THESIS DEFENSE

Dynamic Pick-and-Place System for a Manipulator on a Quadruped Using Object Detection

Abhishek Jagadeesh Kasaragod
Master of Science in Mechanical Engineering
University of Illinois at Chicago, 2024

Thesis Committee:
Dr. Pranav Bhounsule, Chair and Advisor
Dr. Michael Scott, Department of Mechanical Engineering
Dr. Jonathan Komperda, Department of Mechanical Engineering

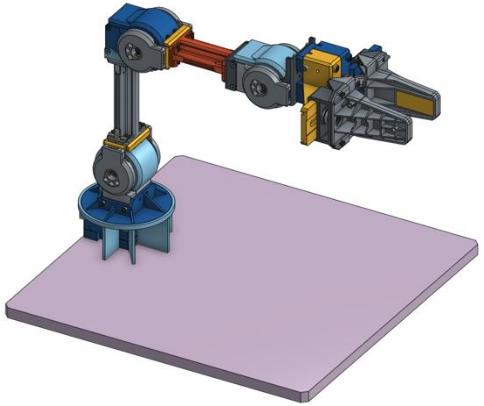
INTRODUCTION

- Development in Robotics
  - AI and machine learning
  - Sensor technology
  - Efficient and powerful robotic actuators

- Applications in various Fields
  - Agriculture, Military, Medicine
  - Collaborative Robots (cobots)
  - Drones for commercial and industrial applications

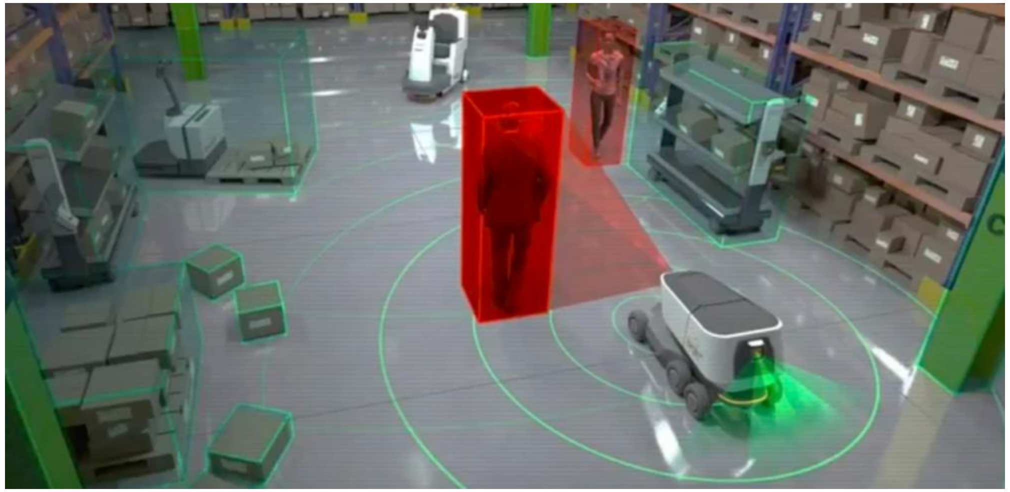
INTRODUCTION

- Importance of Mobile Manipulation
  - Enhanced Flexibility and Reach
  - Autonomous Operations
  - Versatility in Applications
- Challenges
  - Control and Coordination
  - Manipulation in Unstructured Environments
  - Energy Efficiency



OpenManipulatorX

WidowX 250s



Navigation Difficulties in Autonomous Robotics

INITIAL WORK

OpenManipulatorX

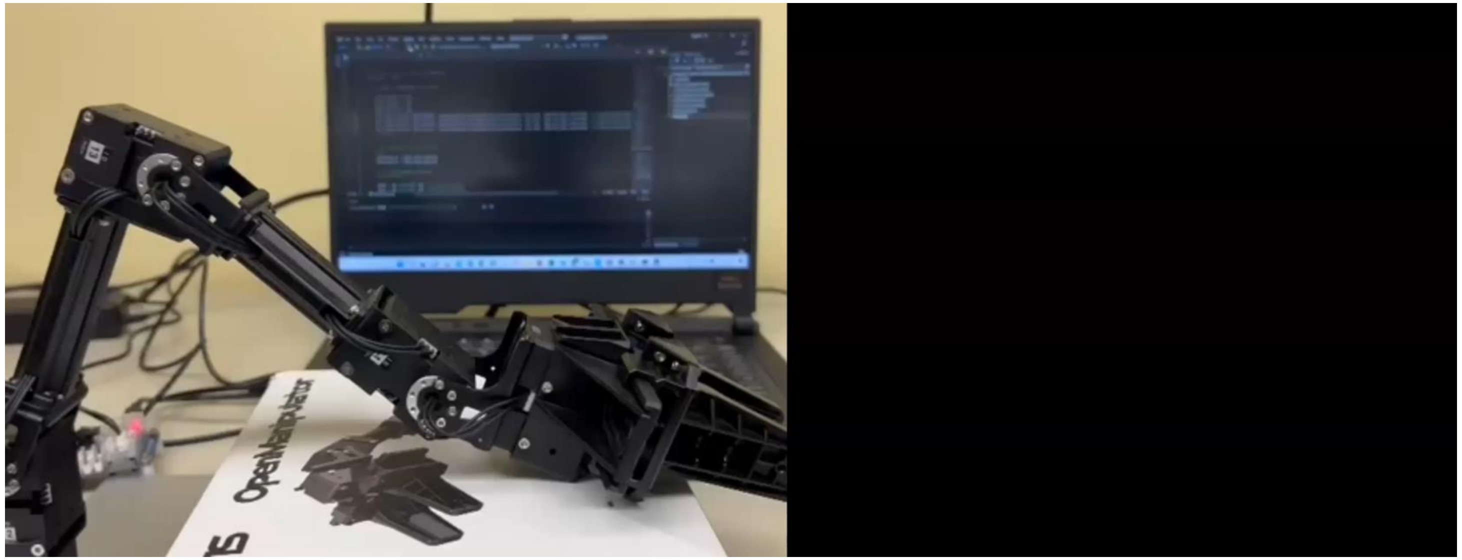
- Forward Kinematics (FK) & Inverse Kinematics (IK)
- High Level Control

Challenges Faced:

- Less Reach
- Improving motion
- Mobile object detection limitations
- Compatibility Issues

Object Detection

- cvlib python library
- PD control-based tracking
- Pixel-to-real distance scaling



https://github.com/arunponnusamy/cvlib

INITIAL WORK

- Motivation
  - Enable mobile applications and custom object detection
  - Improve robotic accuracy and reliability

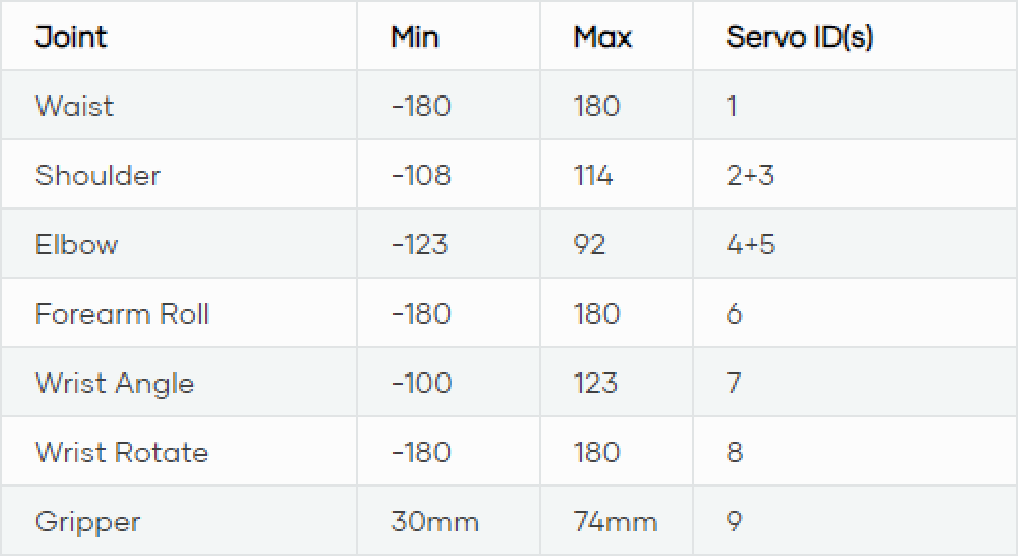
- Objectives
  - Develop robust FK and IK
  - Depth camera integration
  - Mobile platform implementation
  - Train custom object detection models

Robotic Arm Setup

- Specifications
  - Arm used - WidowX 250S by Trossen Robotics
  - Degrees of Freedom: 6 Degrees of Freedom (DOF)
  - Payload Capacity: Up to 250 grams
  - Reach: approx. 650 mm
  - Dynamixel Motors used: Seven XM430-W350 & two XL430-W250

XM430-W350

XL430-W250

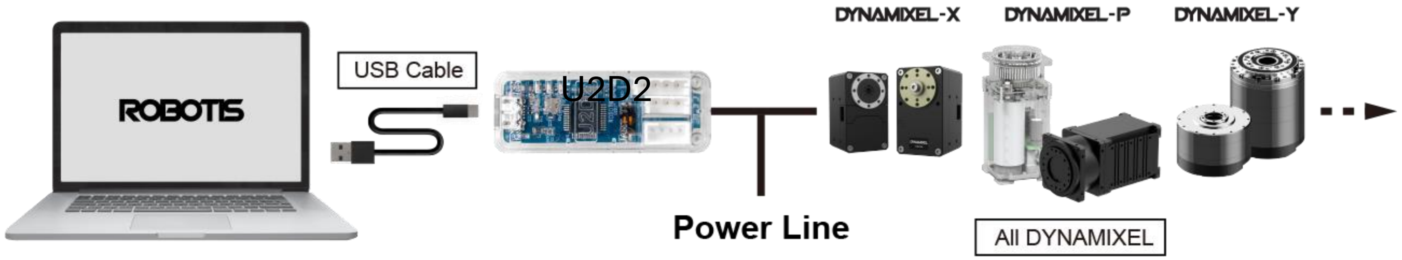


Joint	Min	Max	Servo ID(s)
Waist	-180	180	1
Shoulder	-108	114	2+3
Elbow	-123	92	4+5
Forearm Roll	-180	180	6
Wrist Angle	-100	123	7
Wrist Rotate	-180	180	8
Gripper	30mm	74mm	9

