Moo Muk Khing

# ROBOCUP: VISION-GUIDED CARTESIAN GRIPPER AND LIDAR-BASED AUTO PARKING

Final Presentation for VISION SYSTEMS FOR MOBILE ROBOT


- 67011743 Thammasorn Srimanee
- 67011383 Wikorn Pargornsatien
- 67011366 Tunwa Raipuang
- 67011698 Lucky Agarwal
- 67011351 Theechutha Suphachitkulchai

- 68011291 Atikhun Chaiwanna
- 68011479 Plainakarin Nalintusnai
- 67011644 Phuree Poopuang


LITERATURE REVIEW
&
THEORETICAL FOUNDATION

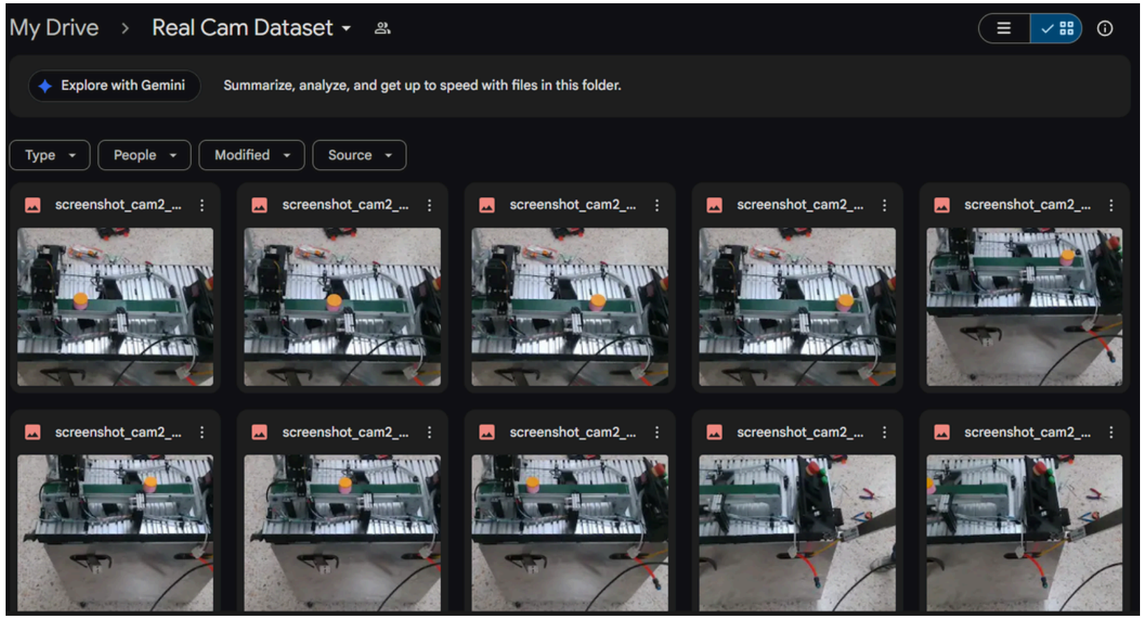
# OBJECT DETECTION USING YOLO

## What is YOLO?

- From the RealSense camera we use two streams.
  - **RGB image,** run the YOLO model to get a mask of the object.
  - **Depth Image,** extract depth values from the same masked region in the aligned

- A YOLO segmentation model is used to both detect and segment objects. In order to generate a segmentation mask, which is essential for isolating an object from the rest of the scene for further processing.

- The model was developed using a custom dataset of **over 100 images**, ensuring it to be able to detect the object.

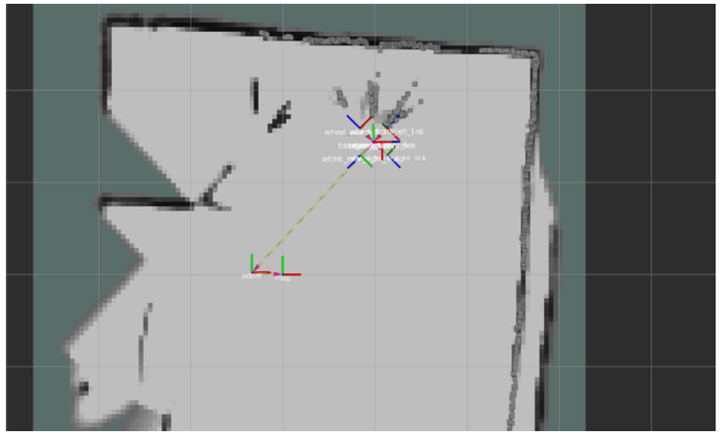


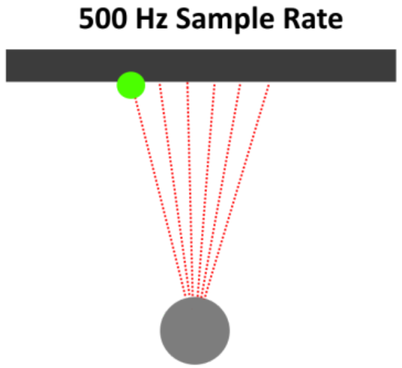
# LIDAR-BASED PERCEPTION

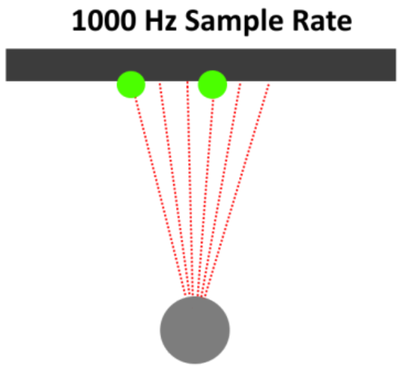
Docking perception is built on **2D LiDAR geometry**: scan data are interpreted in the sensor/robot frame, filtered to isolate the docking region, and used to estimate the station structure for positioning.

**2D LiDAR scan** → **geometric filtering** → **feature extraction** → **station-relative perception**

- The perception foundation of this work is **LiDAR-based geometric sensing**.
- Rather than using raw scan points directly, docking perception relies on **extracting stable geometric information** from the front docking region.
- This provides a reliable basis for estimating the station relative to the robot.







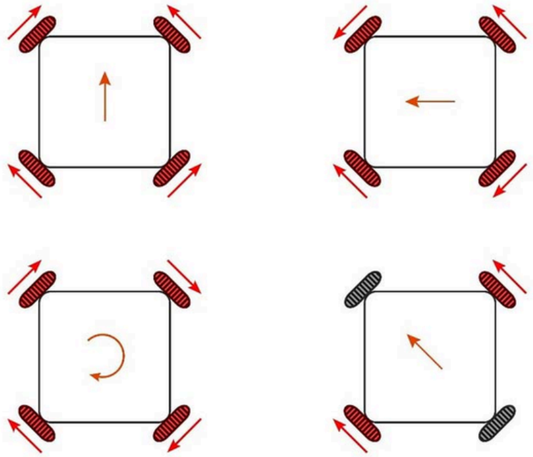
Hokuyo Laser UST-10LX / UST-20LX

# PHASED CONTROLLER

**Docking control is treated as a staged alignment task, where control priorities change as the robot moves from coarse approach to precise final docking.**

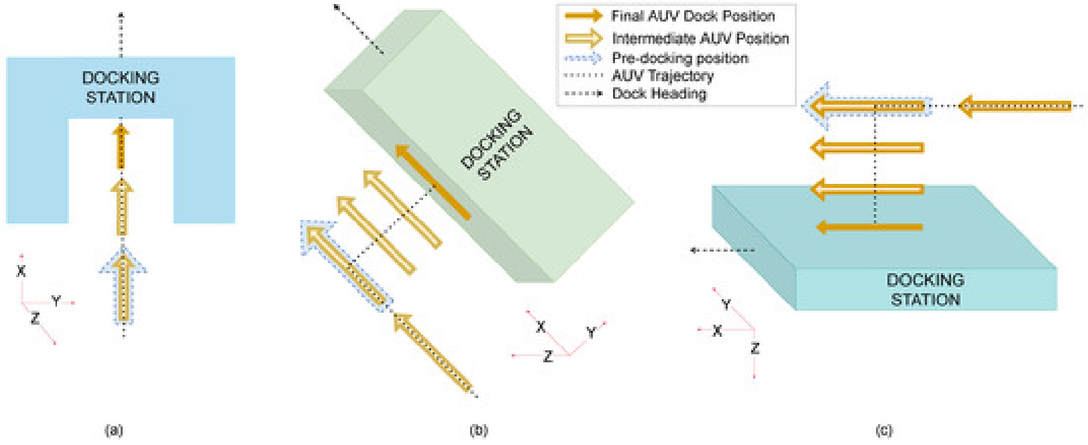
**Why phased control?**

- Far from the station, coarse approach is sufficient
- At mid range, **heading and lateral** alignment become critical
- Near the station, **precision and stability** matter more than speed



