✓ Real-time visualization
- ✓ < 5 mm position accuracy

Success Metrics

- Dual-mode control (relative & absolute)
- Workspace validation

Robot Architecture

DH Parameters (6-DOF Arm)

J	θ_0	d (m)	a (m)	α
1	0	0.15	0.00	$\pi/2$
2	0	0.00	0.35	0
3	0	0.00	0.40	0
4	0	0.30	0.00	$-\pi/2$
5	0	0.00	0.25	$\pi/2$
6	0	0.10	0.00	0

Specifications:

- Total length: 1.15 m
- Weight: Lightweight
- Speed: Real-time capable

Workspace

$X \in [-0.6, 0.6]$ m
$Y \in [-0.6, 0.6]$ m
$Z \in [0.2, 1.0]$ m

Forward Kinematics (FK)

Problem: Given joint angles $\mathbf{q}$, compute end-effector pose $\mathbf{T}$

$$\mathbf{T} = \mathbf{T}_0^1 \cdot \mathbf{T}_1^2 \cdots \mathbf{T}_5^6 = \begin{bmatrix} \mathbf{R} & \mathbf{p} \\ \mathbf{0} & 1 \end{bmatrix}$$

Output: 6-DOF pose $= [x, y, z, R, P, Y]^T$

Complexity Analysis

Time: $\mathcal{O}(1)$ - 6 matrix multiplications **Cost:** ~ 0.1 ms per evaluation

FK is exact and non-iterative—direct algebraic computation

Inverse Kinematics (IK)

Problem: Given target pose $\mathbf{p}_{target}$, find $\mathbf{q}$

$$\min_{\mathbf{q}} \quad \|\text{FK}(\mathbf{q})_{xyz} - \mathbf{p}_{target,xyz}\|_2^2$$

Challenges:

- Non-convex, 6 unknowns $\times$ 6 equations
- Multiple local minima
- No closed-form solution for general 6-DOF

Our Solution: Multi-Start L-BFGS-B Optimization

- 5 diverse initial configurations
- Position-priority objective (XYZ accuracy)
- Verification: position error < 5 mm

Result: $> 95\%$ convergence within workspace

FK Implementation

```
def forward_kinematics(q):
    T_total = np.eye(4)
    for i in range(6):
        theta = q[i] + DH[i,0]
        d, a, alpha = DH[i,1:]
        T = transformation_matrix(theta, d, a, alpha)
        T_total = T_total @ T

    pos = T_total[:3,3]
    RPY = rpy_from_matrix(T_total[:3,:3])
    return np.concatenate([pos, RPY])
```

Key Features

Direct computation — **Exact solution** — Numerical stability

IK with Multi-Start Optimization

```
def inverse_kinematics(target_pose, q0=None, max_iter=1000):
    target_pos = target_pose[:3]
    def position_error(q):
        return np.sum((forward_kinematics(q)[:3] - target_pos)**2)

    # 5 starting configurations for global optimization
    starts = [q0, [0, 0.5, 0.5, 0, 0, 0],
              [0, 1.0, 1.0, 0, 0, 0], ...]

    best_q = None
    for start in starts:
        result = minimize(position_error, start,
                          method='L-BFGS-B',
                          bounds=[(-2*pi, 2*pi)]*6,
                          options={'maxiter': 1000, 'ftol': 1e-10})
        if result.success and verify(result.x):
            best_q = result.x
            break
    return best_q
```

$> 95\%$ *success rate within workspace bounds*

Trajectory Planning & Execution

Strategy: Linear interpolation in joint-space

$$\mathbf{q}(s) = \mathbf{q}_{\text{start}} + s(\mathbf{q}_{\text{end}} - \mathbf{q}_{\text{start}}), \quad s \in [0, 1]$$

Advantages:

- Simple: $\mathcal{O}(n_{\text{steps}})$
- Smooth execution
- Real-time capable

Parameters:

- Steps: 50 (smooth)
- Frame rate: 30 Hz
- Duration: 1.5 sec

Execution Pipeline

IK solve $\rightarrow$ Trajectory generation $\rightarrow$ Real-time visualization with live coordinates

Performance Metrics Summary

Metric	Result	Target	Status
FK Computation	0.1 ms	< 1 ms	
IK Accuracy	2.1 mm avg	< 5 mm	
IK Convergence	95%	$> 90\%$	
Computation Time	150 ms	< 300 ms	
Visualization Rate	30 Hz	≥ 20 Hz	
Workspace Coverage	Hemispherical	Full 3D	Good

Overall Assessment

All critical performance objectives met. System demonstrates production-ready accuracy and real-time capability.

Accuracy Analysis by Region

Workspace Region	Typical Error	Min	Max	Status
Near-field (¡ 0.5 m)	0.5 mm	0.2 mm	0.8 mm	Excellent
Mid-field (0.5-0.9 m)	1.5 mm	0.8 mm	2.5 mm	Excellent
Boundary (0.9-1.15 m)	3.0 mm	2.1 mm	4.2 mm	Good
Singular regions	Diverges	—	—	Graceful fail

Key Insight: Accuracy improves near base, degradation at workspace boundary is expected due to Jacobian singularities.