Joint Limits

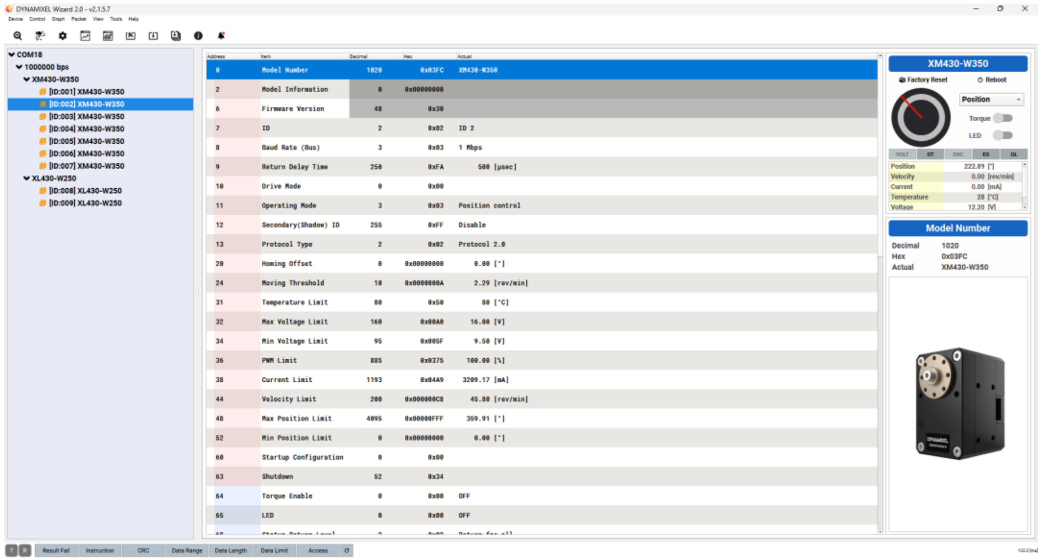
Robotic Arm Setup

- Control System
  - U2D2 Microcontroller
  - Dynamixel SDK for motor control
  - Programmed in Python



PC to DYNAMIXEL

- Dynamixel Wizard 2.0
  - Configuring the motors
  - Tuning, real-time monitoring and firmware updates
  - Setup of control parameters



Address	Item	Decimal	Hex	Actual
0	Model Number	1020	0x03FC	XM430-W350
2	Model Information	0	0x00000000	
6	Firmware Version	48	0x30	
7	ID	2	0x02	ID 2
8	Baud Rate (Bus)	3	0x03	1 Mbps
9	Return Delay Time	250	0xFA	500 [μsec]
10	Drive Mode	0	0x00	
11	Operating Mode	3	0x03	Position control
12	Secondary(Shadow) ID	255	0xFF	Disable
13	Protocol Type	2	0x02	Protocol 2.0
20	Homing Offset	0	0x00000000	0.00 [°]
24	Moving Threshold	10	0x0000000A	2.29 [rev/min]
31	Temperature Limit	80	0x50	80 [°C]
32	Max Voltage Limit	160	0x00A0	16.00 [V]
34	Min Voltage Limit	95	0x005F	9.50 [V]
36	PWM Limit	885	0x0375	100.00 [%]
38	Current Limit	1193	0x04A9	3209.17 [mA]
44	Velocity Limit	200	0x000000C8	45.80 [rev/min]
48	Max Position Limit	4095	0x00000FFF	359.91 [°]
52	Min Position Limit	0	0x00000000	0.00 [°]
60	Startup Configuration	0	0x00	
63	Shutdown	52	0x34	
64	Torque Enable	0	0x00	OFF
65	LED	0	0x00	OFF

Position

Position	222.89	[°]
Velocity	0.00	[rev/min]
Current	0.00	[mA]
Temperature	28	[°C]
Voltage	12.20	[V]

Decimal 1020
Hex 0x03FC
Actual XM430-W350

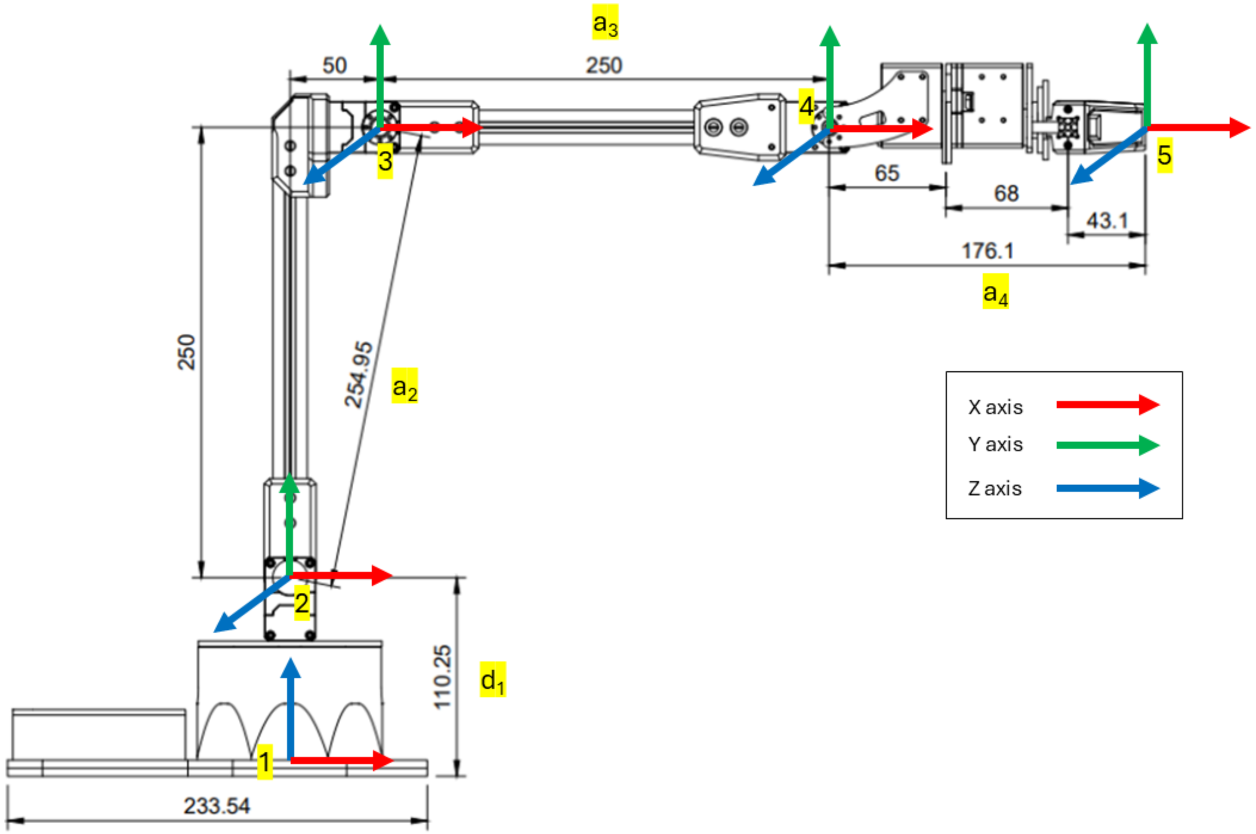
Dynamixel Wizard 2.0

Kinematics Modelling - Forward Kinematics

- Denavit–Hartenberg (DH) Parameters
- 4 DOF was considered
- Defined parameters: link length (a_i), link twist (α_i), link offset (d_i), and joint angle (θ_i), β = 11.537˚(offset angle)

DH Table

Link	$a_i\ (m)$	$\alpha_i\ (°)$	$d_i\ (m)$	$\theta_i\ (°)$
1	0	90	0.11025	θ_1
2	0.25495	0	0	θ_2 - β
3	0.25	0	0	θ_3 + β
4	0.17415	0	0	θ_4



Configuration of Robot Arm

Kinematics Modelling - Forward Kinematics

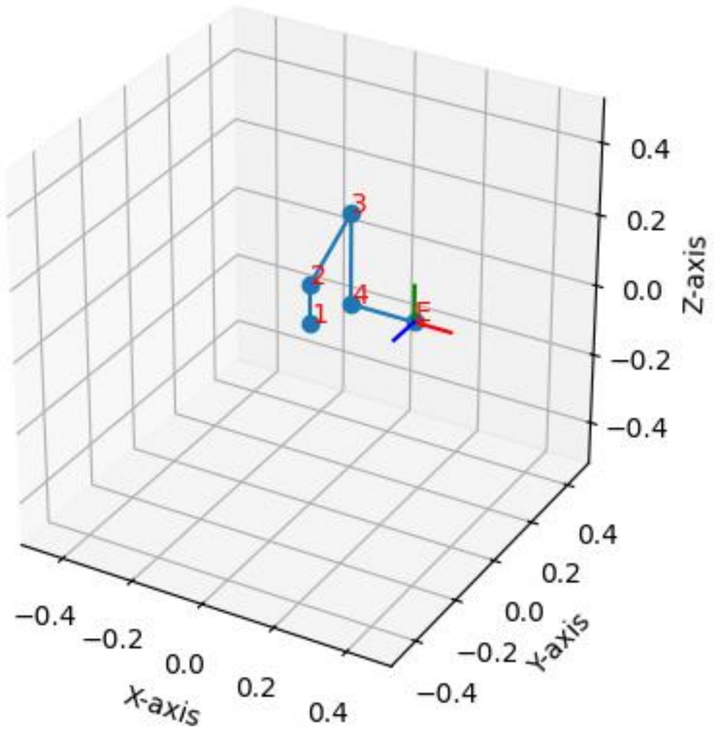
- H_i^{i-1} describes the position and orientation of joint i with respect to joint $i-1$

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

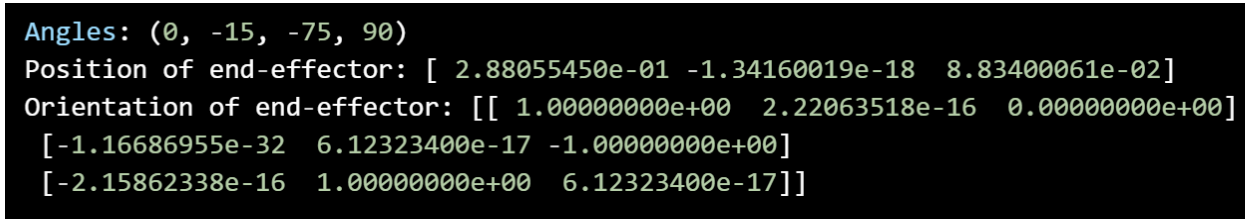
Where $s\theta_i = \sin\theta_i$, $c\theta_i = \cos\theta_i$, $s\alpha_i = \sin\alpha_i$, $c\alpha_i = \cos\alpha_i$.