**Omnidirectional motion**

Enables planar motion with decoupled correction in translation and rotation




**Controller implication**

Different axes can be prioritized depending on docking distance and task phase

**Phase concept timeline**

**Coarse approach**

**Heading alignment**

**Fine lateral correction**

**Final docking**

- **Far** = approach
- **Mid** = heading + lateral alignment
- **Close** = precise final docking

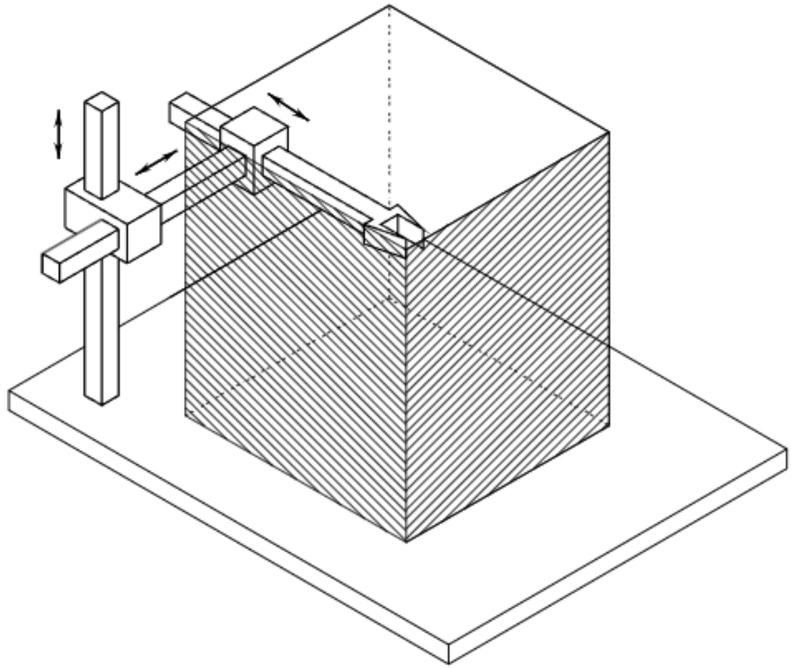
# 3-AXIS CARTESIAN & GRIPPER

## Cartesian Kinematics

- To move the robot, we first have to define its theoretical limits.
- Our physical machine is a **3-axis gantry** with orthogonal prismatic joints. In kinematic theory, Cartesian robots possess a **decoupled workspace.** This means **moving the X-axis has no mathematical effect on the Y or Z axes.**
- Because of this, our Inverse Kinematics solution is an identity matrix, a direct 1-to-1 mapping from our operational space to our joint space.

$$\begin{bmatrix} q_1 \\ q_2 \\ q_3 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

- **q1, q2, q3** = Motor Joint Positions
- **X, Y, Z** = Target Coordinates



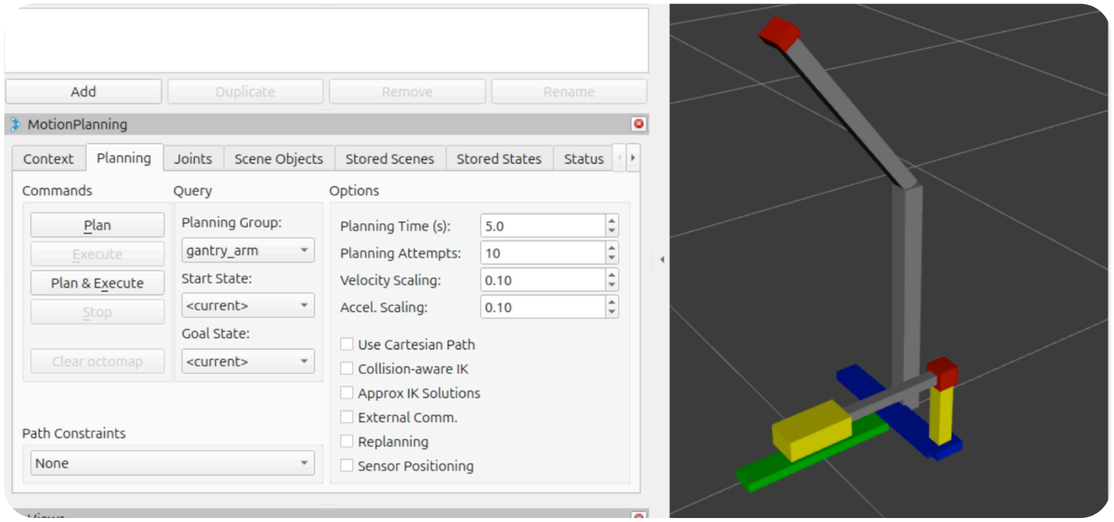

- Siciliano, B., Sciavicco, L., Villani, L., & Oriolo, G. (2009). "Robotics: Modelling, Planning and Control." Springer.

# 3-AXIS CARTESIAN & GRIPPER

## URDF Implementation & Motion Planning

- We formalized these geometric properties, link lengths, and physical joint limits using the **Unified Robot Description Format (URDF).** We then plan our movement with **Moveit.**

- This creates the **Configuration Space (C-Space),** allowing our software to know exactly where the **physical boundaries** of the robot are before it ever attempts to move.





- Chitta, S., et al. (2012). "MoveIt!: ROS Motion Planning Framework." IEEE Robotics & Automation Magazine.
- LaValle, S. M. (2006). "Planning Algorithms." Cambridge University Press.


## Movement Safeguards

- **Deterministic Waypoint Generation**
  - To avoid dragging the gripper across the table, the system implements a **"Safe Z Transfer."**
  - If the lateral movement exceeds a set threshold while the gripper is lowered, the system **automatically raises the gripper before the lateral movement.**

$$(\Delta XY \geq 30\text{mm}) \wedge (Z_{current} > 5\text{mm}) \implies \text{Trigger Safe Lift}$$

- **Deadband Filtering**
  - A spatial threshold (5mm) acts as a low-pass filter against sensor noise, **preventing churning over micro-adjustments.**

$$\Delta_{target} = \sqrt{(X_{new} - X_{old})^2 + (Y_{new} - Y_{old})^2}$$

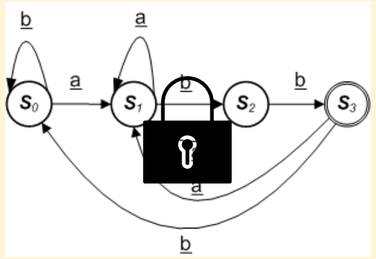
$$\text{If } \Delta_{target} < 5\text{mm}, \text{ drop coordinate.}$$

# 3-AXIS CARTESIAN & GRIPPER

## Hardware Execution & Open-Loop control


- Macenski, S., et al. (2022). "Robot Operating System 2: Design, architecture, and uses in the wild." Science Robotics.
- Acarnley, P. P. (2002). "Stepping Motors: A Guide to Theory and Practice." IET.
- Axelson, J. (2007). "Serial Port Complete: COM Ports, USB Virtual COM Ports, and Ports for Embedded Systems." Lakeview Research.


### Finite State Machine (FSM)

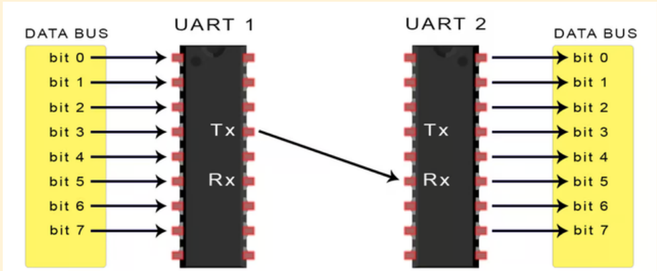



- Our stepper motors operate on an **open-loop control scheme** without absolute encoders; **they are blind to their real-world position.**
- **FSM locks and unlocks the gantry,** ensuring that trajectory requests are dropped if the hardware is already executing a move.