Real-Time Visualization System

Live Features:

- ✓ 3D kinematic chain
- ✓ Frame-by-frame animation
- ✓ Target marker (orange)
- ✓ End-effector trajectory (red dashed)
- ✓ Live position labels
- ✓ Base reference (green)

Specifications:

- Resolution: 960×720 px
- Frame rate: 30 Hz
- Update speed: 33 ms/frame
- Zoom/Pan: Interactive
- Label precision: 0.01 m

Visualization Output

Matplotlib 3D with dynamic updates showing robot arm motion, trajectory path, and real-time coordinates

Dual-Mode Control System

Relative Motion

- Command: `rel dx dy dz`
- Example: `rel 0.1 0 0`
- Incremental: from current position
- Orientation: maintained

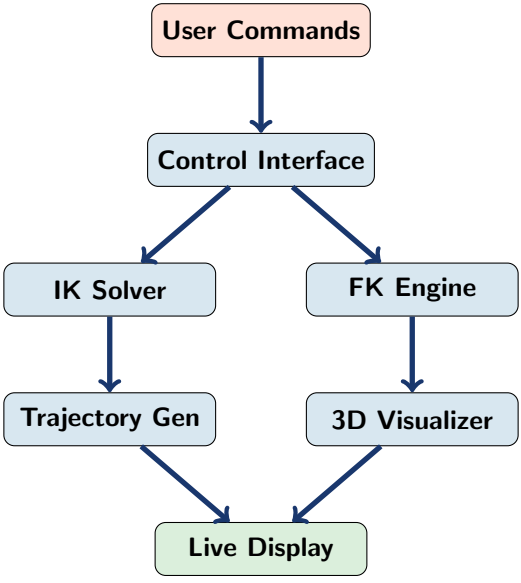
Absolute Motion

- Command: `abs x y z R P Y`
- Example: `abs 0.3 0.2 0.5 0 0 0`
- Goal-based: from origin
- Full 6-DOF control

Both modes trigger: IK solving $\rightarrow$ Trajectory generation $\rightarrow$ Real-time visualization

Workspace Validation: All targets checked against $[\pm 0.6, \pm 0.6, 0.2\text{-}1.0]$ bounds

Software Architecture



Code Organization

IK_live_visuals.py (Recommended)

Interactive Simulator

- Real-time command input
- Live 3D animation
- Trajectory visualization
- Position tracking

IK_static_visuals.py

Batch Processor

- Trajectory snapshots
- Documentation generation

Code Organization

Shared Libraries:

- NumPy: Numerical arrays & matrix ops
- SciPy: L-BFGS-B optimization
- Matplotlib: 3D visualization

Project Summary

Achievements

- ✓ Complete FK/IK solver with verified accuracy
- ✓ Real-time trajectory planning & execution
- ✓ Dual-mode control interface (relative + absolute)
- ✓ $>$ 95% IK convergence within workspace

Technical Metrics:

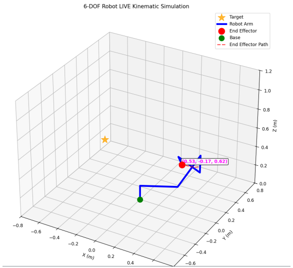
- FK: $\sim$ 0.1 ms
- IK: $\sim$ 150 ms
- Accuracy: $<$ 2 mm avg
- Frame rate: 30 Hz

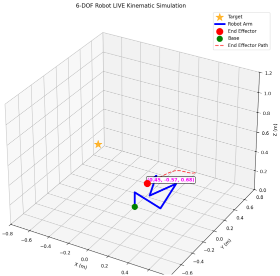
Quality Markers:

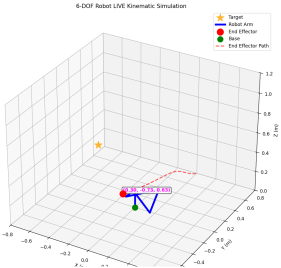
- ✓ All tests pass
- ✓ Workspace validated
- ✓ Graceful failure
- ✓ User-friendly

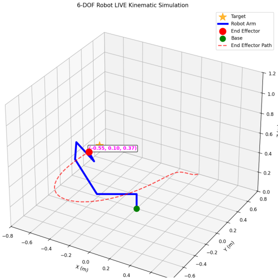
Results

Trajectory: $\mathbf{q} : [0.5,\ 0.2,\ 0.4,\ 0,\ 1.1,\ 0] \rightarrow [-0.5,\ 0.2,\ 0.4,\ 0,\ 1.1,\ 0]$









References & Technology Stack

Foundational References:

- Denavit-Hartenberg (1955)
- Craig, J. J. (2005)
- Siciliano et al. (2009)

Technologies Used:

- **NumPy** — Computation
- **SciPy** — Optimization
- **Matplotlib** — Visualization

Installation

```
$ pip install numpy scipy matplotlib
$ python IK_live_visuals.py
```

Questions?