- The position and orientation of the end−effector is found using the formula:

$$H_4^0 = H_1^0\, H_2^1\, H_3^2\, H_4^3 = \begin{bmatrix} R_4^0 & d_4^0 \\ 0 & 1 \end{bmatrix}$$



Initial position configuration



```
Angles: (0, -15, -75, 90)
Position of end-effector: [ 2.88055450e-01 -1.34160019e-18  8.83400061e-02]
Orientation of end-effector: [[ 1.00000000e+00  2.22063518e-16  0.00000000e+00]
 [-1.16686955e-32  6.12323400e-17 -1.00000000e+00]
 [-2.15862338e-16  1.00000000e+00  6.12323400e-17]]
```

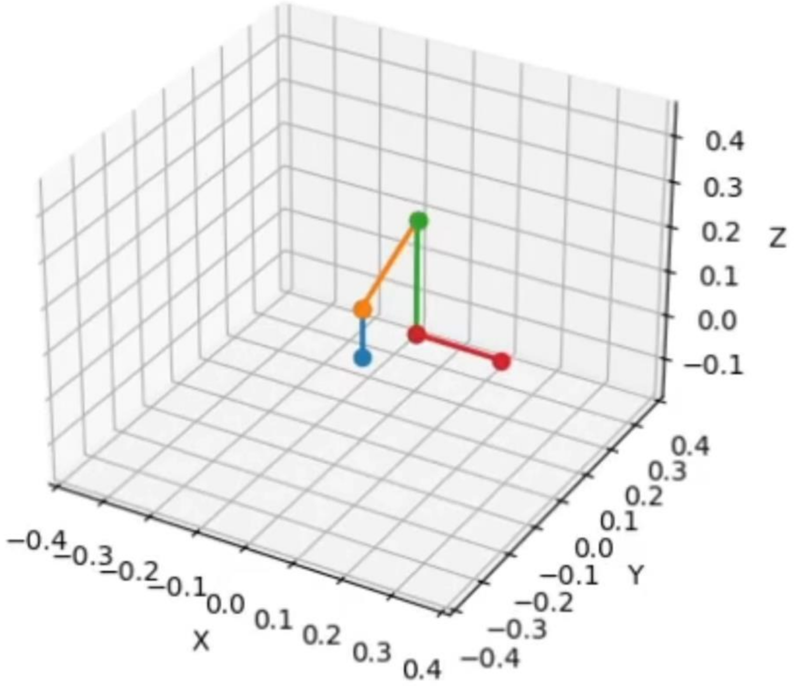
Kinematics Modelling - Inverse Kinematics

Method Used – Geometric:

- Chosen for its simplicity and clarity
- Suitable for the specific robotic arm configuration

Trajectory Planning

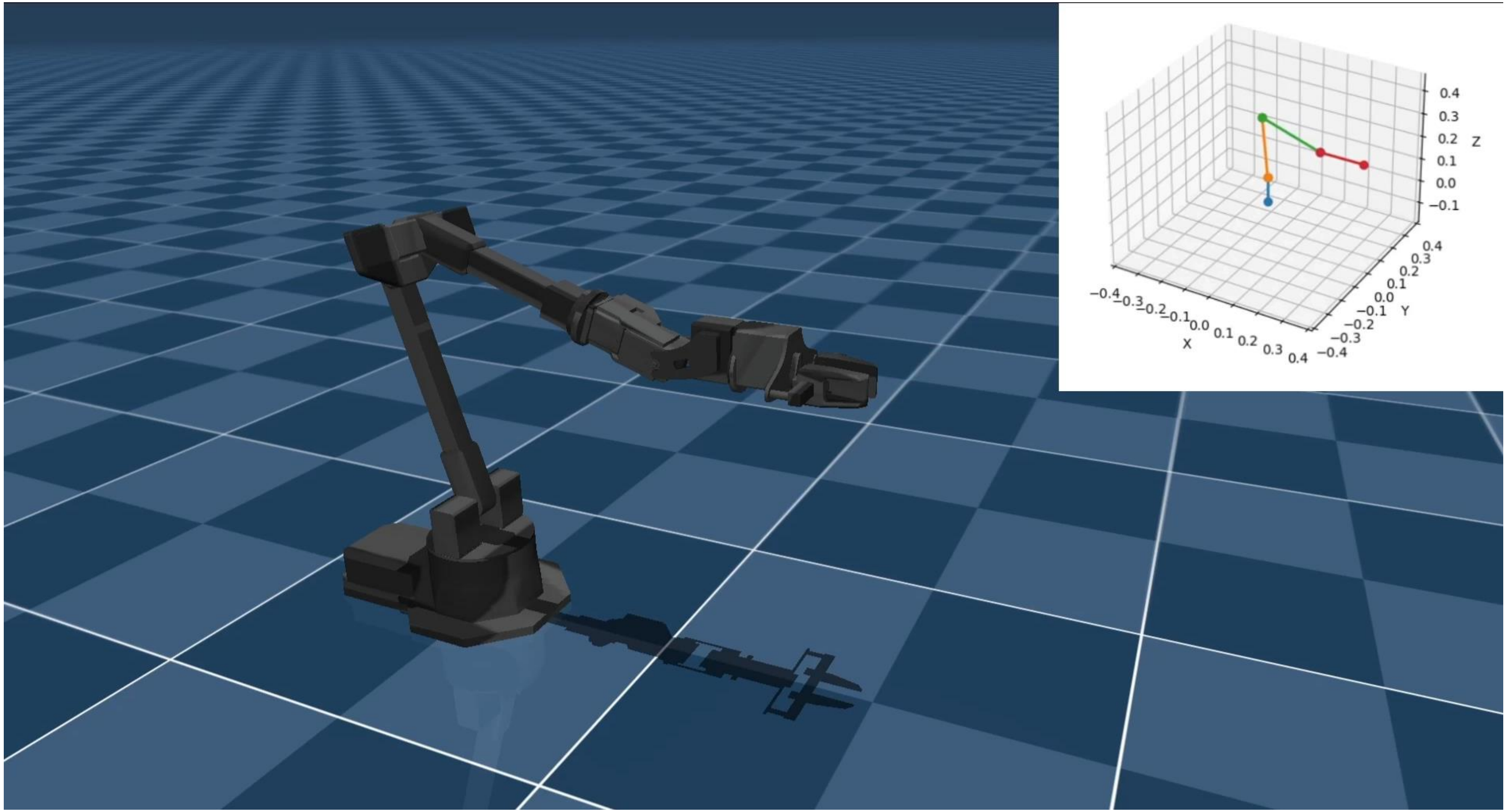
- Path Generation
- Inverse Kinematics
- Interpolation of Joint Angles
- Velocity and Acceleration Profiles



Trajectory Plot of Robotic Arm using Matplotlib

Note: Detailed angle equations are included in the appendix

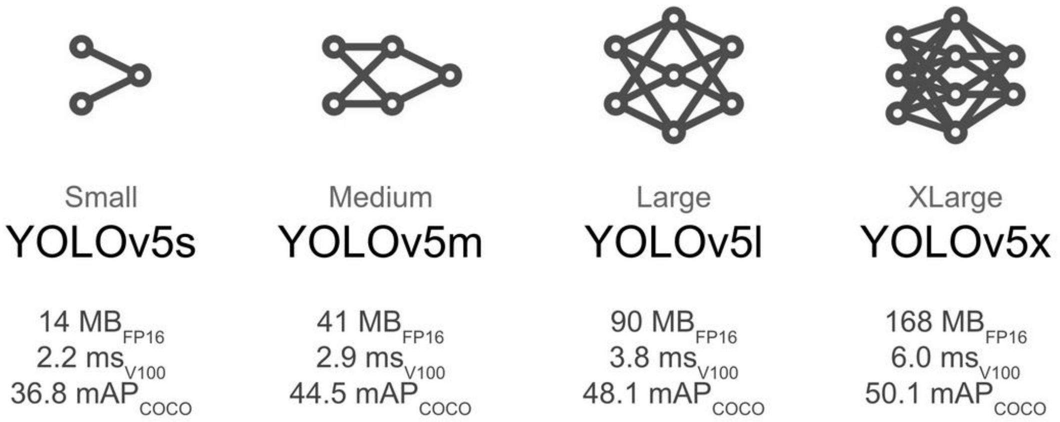
Kinematics Modelling Simulation



Object Detection

- Why YOLOv5s?
  - Computational Efficiency
  - High Accuracy
  - Real-Time Performance
  - Ease of Training and Deployment

- Model Training Workflow
  - Dataset Preparation
  - Data Augmentation
  - Training the Model
  - Validation and Testing



Family of YOLOv5

https://github.com/ultralytics/yolov5

Synthetic Image Generation

- 3D Modeling and Scene Creation
  - 3D models created using SolidWorks and exported as STL files
  - HDRI images for realistic backgrounds