### Open-loop Control


- The gantry is driven by stepper motors operating in an open-loop control scheme via the **Arduino microcontroller.**
- **Kinematic accuracy** relies entirely on establishing a **strict mechanical zero point (datum)** via the hardware homing sequence before the FSM unlocks operations.

### Hardware Bridging




$$\text{Checksum} = B_1 \oplus B_2 \oplus B_3 \cdots \oplus B_7$$

- Communication to the microcontrollers relies on **UART serial protocols,** utilizing fixed-precision string encoding for kinematic target points and XOR checksum validation for the proprietary 8-byte gripper packet structure.

# AUTO PARKING: STATION DETECTION

## LiDAR Scan & Region of Interest

### INPUT

**/scan (LaserScan)**

- 2D range measurements from the **front-facing LiDAR**
- Angles are defined in the sensor frame
- Invalid ranges are discarded

### COORDINATE FRAME

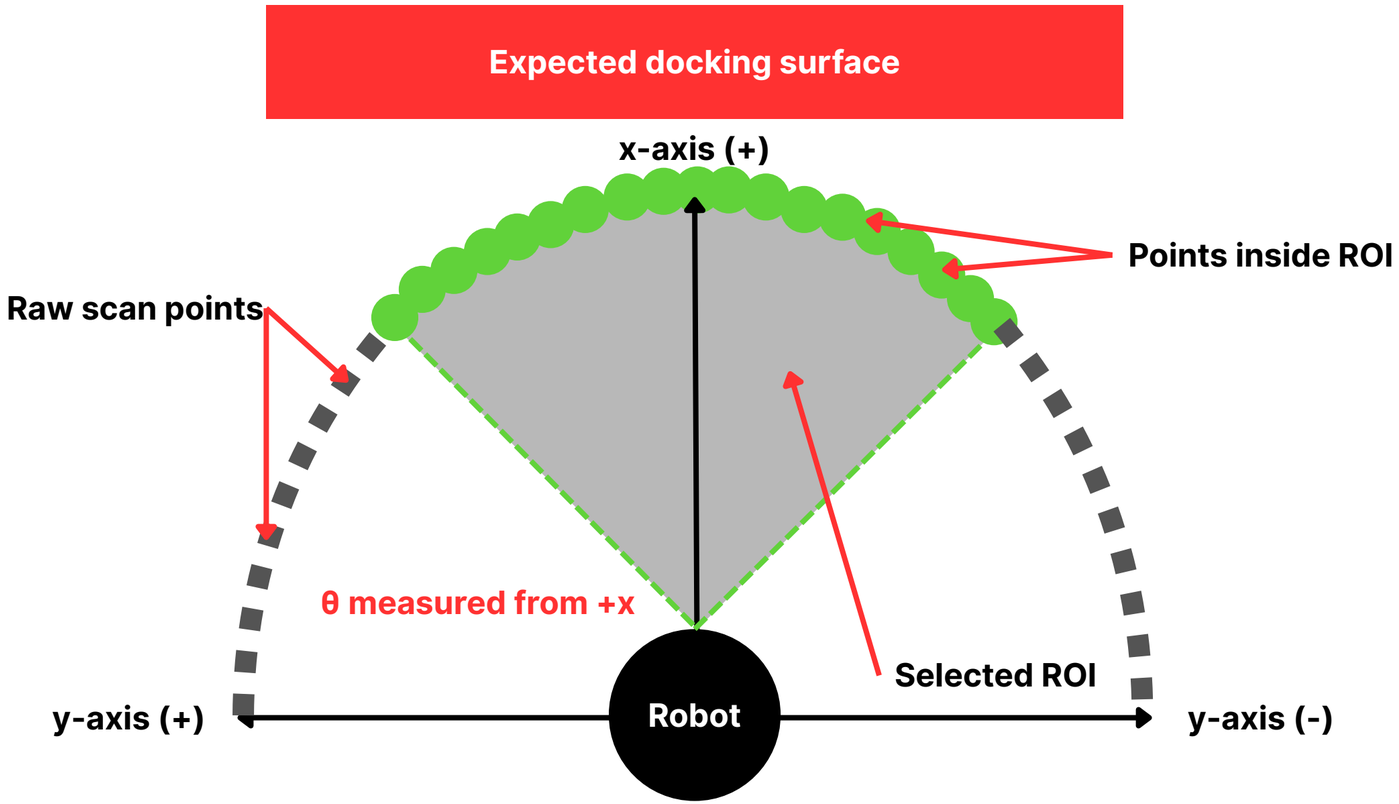
Robot-frame representation

- **x-axis:** forward distance $x=r\cos\theta$
- **y-axis:** lateral position $y=r\sin\theta$

### REGION OF INTEREST

**Region of Interest (ROI)**
Only the **forward docking region** is kept for further processing.

### TOP-VIEW ROI SELECTION




**Example:** forward FOV, valid scan range, and bounded lateral region

# AUTO PARKING: STATION DETECTION

## Point Filtering Strategy

The LiDAR scan is **filtered progressively** to remove irrelevant returns and preserve only candidate station points.

### 1 Field-of-View filter

Keeps only points within the **forward-facing** angular window around the robot's **+x direction**.

### 2 Range gate

Rejects points that lie **outside the expected docking** distance from the robot.

### 3 Lateral width filter

Keeps only points within the **bounded lateral corridor** where the station width is expected.

**Before filtering**

- raw Cartesian points from LiDAR
- scattered points across scene
- possible clutter / walls / irrelevant returns

**FILTERING**

**After filtering**

- only forward docking points remain
- much cleaner, narrower cluster

# AUTO PARKING: STATION DETECTION

## RANSAC-Based Station Face Detection

### What RANSAC does?

Finds the dominant line structure in filtered **LiDAR data**

### Why it is needed?

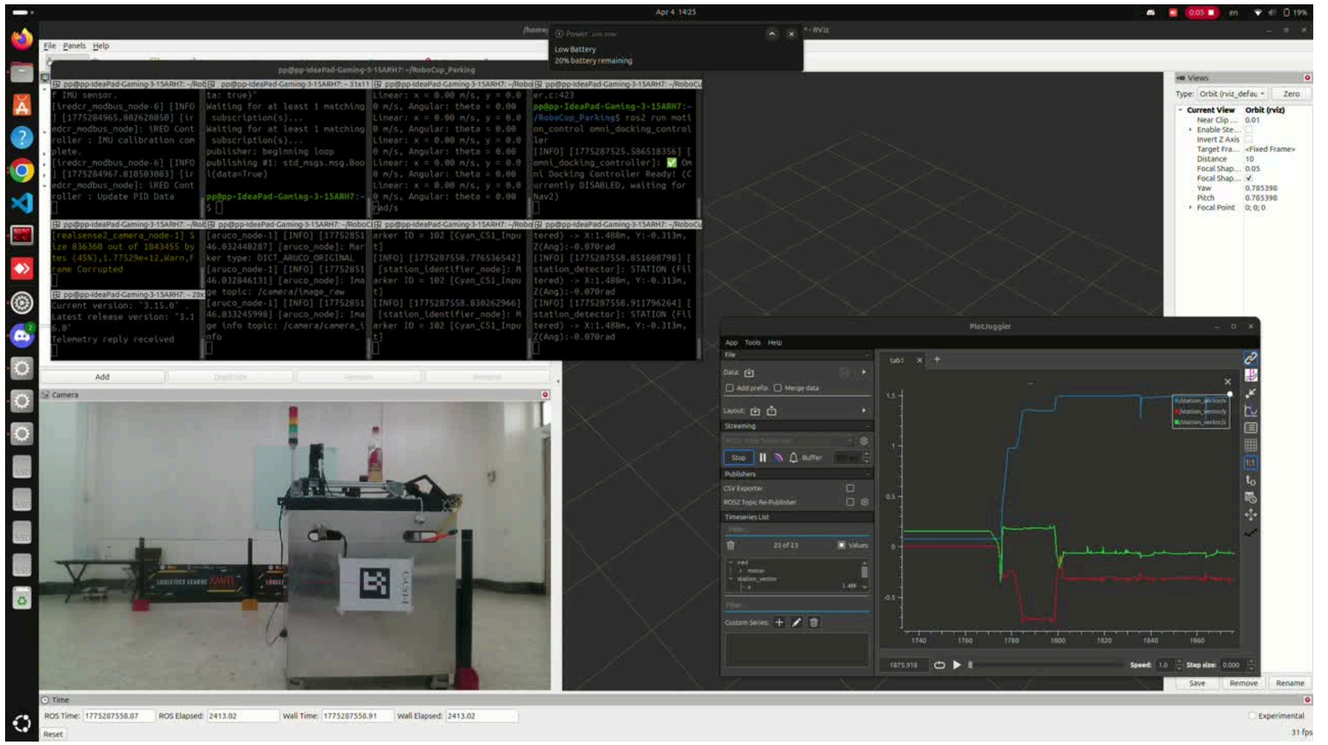
Remains robust even when clutter or spurious scan points are still present