Transform

Location:
X 1.0392 m
Y -5.9439 m
Z -1.0546 m

Rotation:
X 89.586°
Y -20.233°
Z 180.07°

XYZ Euler

Scale:
X 0.013
Y 0.013
Z 0.013

Dimensions:
X 0.538 m
Y 0.673 m
Z 0.539 m

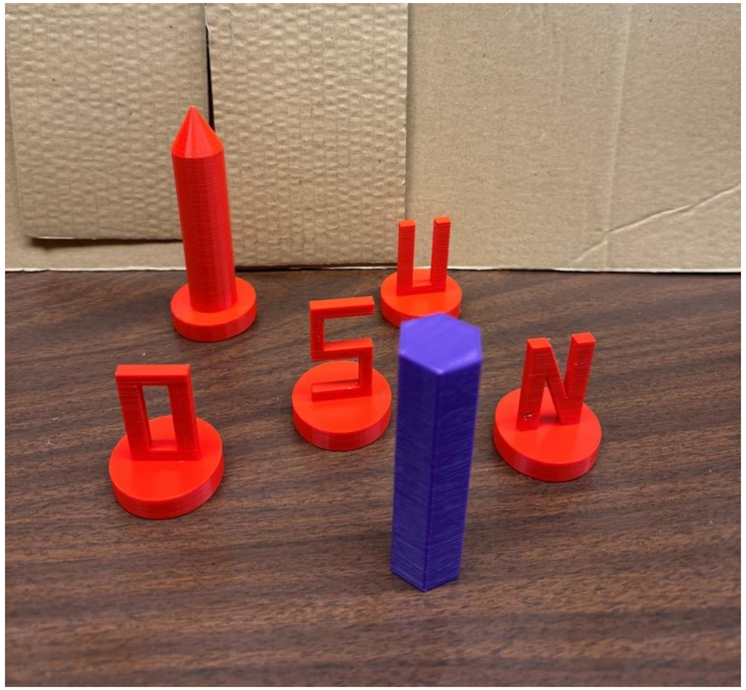
World

Surface

Use Nodes

Surface: Background
Color: 2.brown_photostudio_02_4k.exr
2.brown_photostudio_02_4k.exr
Linear
Equirectangular
Single Image
Color Space: Linear Rec.709
Alpha: Premultiplied
Vector: Default
Strength: 1.000

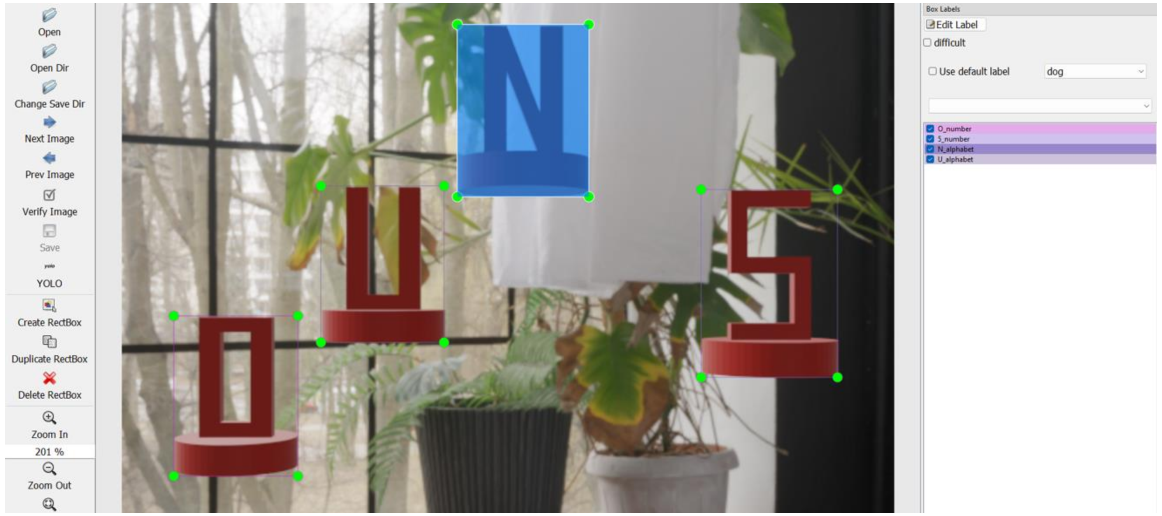
Volume
Mist Pass
Viewport Display
Custom Properties



3D printed Objects

Synthetic Image Generation

- Rendering and Annotation
  - Image Rendering
  - Annotation Generation with Python
  - Verification with labelImg
  - 8000 images are generated



☐ difficult

☐ Use default label dog

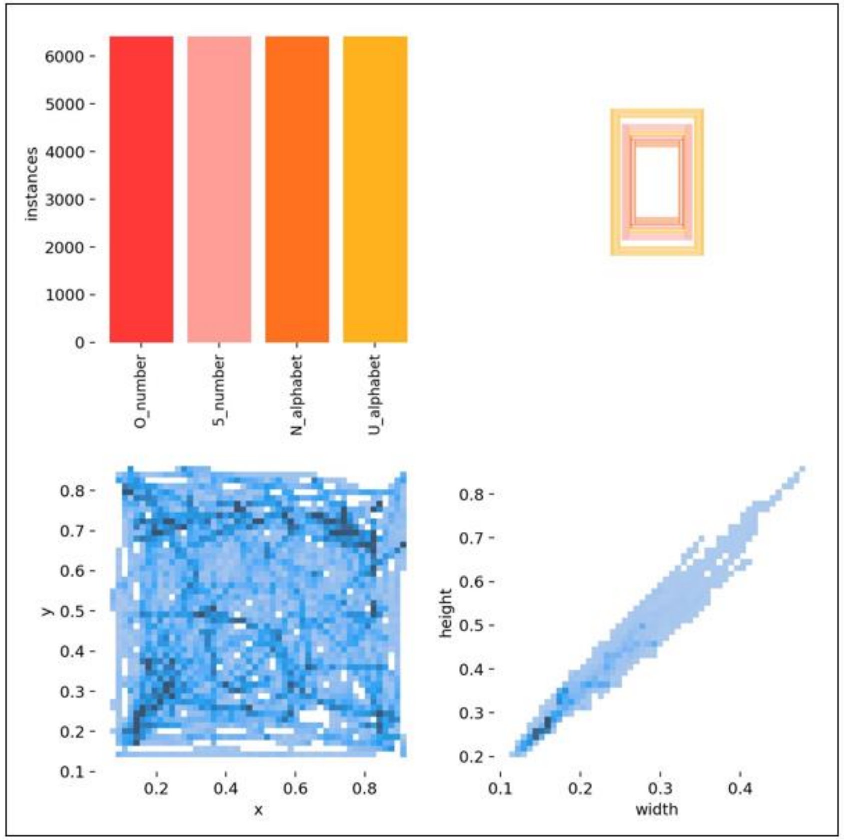
☑ O_number
☑ S_number
☑ N_alphabet
☑ U_alphabet

Annotation verification using labelImg

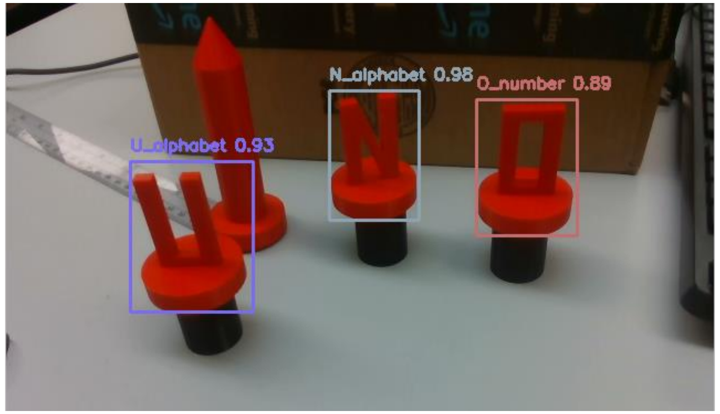
Collage of generated images

Model Training and Validation

- Classes and Distribution
  - Object Classes: 0_number, 5_number, N_alphabet, and U_alphabet
  - Dataset Composition
  - Class Distribution
- Model Training
  - Image size – 640x480 pixels
  - Number of epochs - 100
- Validation and Testing
  - Validation Process
  - Testing on Real-World Data

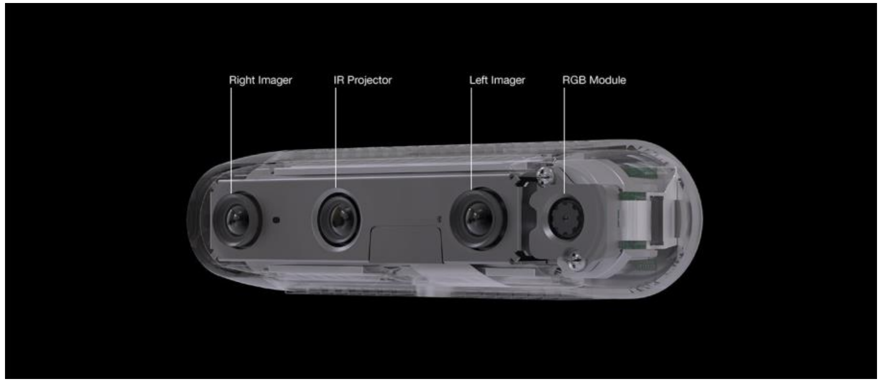


Class Distribution

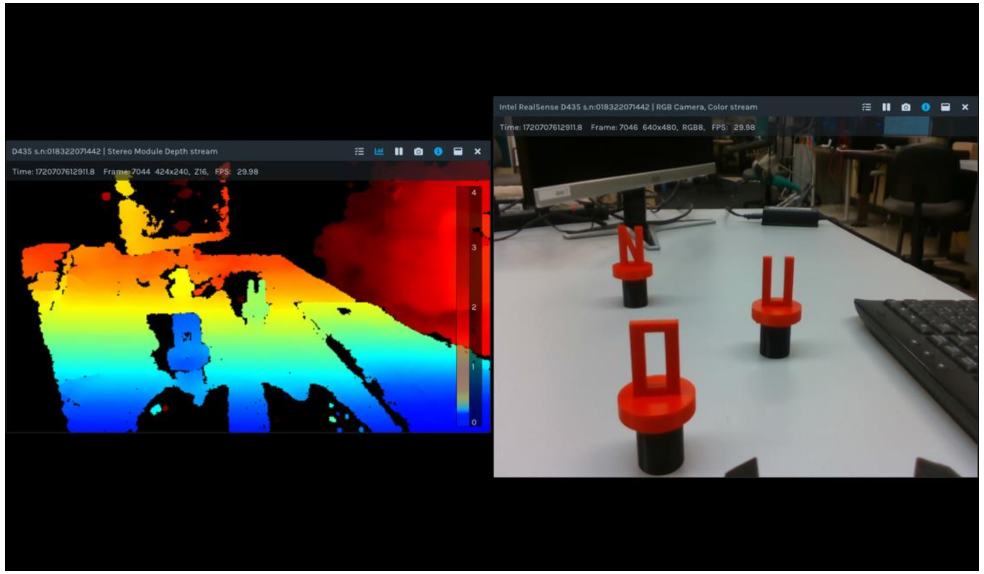


Hardware - Vision

- Depth Camera - Intel RealSense D435
  - Specifications:
    - Resolution: Up to 1280 x 720 for depth and RGB streams
    - Field of View: 87° × 58° × 95° (±3°)
    - Depth Range: 0.2 to 10 m
    - Frame Rate: 90 fps for depth data



- Functionality and Integration
  - Captures both RGB and depth information
  - Connected to the Jetson Nano via USB 3.0
  - pyrealsense2: python library

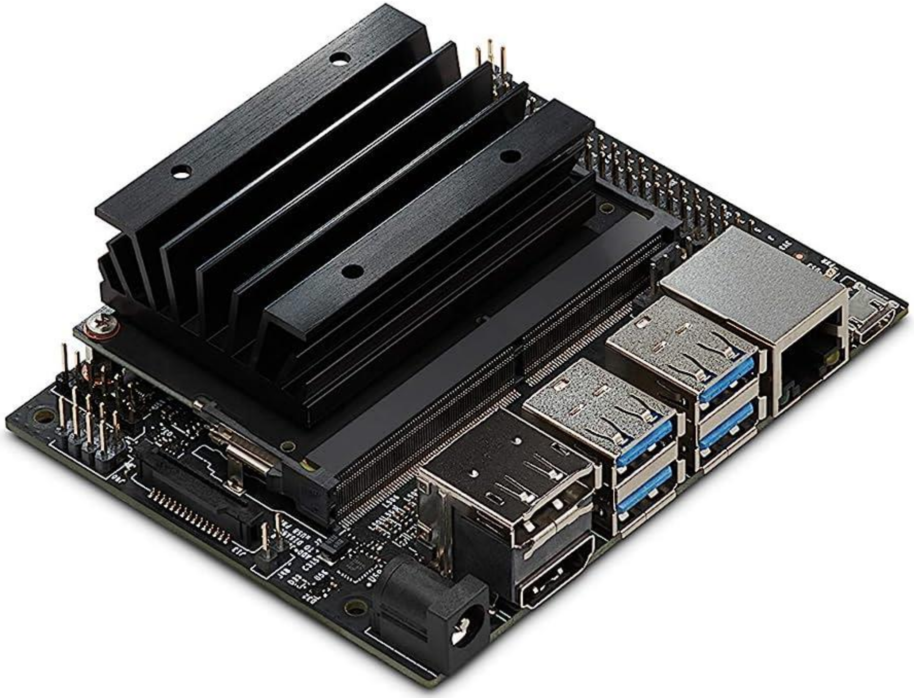


```
D435 s.n:018322071442 | Stereo Module Depth stream
Time: 1720707612911.8   Frame: 7044  424x240,  Z16,  FPS:  29.98
```

```
Intel RealSense D435 s.n:018322071442 | RGB Camera, Color stream
Time: 1720707612911.8   Frame: 7046  640x480,  RGB8,  FPS:  29.98
```

Computing Hardware

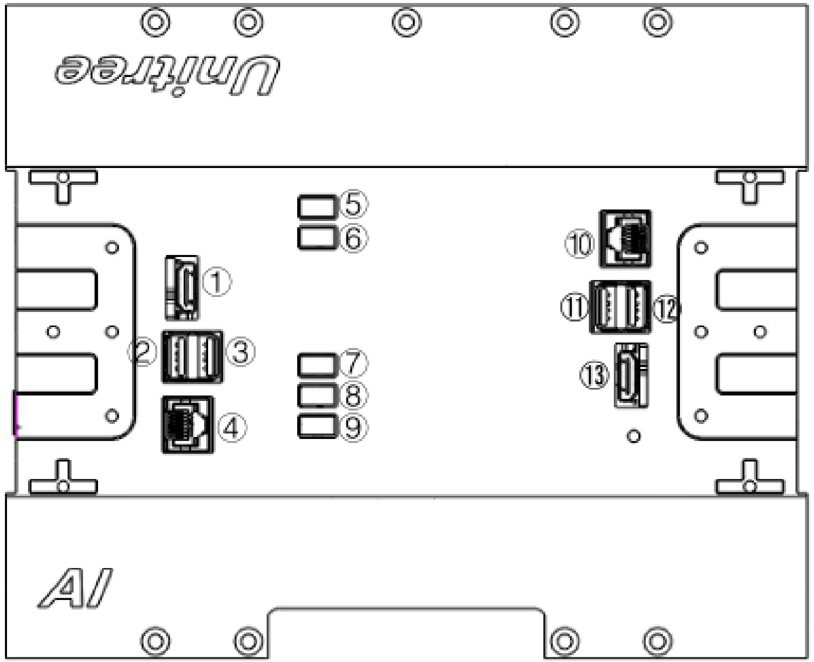
- Jetson Nano
  - Specifications:
    - CPU: Quad-core ARM Cortex-A57 MPCore processor
    - GPU: 128-core Maxwell GPU
    - Memory: 4GB LPDDR4
    - Storage: microSD card slot
    - Connectivity: Includes USB 3.0, HDMI, and Ethernet port



- Functionality and Integration
  - Run the YOLOv5s custom trained model
  - Central Processing Unit for the robotic system
  - Operates on a Linux-based system

Hardware – Unitree A1 Quadruped

- Specifications and Features
  - Speed: Reach speeds up to 3.3 m/s
  - Battery provides up to 2.5 hours of operation
  - Can output power to attached devices
  - Maximum payload of 5 kg

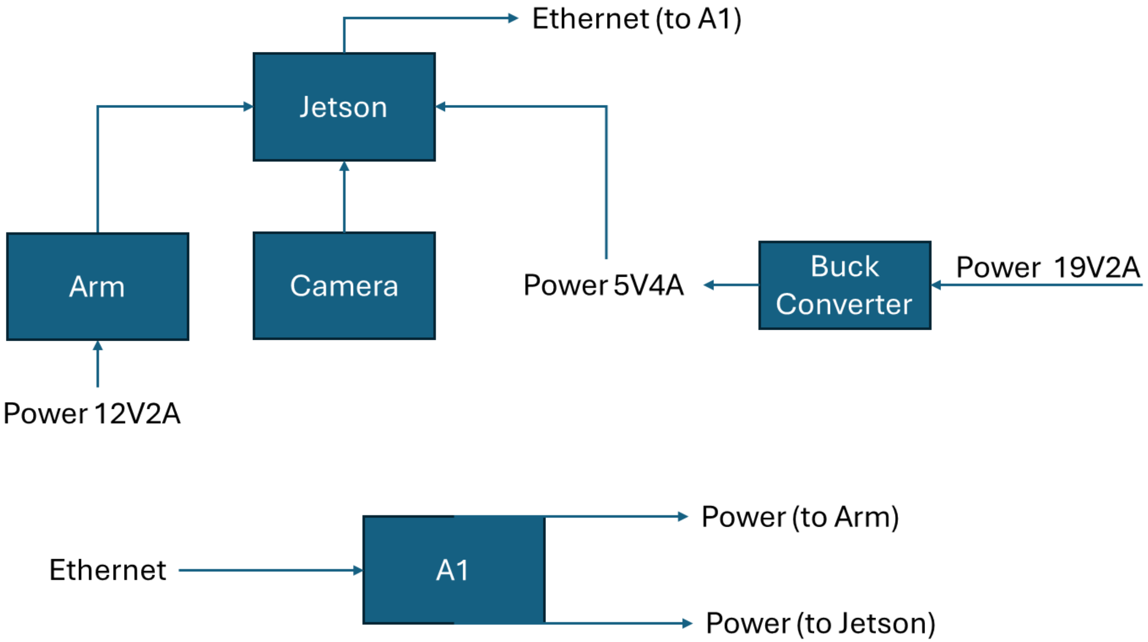


1. TX2 HDMI
2. TX2 USB3.0
3. TX2 USB2.0
4. Ethernet Interface 1
5. Power Input 24V
6. Power Input 24V
7. Power Output (5V，2A)
8. Power Output (12V，2A)
9. Power Output (19V，2A)
10. Ethernet Interface 2
11. MiniPC USB2.0
12. MiniPC USB3.0
13. MiniPC HDMI

Hardware - Power Setup

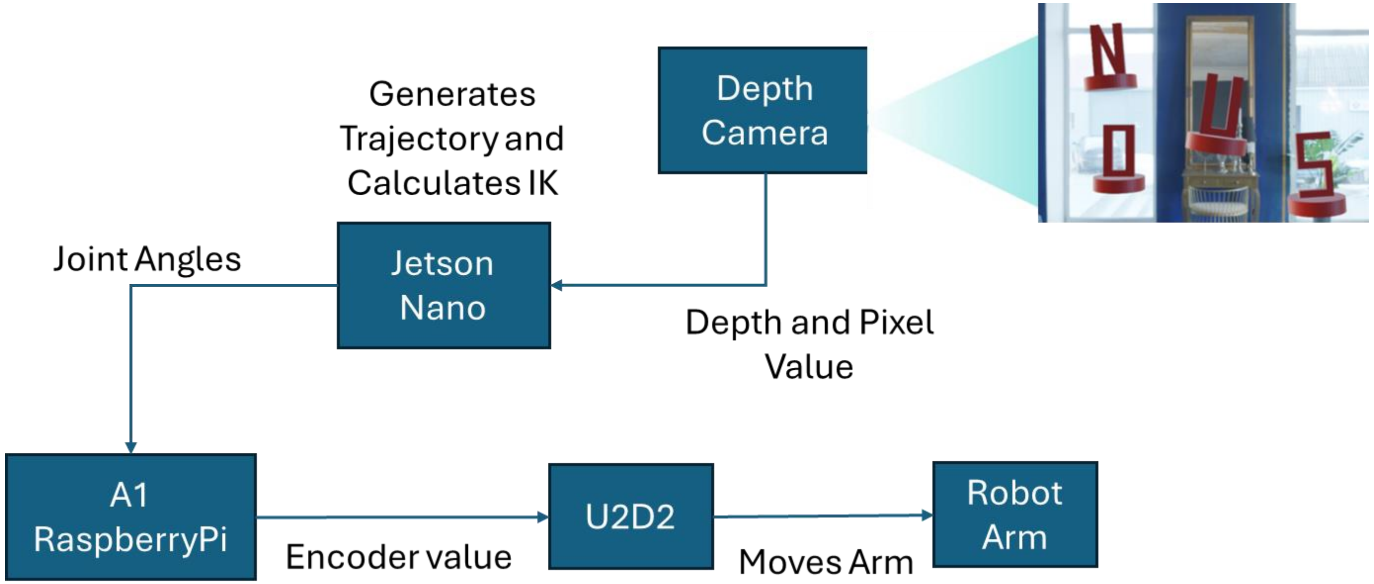
- Power Requirements
  - Jetson Nano: Requires a 5V 4A power supply
  - Robotic Arm: Requires a 12V 5A power supply
  - Intel RealSense D435: Powered via USB 3.0 from the Jetson Nano