### Output for the next stage?

Produces a **stable station-face line** for downstream pose estimation

**RANSAC converts noisy candidate points into a stable geometric target (linear form) for the next perception stage.**


"The Ultimate Guide to the RANSAC Algorithm," Dec. 16, 2024.



**Demo video when parking by using RACSAC help**

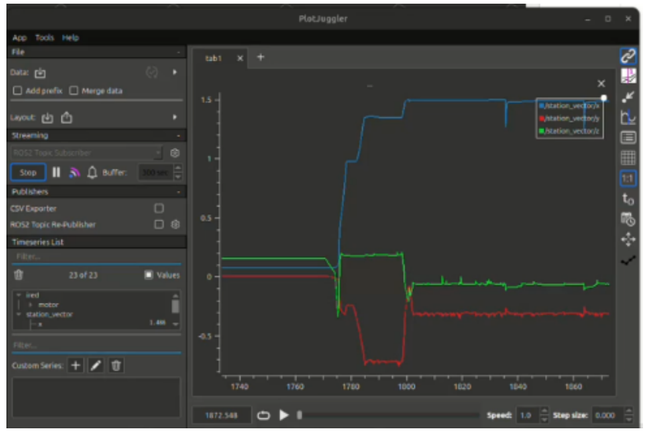
# AUTO PARKING: DOCKING POSE ESTIMATION

**From Detected Station Face to Relative Docking Pose**

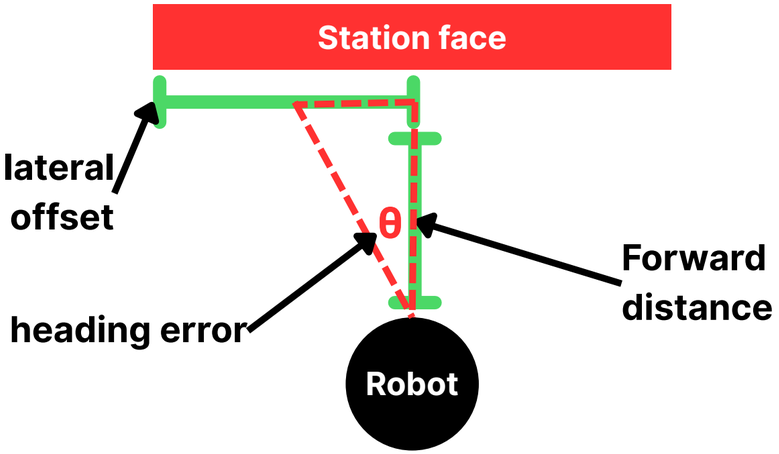
**The detected station face is converted into a compact relative docking pose for the downstream controller.**

## Input geometry

- **RANSAC** provides the dominant line representing the station face
- **Inlier points** are used to compute geometric features
- **The robot frame** is used as the reference for docking




## Pose computation




**Equation for tuning**

$$\text{vec.x} = x_c - d_{offset}$$

$$\text{vec.y} = y_c$$

$$\text{vec.z} = \phi_{err}$$

$$\phi_{err} = \arctan(m)$$

## Output vector

**/station_vector (Vector3)**

- **x:** forward distance to station face
- **y:** lateral offset
- **z:** heading error in radians

**OmniDockingController**

# AUTO PARKING: DOCKING CONTROLLER & SAFETY LOGIC

**Phase-Based Docking Controller**

## Input to controller

- **Input =** `/station_vector`
- **Contains** `x, y, yaw error`
- **Controller computes** `/cmd_vel`
- **Output type =** `TwistStamped`

## Why phased control is needed

- **Far range:** only approach is needed
- **Mid range:** heading and lateral alignment become meaningful
- **Close range:** final motion must be stable despite sensing limitations

## Phase flow

**Phase 1:** Approach only → Active axes: **X**

↓

**Phase 2:** Heading correction → Active axes: **X**, **Yaw**

↓

**Phase 3:** Full alignment → Active axes: **X, Y, Yaw**

↓

**Gate Wait:** Stability check before commit
→ X frozen, Y/Yaw monitored

↓

**Phase 4:** Blind final dock → Active axes: **X (crawl)**

↓

**Complete**

# AUTO PARKING: DOCKING CONTROLLER & SAFETY LOGIC

## Safety & Robustness Mechanisms

Safety logic is used to **prevent unstable behaviour**, **false phase transitions**, and **unsafe motion** near the station.

| Motion robustness | Decision safety |
| --- | --- |
| **Anti-windup / Integral reset**<br>Prevents velocity spikes when suppressed axes are re-enabled. | **Gate check**<br>Prevents premature transition to final docking. |
| **Deadband filtering**<br>Removes small noise near the setpoint to avoid motor jitter. | **Bailout recovery**<br>Allows the robot to back out and retry if close-range misalignment is too large. |
| **Minimum velocity injection**<br>Overcomes motor deadzone during the final crawl phase. | **Watchdog stop**<br>Immediately stops the robot if perception updates are lost. |

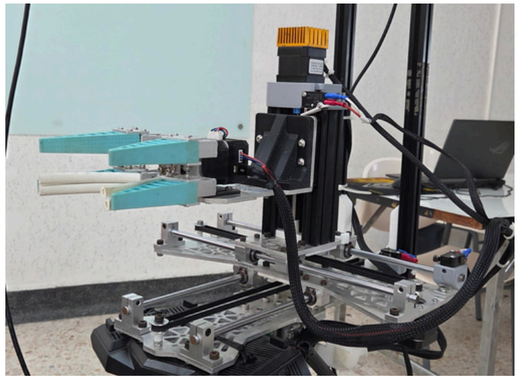
These mechanisms ensure that the **controller** remains stable, does not commit too early, and can safely stop or recover when docking conditions become **unreliable**.