- Buck Converter
  - Input: 19V 2A
  - Outputs: 5V for Jetson and 12V for arm



Hardware - Integration and Connectivity

- Communication and Control Flow
  - The Jetson Nano serves as the central processing unit
  - Ethernet Connection: Jetson Nano to A1 RaspberryPi
  - USB Connection: Depth Camera to Jetson and Arm to A1 RaspberryPi



Generates Trajectory and Calculates IK

Depth Camera

Joint Angles

Jetson Nano

Depth and Pixel Value

A1 RaspberryPi

Encoder value

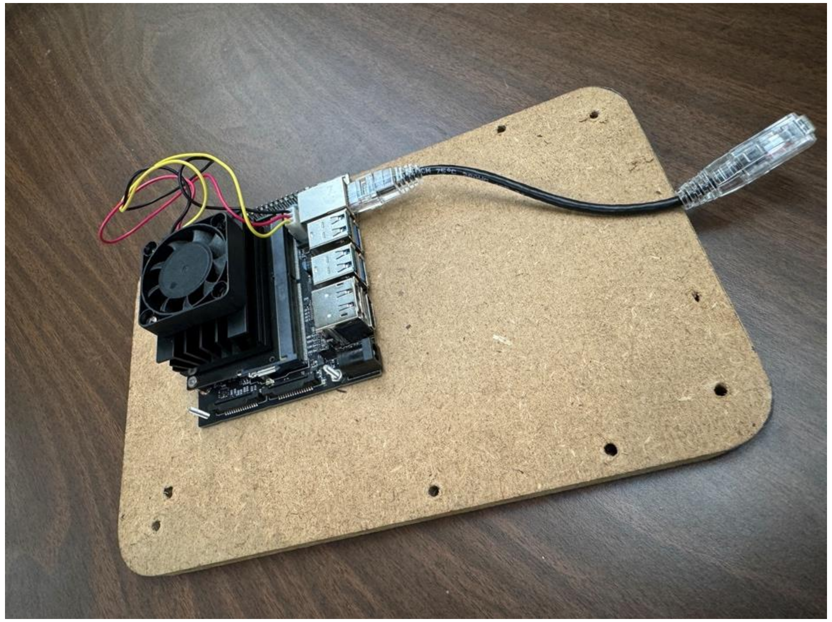
U2D2

Moves Arm

Robot Arm

Hardware – Custom Components

- 3D Printed Stand for Wood Base
  - Attach the wood base to the Unitree A1 quadruped
  - Ensures stable and reliable mounting
- Wood Base for Arm and Jetson Nano
  - Foundation for the arm and to house the Jetson



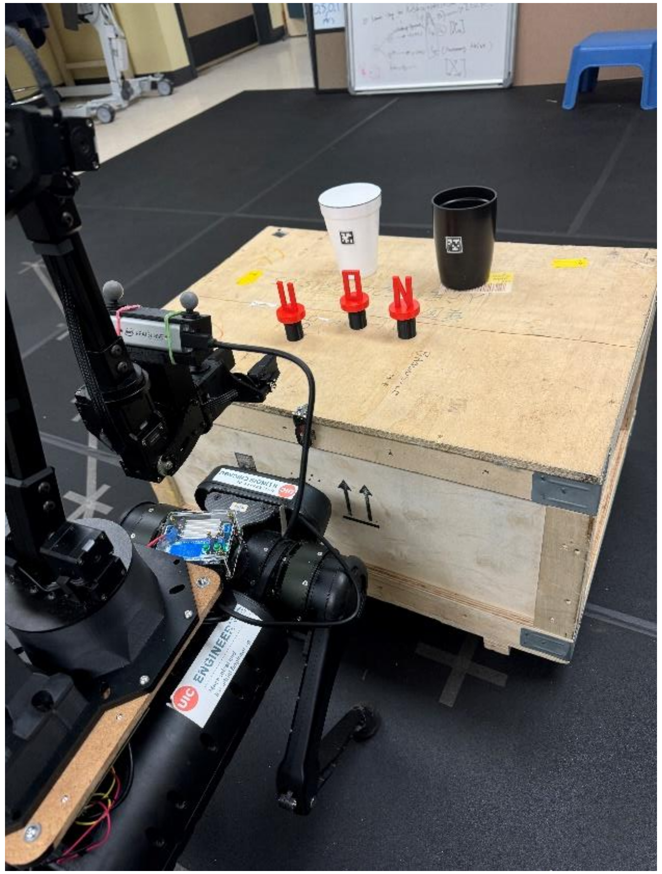
Hardware – Custom Components

- 3D Printed Support for Camera
  - To mount the depth camera for the arm
  - To hold the camera at an optimal angle
- AprilTag Labeled Glasses
  - Each labeled with a unique AprilTag ID, to detect alphabets and numbers.
  - Precise identification and localization of the glasses

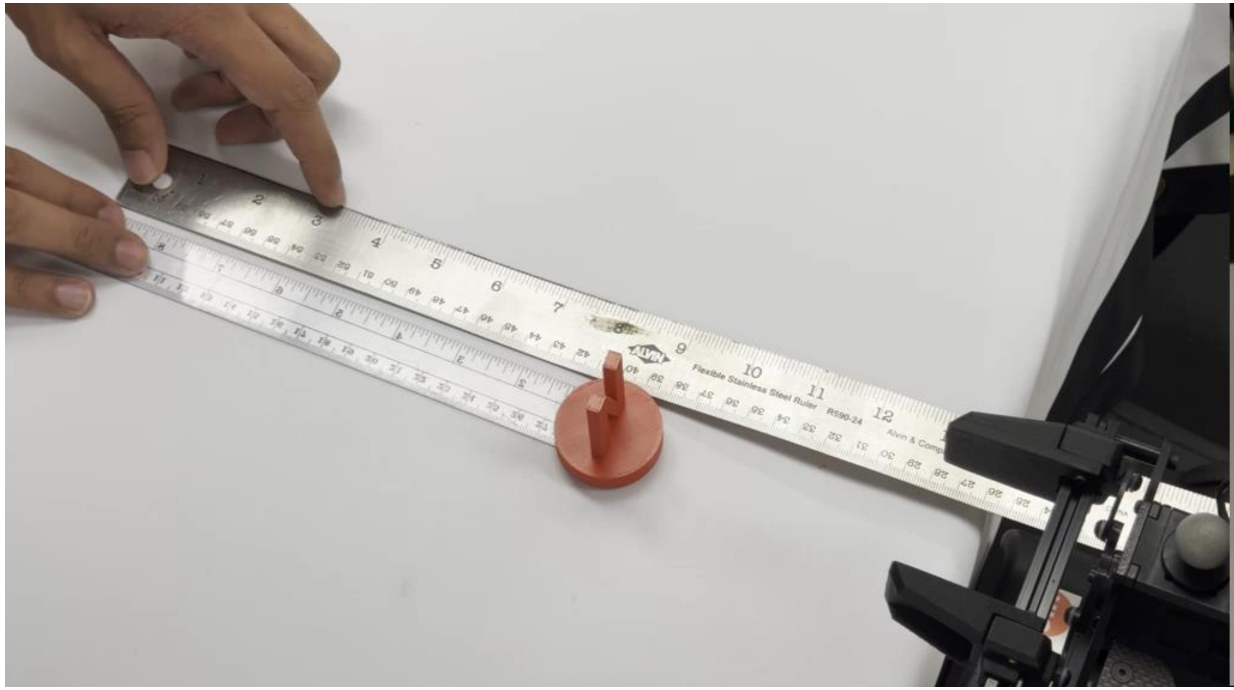
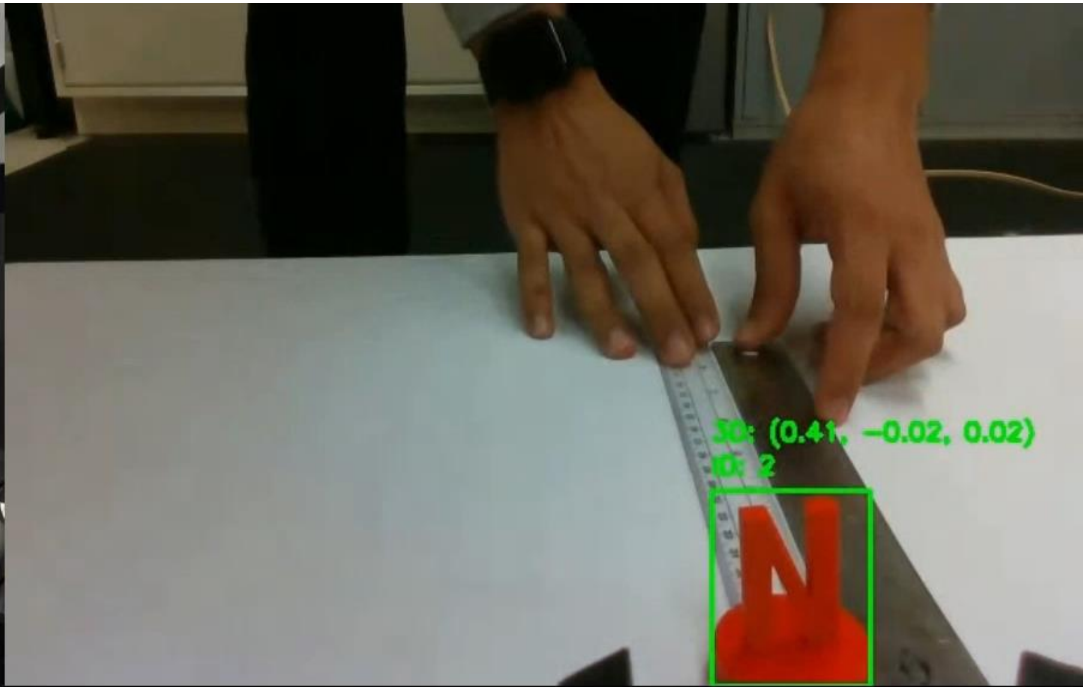
Operating Modes

- Teleoperation
  - Manually controlled by an operator
  - Precise and direct control of the quadruped movements
  - Remote Control
  - Manual Overrides

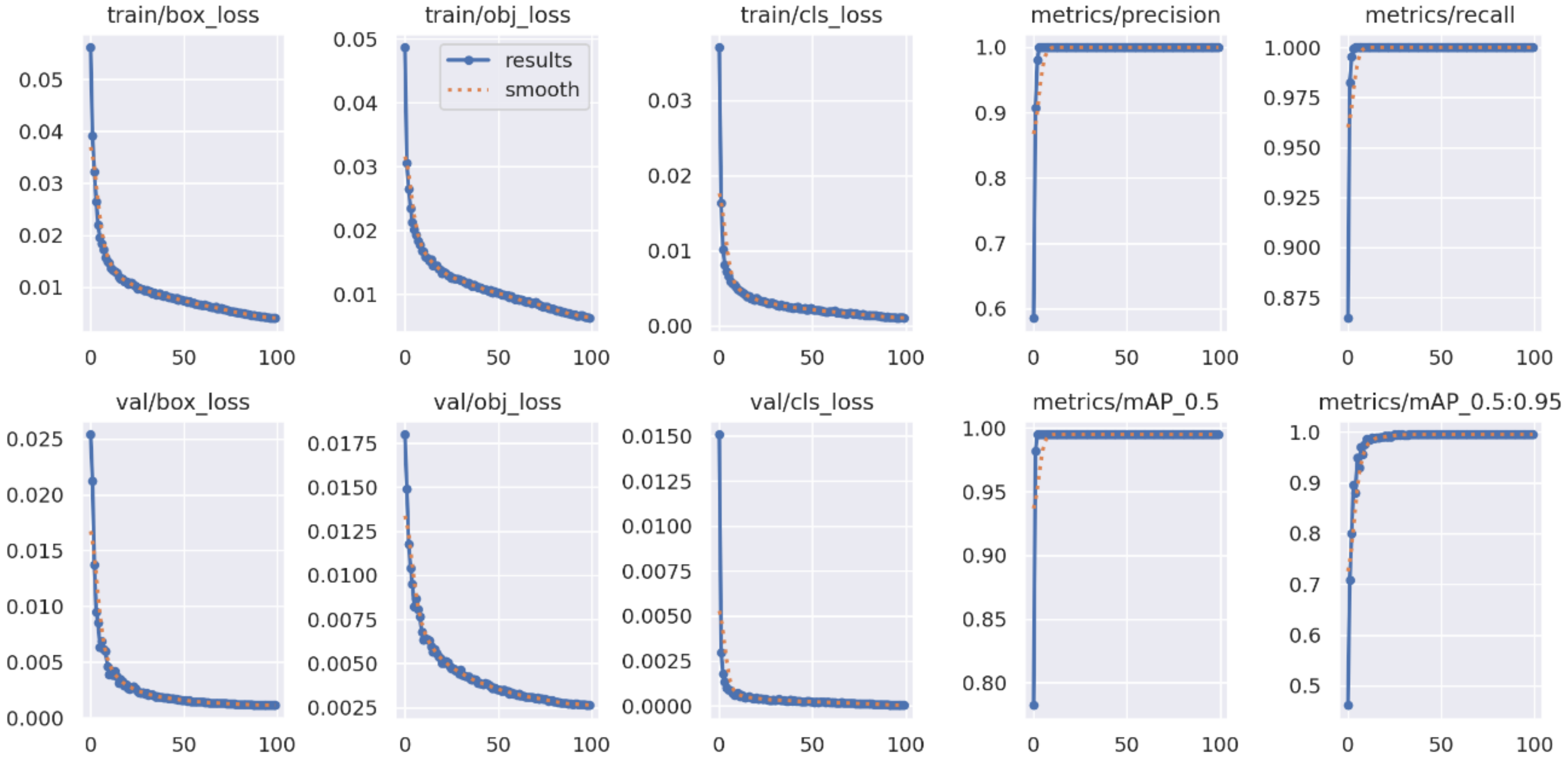
- Autonomous
  - Object Detection
  - Autonomous Sorting



Results - Intel RealSense Depth Accuracy

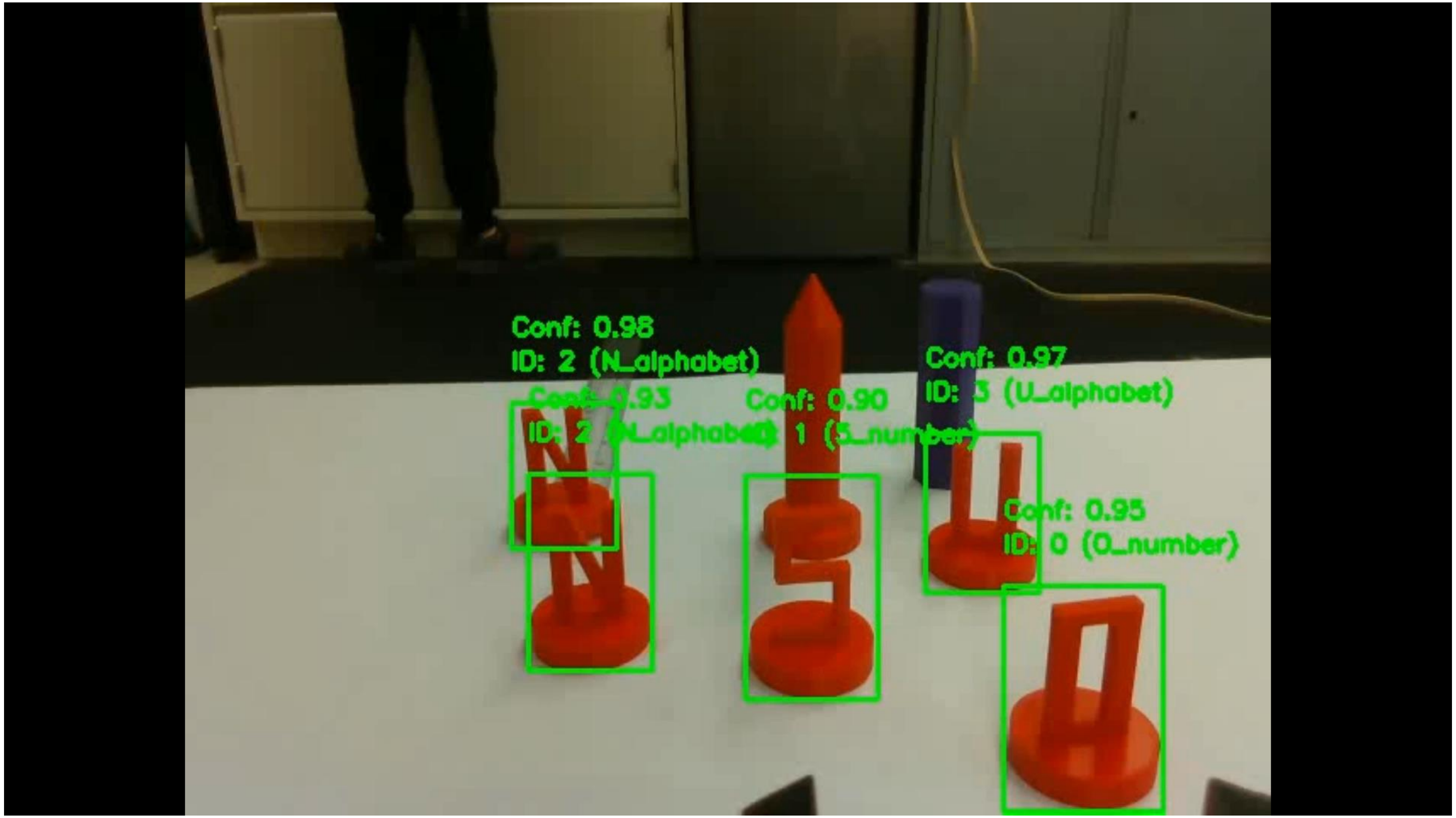



Results – Object Detection



- Training Losses: Training losses decrease over the course of 100 epochs
- Validation Losses: The validation losses decrease over time
- Performance Metrics: The precision and recall metrics are high