STRUCTURE & SOFTWARE OVERVIEW

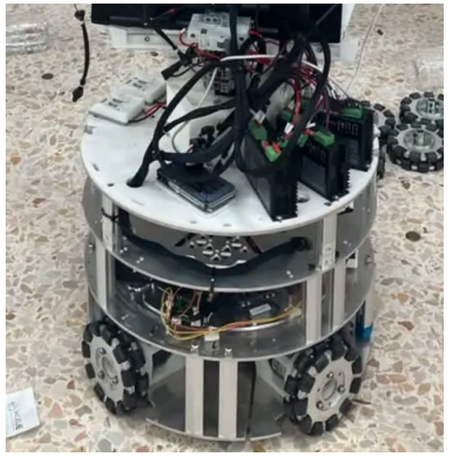
# ROBOT OVERVIEW

## Overall


## Vision + Gripper



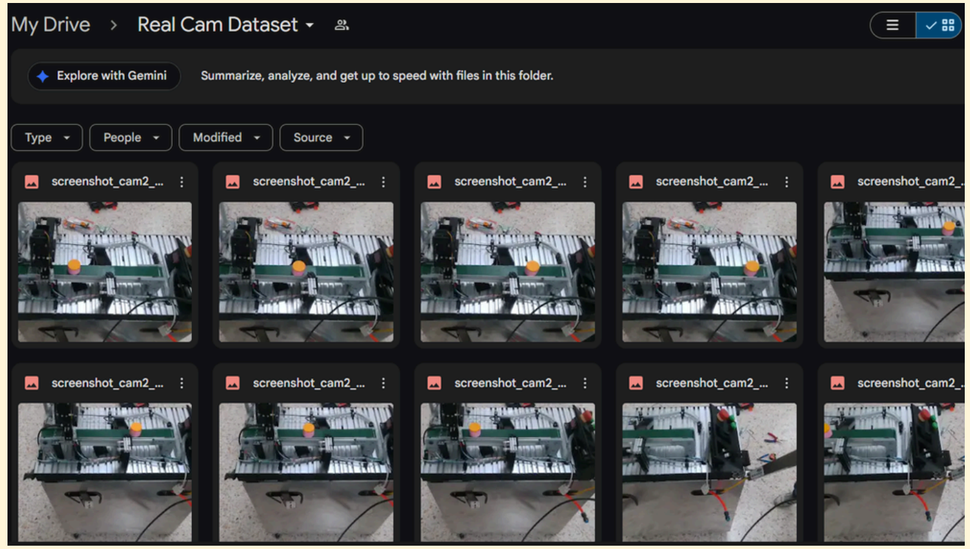
## Omni-directional base

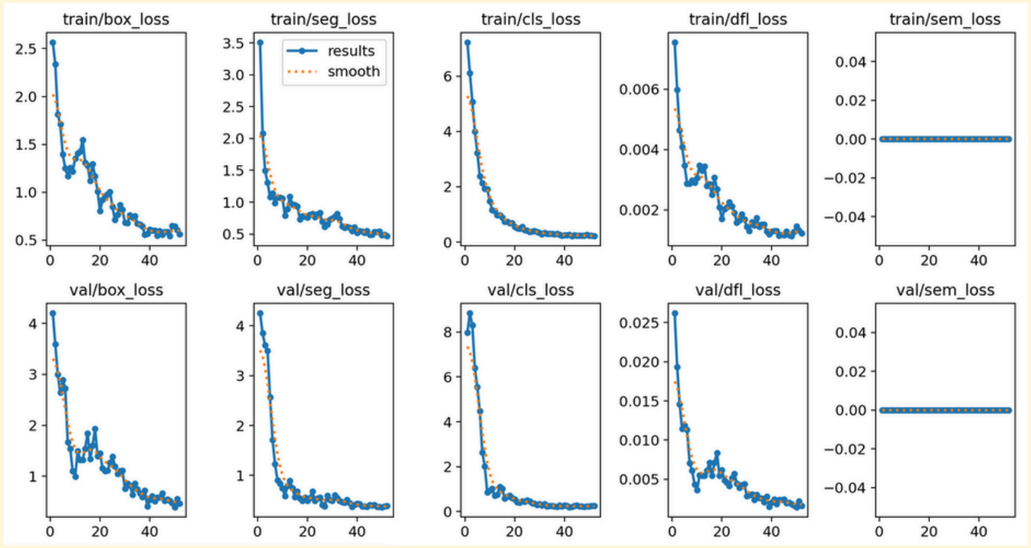


# VISION SYSTEM OVERVIEW

## YOLO Model Training

- The YOLO segmentation model was trained using over 100 images of real datasets for high accuracy object masking and eliminating vague detections.



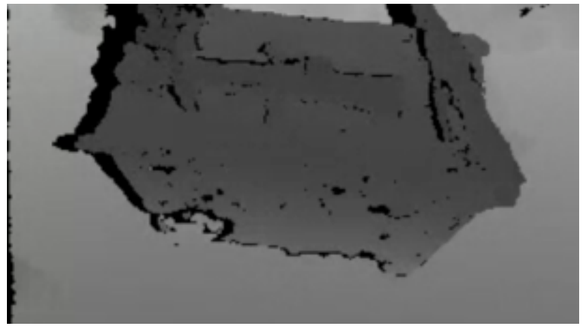



# VISION SYSTEM OVERVIEW

## Depth Estimation

Before extracting the depth, we applied a few filters to get an accurate estimate.

1. **Mask Erosion** - To remove edge pixels and prevent noisy depth readings.
2. **Median Filter** - Taking the median of all valid depth values in the region.
3. **Inlier Filtering** - Only keep values within 50mm of the median to remove outliers.
4. **30$^{th}$ Percentile** - From the filtered values, take only the 30$^{th}$ percentile as the final distance.



**Note:** We take the 30$^{th}$ percentile instead of the mean or median because it ignores distant background pixels, ensuring the depth measurement lands on the true front surface of the object.

# VISION SYSTEM OVERVIEW

## Back-Projection to 3D

Once we have the distance, we back-project the pixel coordinate into a 3D point in the camera frame using the camera intrinsics:

$$X_{cam} = Depth * \frac{Pixel\_x - Cx}{fx}$$

$$Y_{cam} = Depth * \frac{Pixel\_y - Cy}{fy}$$

$$Z_{cam} = Depth$$

**where fx, fy are the focal lengths and cx, cy is the principal point, all obtained from the /camera_info topic published by the RealSense driver.**

# VISION SYSTEM OVERVIEW

## Hand-Eye Calibration

The coordinates from the back-projection are in the camera's frame. We need them in the Gantry's coordinate frame, so the motor controller knows where to move.

**The Problem:** The coordinates are relative to the camera, but the gantry needs coordinates relative to its own physical rails to move accurately.

Instead of relying on manual measurements, we paired the gantry positions with camera readings across the workspace to train a translation matrix.

**The Affine Matrix:** Uses least squares regression to calculate a matrix that handles rotation and translation.

Better than ROS TF tree because it does not rely on the URDF being accurate. Since it uses real measurements, so any mechanical offsets accounted for.

$$\begin{bmatrix} X_{\text{gantry}} \\ Y_{\text{gantry}} \\ Z_{\text{gantry}} \end{bmatrix} = \begin{bmatrix} 1678.14 & -541.63 & -146.38 & 205.68 \\ 204.49 & 611.54 & 2814.65 & -1321.91 \\ 24.10 & -3094.43 & -5655.39 & 3573.16 \end{bmatrix} \begin{bmatrix} X_{\text{cam}} \\ Y_{\text{cam}} \\ Z_{\text{cam}} \\ 1 \end{bmatrix}$$

# VISION SYSTEM OVERVIEW

## Color Classification

The system determines reliable object colors by converting pixels to HSV/LAB color spaces, filtering out shadows, and mathematically isolating the true hue from environmental lighting.

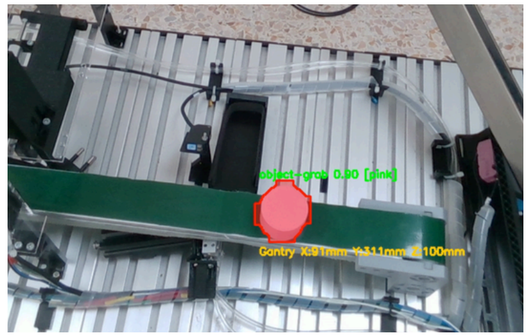
**Beyond RGB:** Converts the object's pixels into HSV and LAB color spaces to separate the object's true color from the environmental lighting/brightness.

**Noise Reduction:** Excludes extremely dark pixels where shadows destroy reliable color data.

**Achromatic Filtering:** Checks saturation and chroma to immediately isolate grayscale objects (black, white, gray, silver) before attempting to find a color.

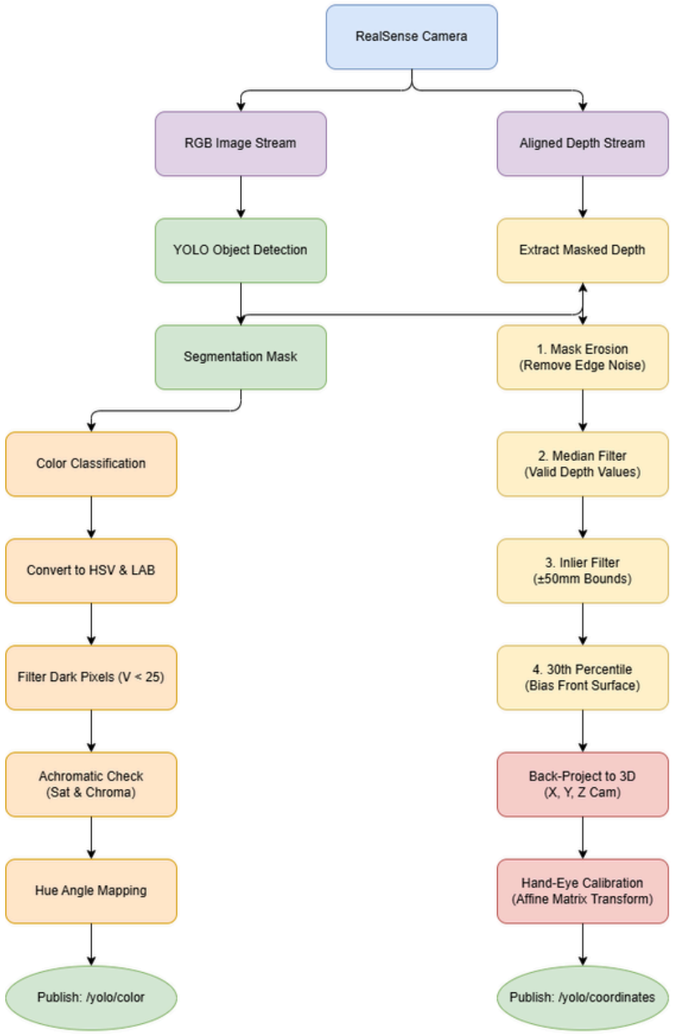
**Smart Averaging:** Uses a "weighted circular mean" on the color wheel to find the true hue, giving priority to vivid pixels so dull spots don't skew the result.

**Final Output:** Broadcasts the physical 3D coordinates and the text-based classification (e.g., "cup:red") to the robot's control system.

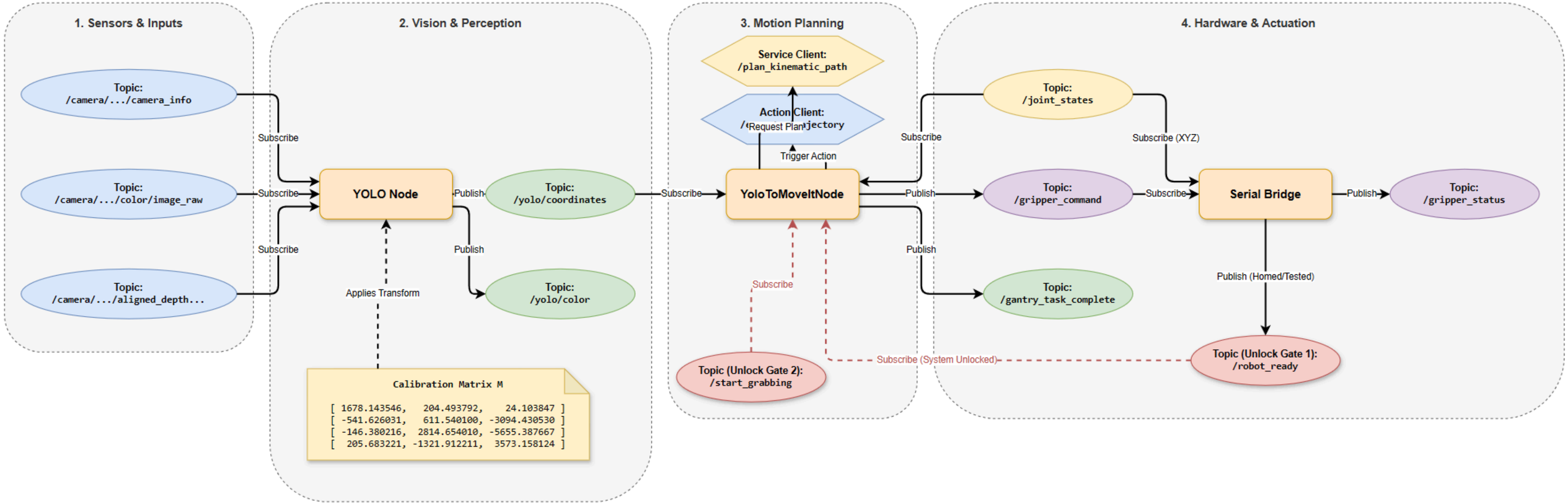


# VISION SYSTEM OVERVIEW

## Flowchart



# ROS2 NODES

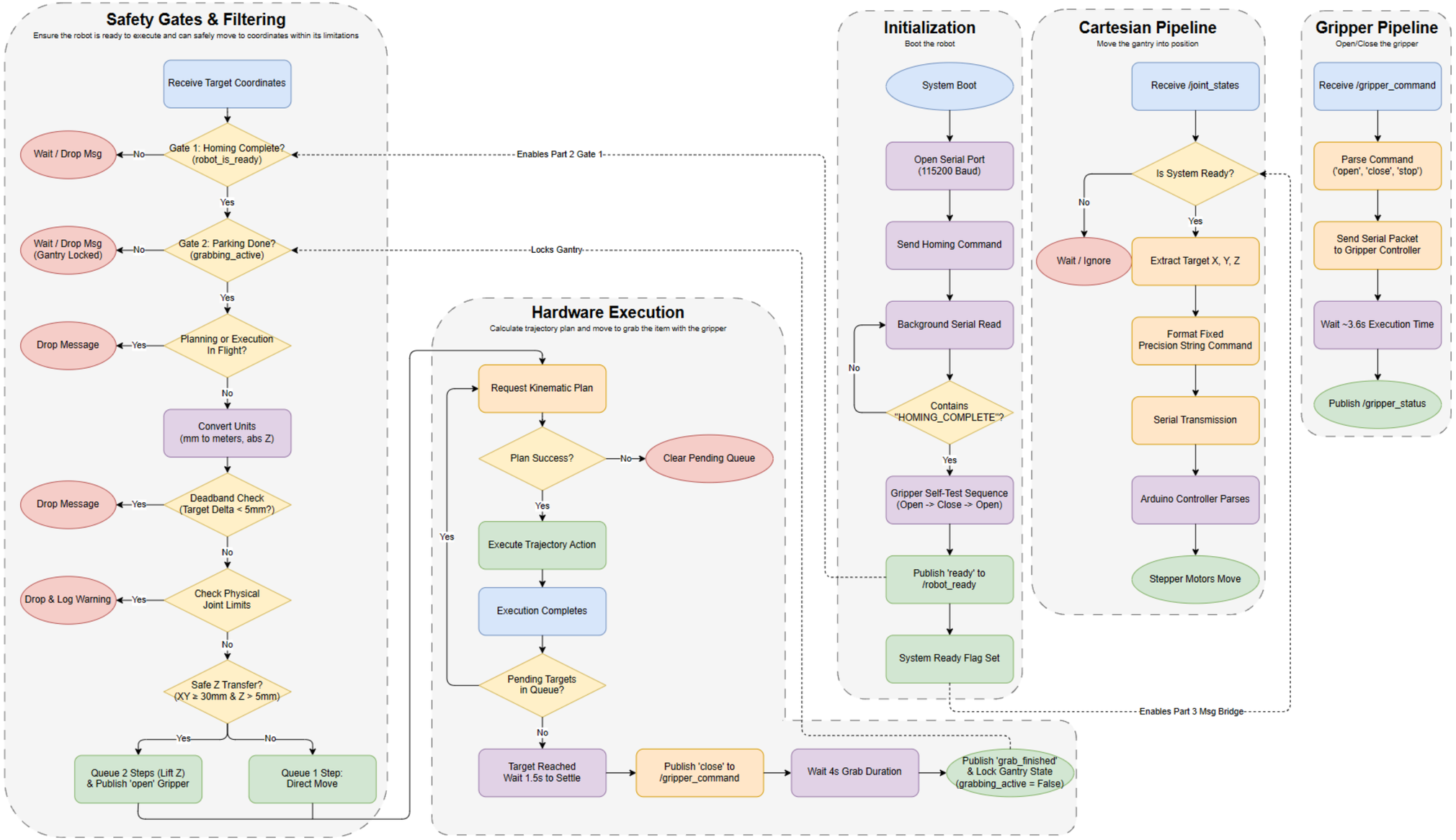

1. Sensors & Inputs
Topic: /camera/.../camera_info
Topic: /camera/.../color/image_raw
Topic: /camera/.../aligned_depth...
Subscribe
Subscribe
Subscribe
2. Vision & Perception
YOLO Node
Publish
Publish
Topic: /yolo/coordinates
Topic: /yolo/color
Applies Transform
Calibration Matrix M
[ 1678.143546, 204.493792, 24.103847 ]
[ -541.626031, 611.540100, -3094.430530 ]
[ -146.380216, 2814.654010, -5655.387667 ]
[ 205.683221, -1321.912211, 3573.158124 ]
Subscribe
3. Motion Planning
Service Client: /plan_kinematic_path
Action Client: /Request Plan*jectory
Trigger Action
YoloToMoveItNode
Subscribe
Subscribe
Topic (Unlock Gate 2): /start_grabbing
Subscribe (System Unlocked)
4. Hardware & Actuation
Topic: /joint_states
Subscribe (XYZ)
Publish
Topic: /gripper_command
Subscribe
Serial Bridge
Publish
Topic: /gripper_status
Publish
Topic: /gantry_task_complete
Publish (Homed/Tested)
Topic (Unlock Gate 1): /robot_ready