Results - Testing on Real-World Data



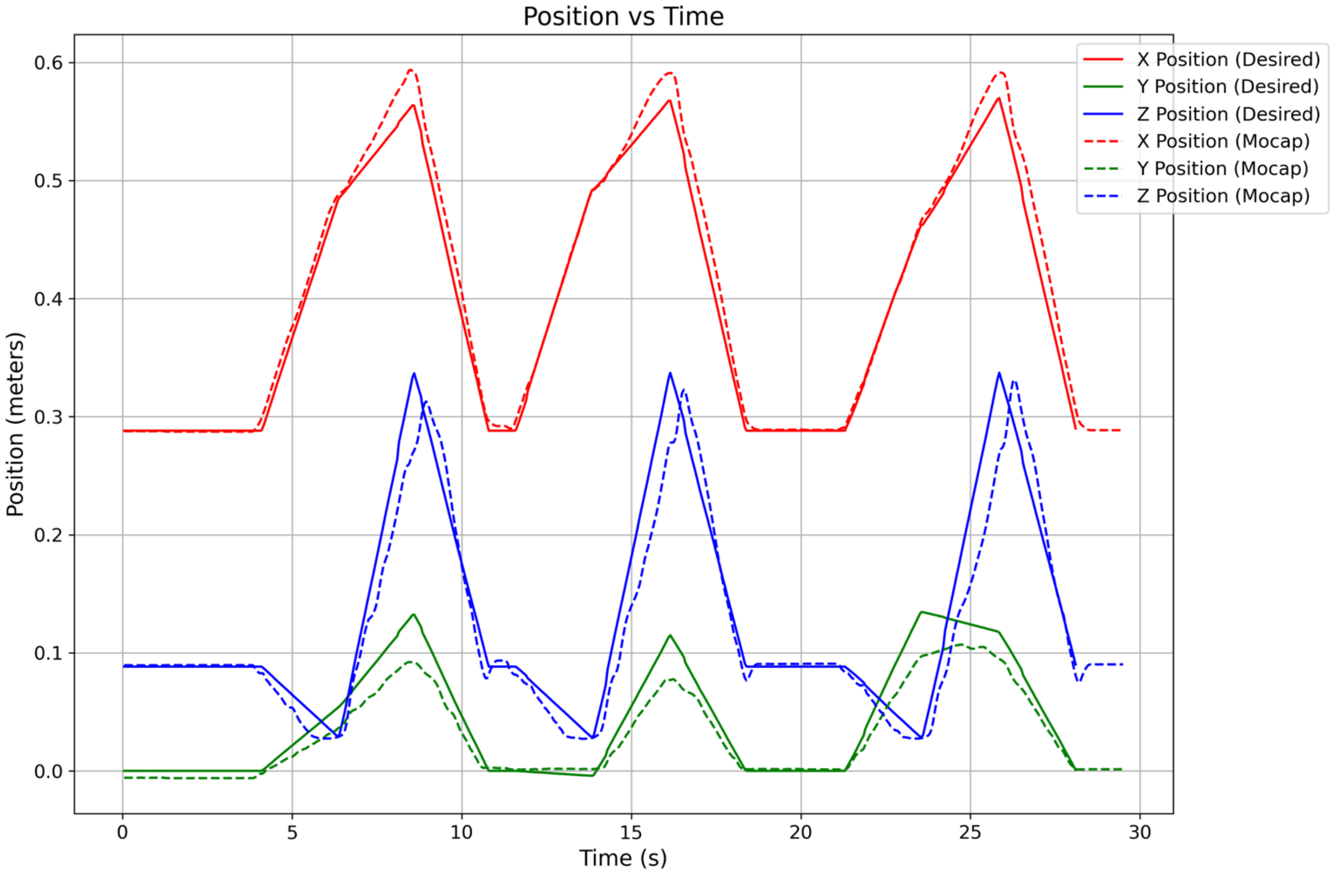
Demonstration

Demonstration

Results - Trajectory Comparison

Position vs Time:

- Y Position: noticeable deviations

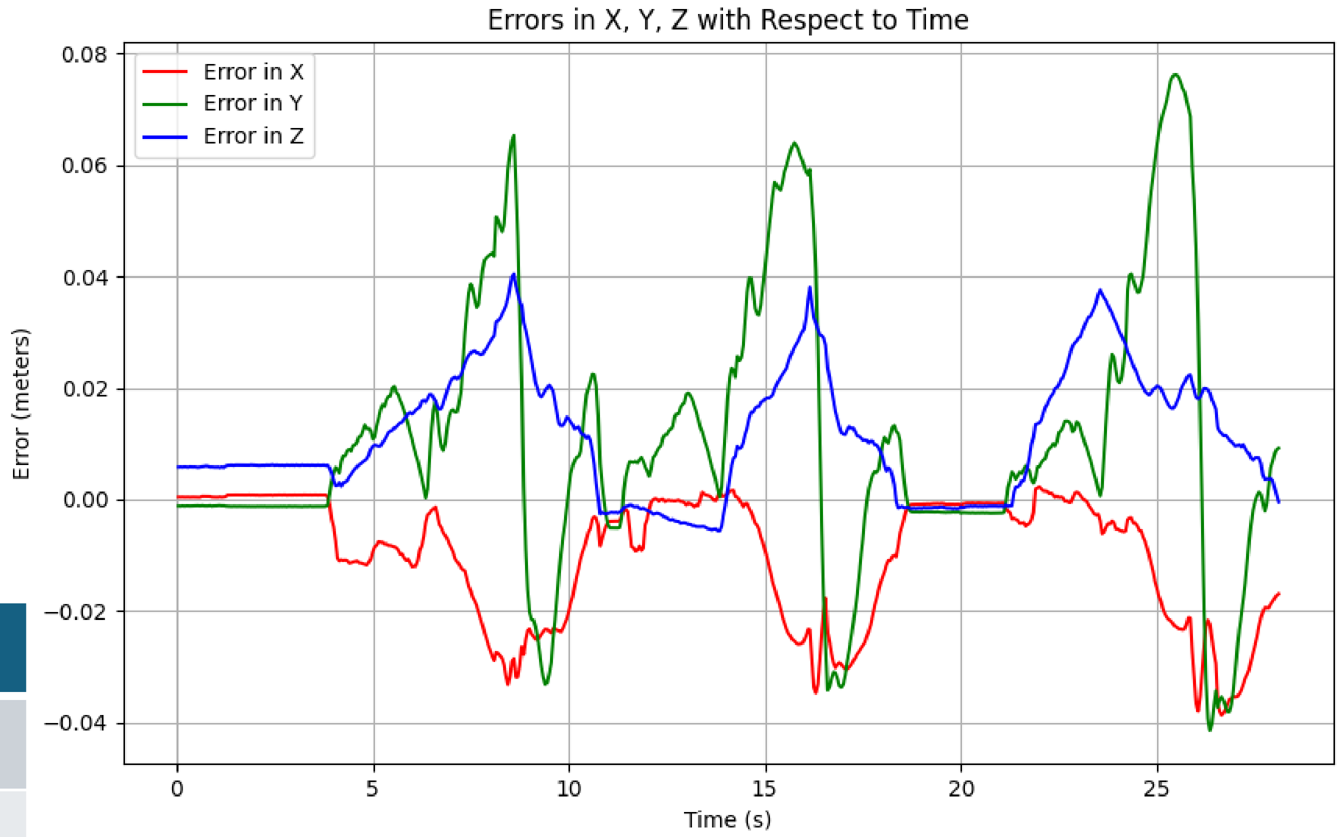


Results - Trajectory Comparison

Error Analysis:

- Mean error and standard deviation were calculated as error metrics

Axis	Mean Error (m)	Standard Deviation (m)
X	-0.0101	0.0113
Y	0.0107	0.0239
Z	0.0116	0.0113



Results – Real Time Grabbing

Performance Evaluation:

- Ability to detect, approach, and successfully grasp objects in real-time scenarios
- Out of 10 objects tested, the system successfully picked up 8 of them

CONCLUSIONS

- Project successfully implemented a robotic arm for pick-and-place tasks utilizing advanced kinematics and computer vision techniques.
- Accurate object detection and positioning, enhancing the system's overall efficiency and precision.

Future Scope:

- Algorithm Optimization
- Integrate Depth camera with SLAM for navigation and Obstacle Avoidance

Q & A

Appendix

In this work, a geometric approach was employed to solve the inverse kinematics for the robotic arm. Each joint angle can be calculated by assuming the position given. It is assumed that $\theta_{234} = \theta_2 + \theta_3 + \theta_4$. To keep the end-effector parallel to the ground, θ_{234} is considered to be 0. θ_1 can be calculates as:

$$\theta_1 = \tan^{-1}\left(\frac{p_y}{p_x}\right)$$

The angle for θ_1 ranges from -180° and 180°.

The x-coordinate and y-coordinate of the end-effector are combined into the s-coordinate using the Pythagorean theorem, as follows:

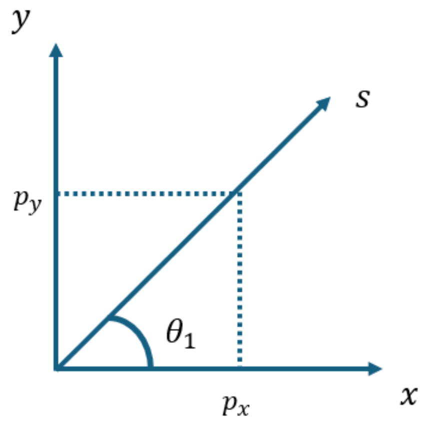
$$p_s{}^2 = p_x{}^2 + p_y{}^2$$

$$p_s = \sqrt{p_x{}^2 + p_y{}^2}$$

The r and z coordinates for joint 3 can be calculated as follows:

$$s_3 = p_r$$

$$z_3 = p_z - d_1$$



Combination of *x* and *y* axis as *s*-Axis.

Appendix

θ_2, θ_3, and θ_4 can be calculated using the following equations:

$$s_2 = s_3 - a_4 \cos\theta_{234}$$

$$z_2 = z_3 - a_4 \sin\theta_{234}$$

$$\cos\theta_3 = \left(\frac{{s_2}^2 + {z_2}^2 - ({a_2}^2 + {a_3}^2)}{2a_2a_3}\right)$$

$$\theta_3 = \pm\cos^{-1}\left(\frac{{s_2}^2 + {z_2}^2 - ({a_2}^2 + {a_3}^2)}{2a_2a_3}\right)$$

$$\cos\theta_2 = \left(\frac{(a_2 + a_3 \cos\theta_3)s_2 + (a_3 \sin\theta_3)z_2}{{r_2}^2 + {z_2}^2}\right)$$

$$\sin\theta_2 = \left(\frac{(a_2 + a_3 \cos\theta_3)z_2 + (a_3 \sin\theta_3)s_2}{{r_2}^2 + {z_2}^2}\right)$$

$$\theta_2 = \tan^{-1}\left(\frac{\sin\theta_2}{\cos\theta_2}\right)$$

$$\theta_4 = \theta_{234} - (\theta_2 + \theta_3)$$

Based on the configuration of the robot arm, the angle range for θ_2 is adjusted to between 0° and 180°, and the angle range for θ_3 is adjusted to between -180° and 0° and angle range for θ_4 is between -90° and 90°.