# CARTESIAN & GRIPPER OVERVIEW

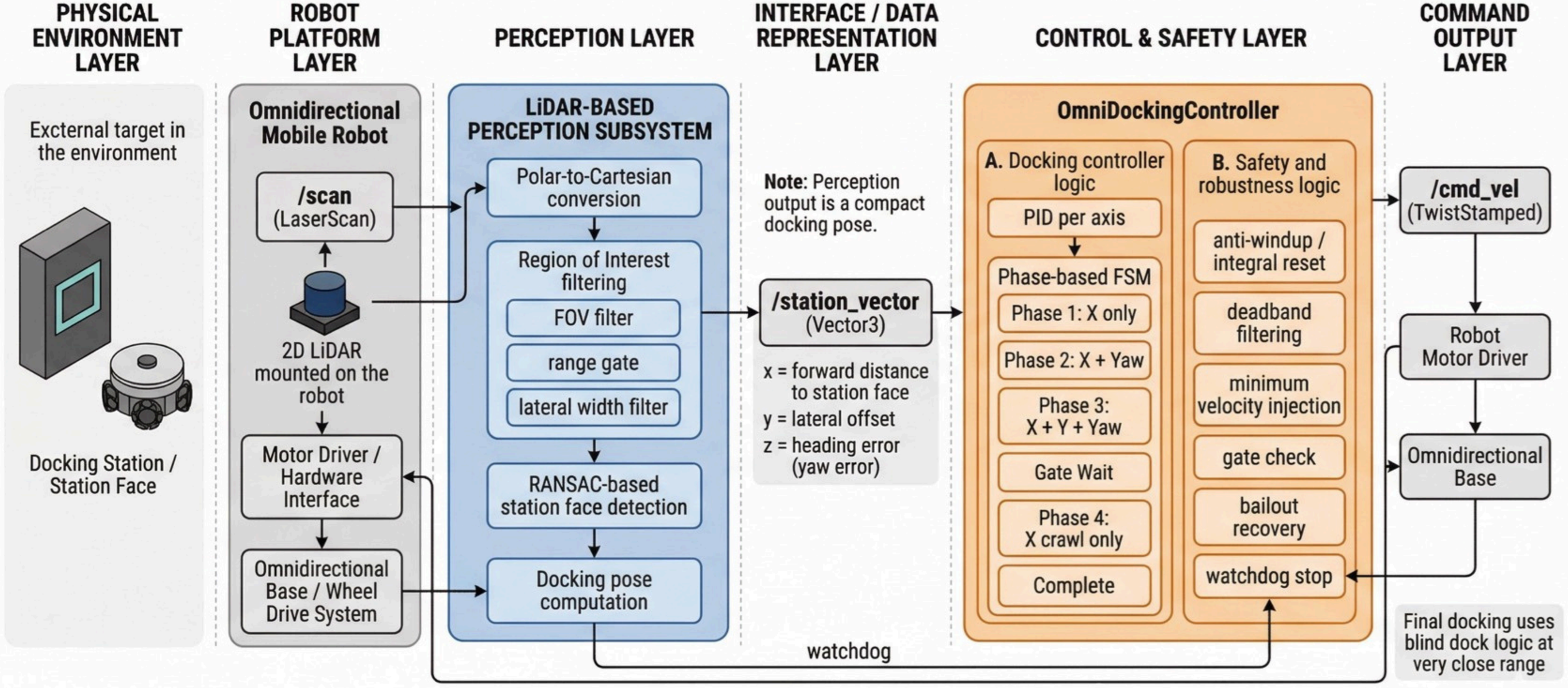
**Flowchart**

## Part 2: Motion Planning & Grabbing

## Part 3: Hardware & Gripper Communication



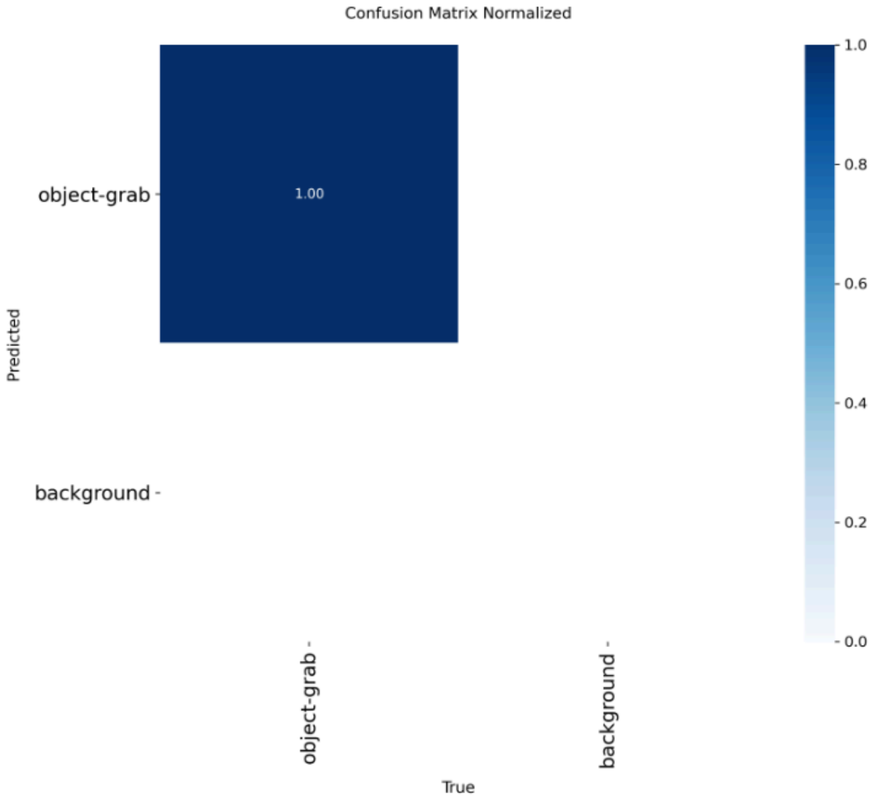
# AUTO PARK OVERVIEW

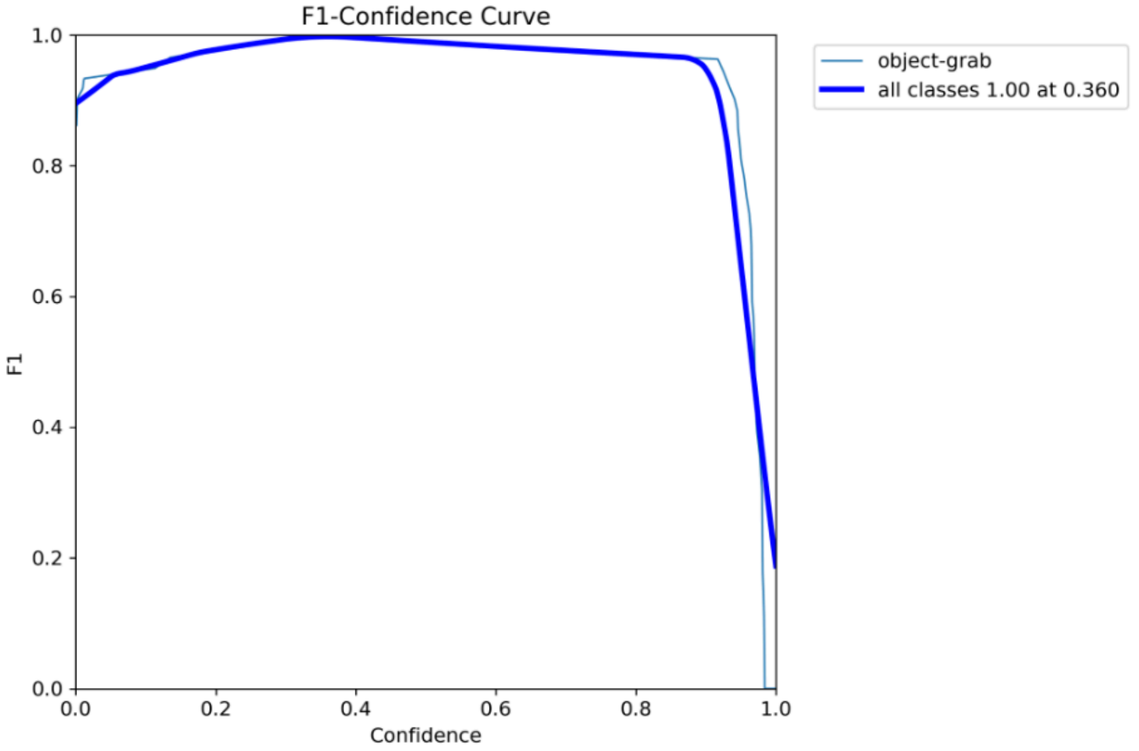



RESULT

# RESULTS

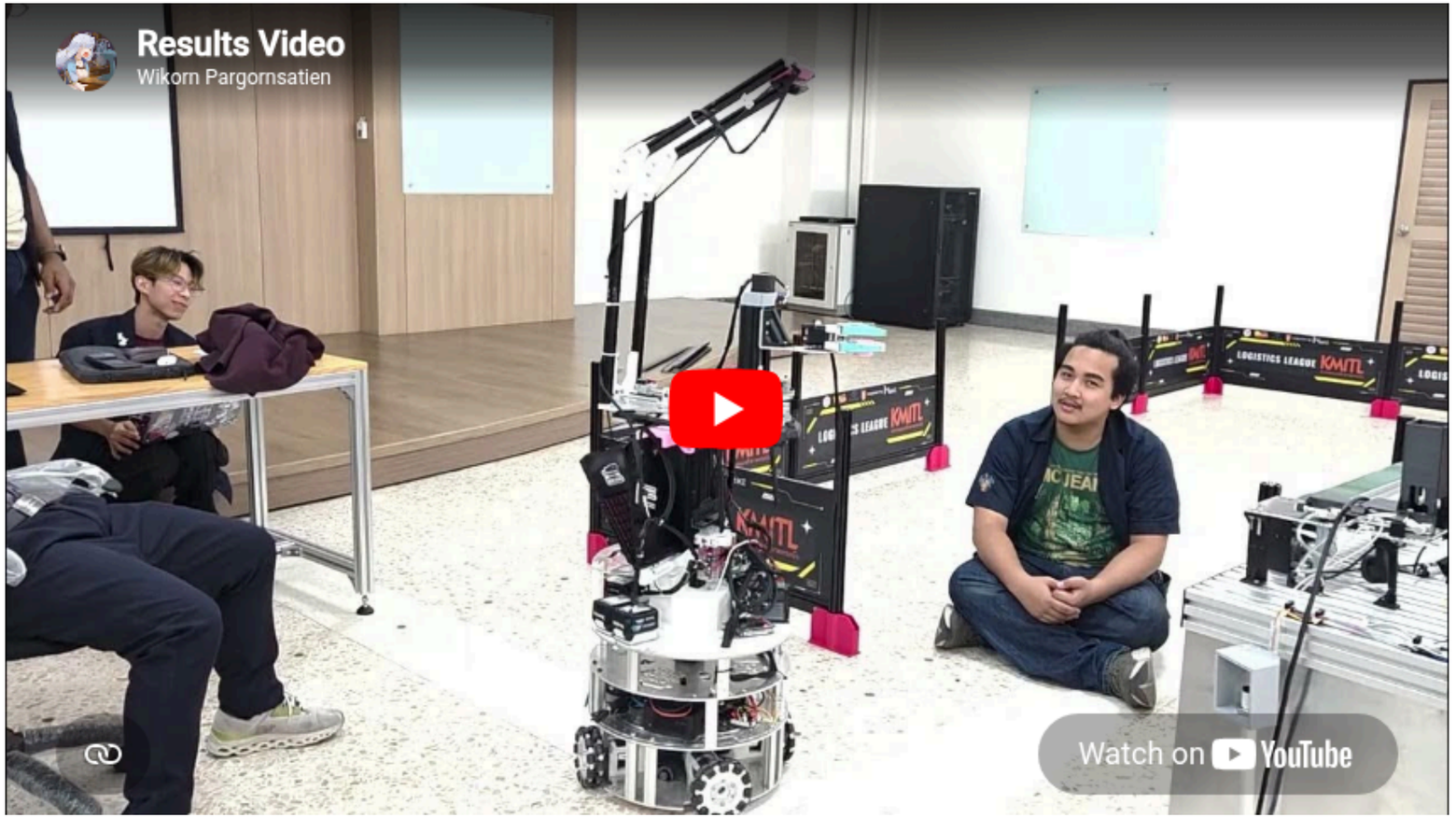
## Yolo Model Performance



# RESULTS

## Video of Result



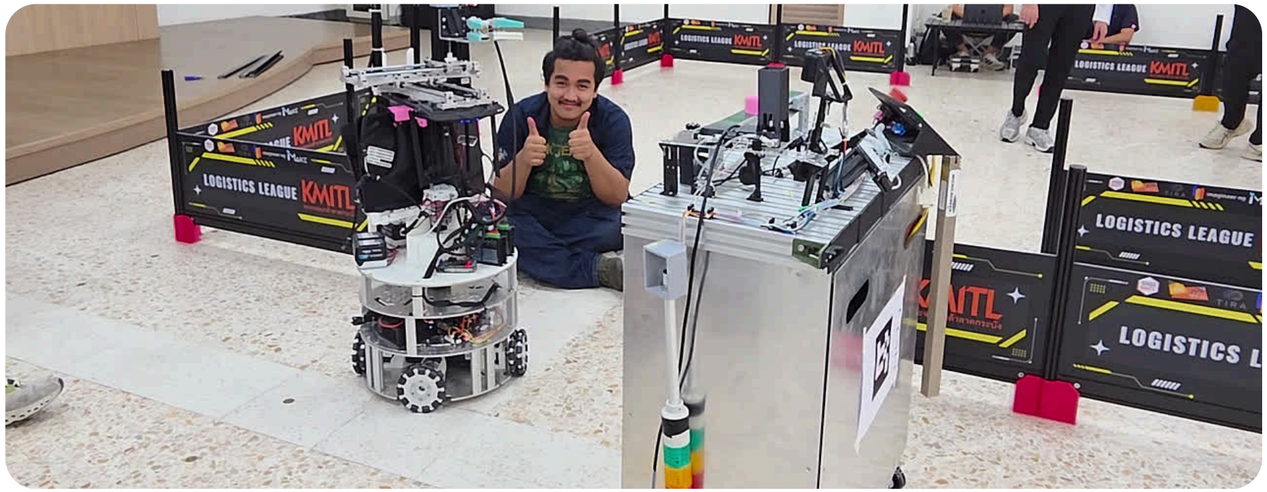
# CONCLUSION

## Closure

- In conclusion, we’ve successfully integrated the Gripper with the Base, successfully implementing a Vision-Guided Cartesian Gripper and LiDAR-Based Auto Parking.

- By unifying the mobile base's LiDAR docking with the Cartesian gantry's vision system under a single ROS 2 state machine, we successfully built a robust, collision-free automated pipeline.

- Thank you for listening and for all the experiences.



Moo Muk Khing

# THANK YOU FOR LISTENING

- 67011743 Thammasorn Srimanee
- 67011383 Wikorn Pargornsatien
- 67011366 Tunwa Raipuang
- 67011698 Lucky Agarwal
- 67011351 Theechutha Suphachitkulchai

- 68011291 Atikhun Chaiwanna
- 68011479 Plainakarin Nalintusnai
- 67011644 Phuree Poopuang