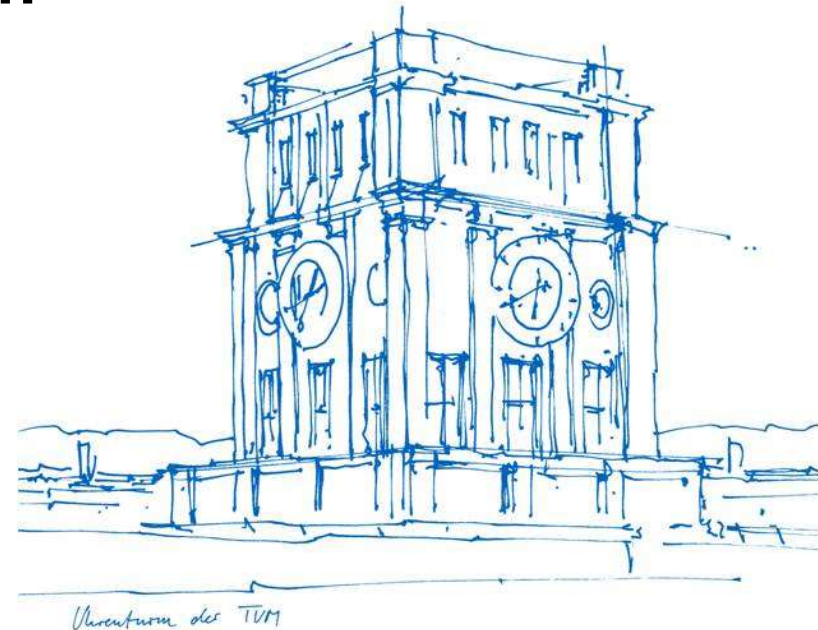


Research and Adaption of Efficient Autonomous Driving methods in Duckietown

Ayush Kumar

Parth Karkar

Servesh Khandwe



Technische Universität München, Campus Heilbronn

Heilbronn, 14th March 2024

Overview

- Mid-term progress
- Demo Video
- Autonomous Navigation in Duckie-town
 - Lane Following
 - Stop Line Detection
 - April-Tags
 - Intersection Navigation
- Observations

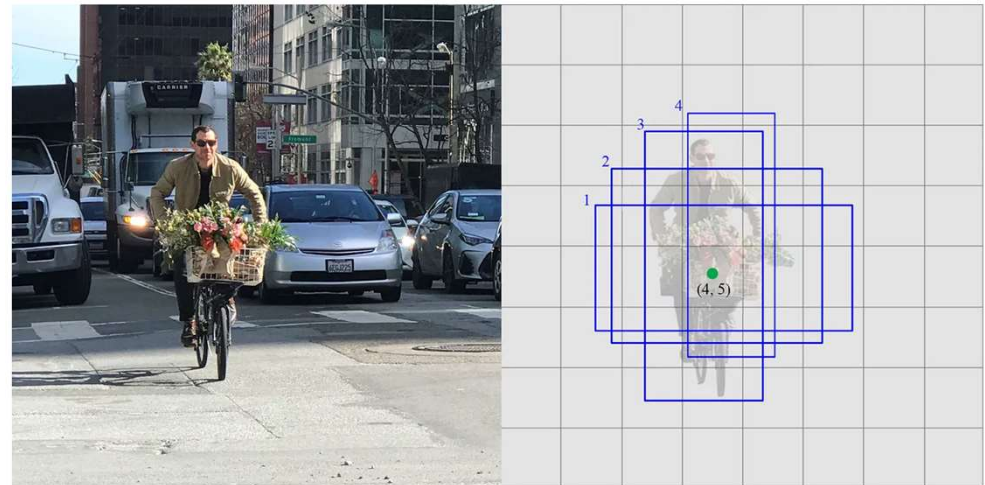


Mid-Term Progress

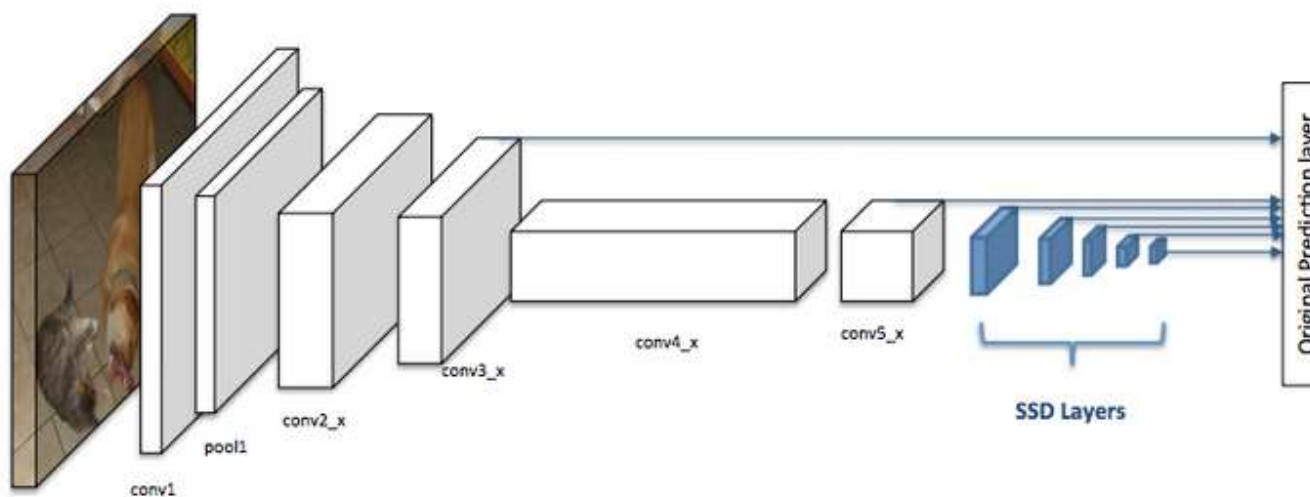


SSD - Single Shot MultiBox Detector

- Single Shot Detection Approach
- Use of Multi-Scale Feature Maps
- Default Boxes and Aspect Ratios



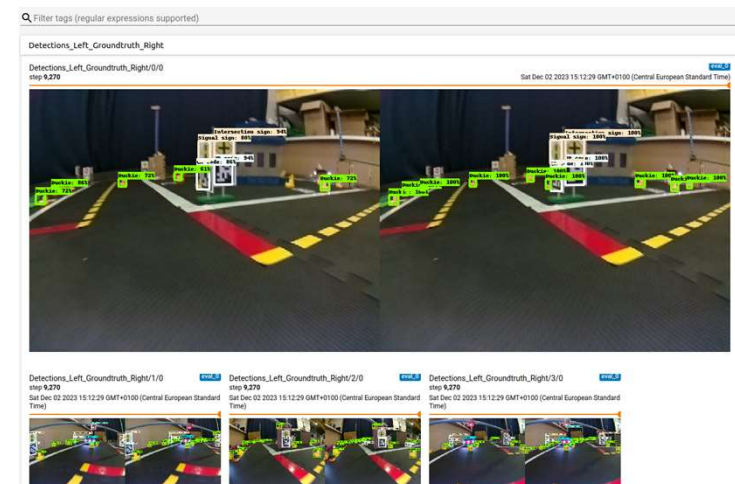
SSD - Single Shot MultiBox Detector



SSD - Single Shot MultiBox Detector

We failed:

- 4 years old repo – was running Google Coral TPU.
- Succeeded to train the model – No inference – unable to communicate with ROS camera node
- Tried re-writing entire camera node Pub-Sub code – in progress

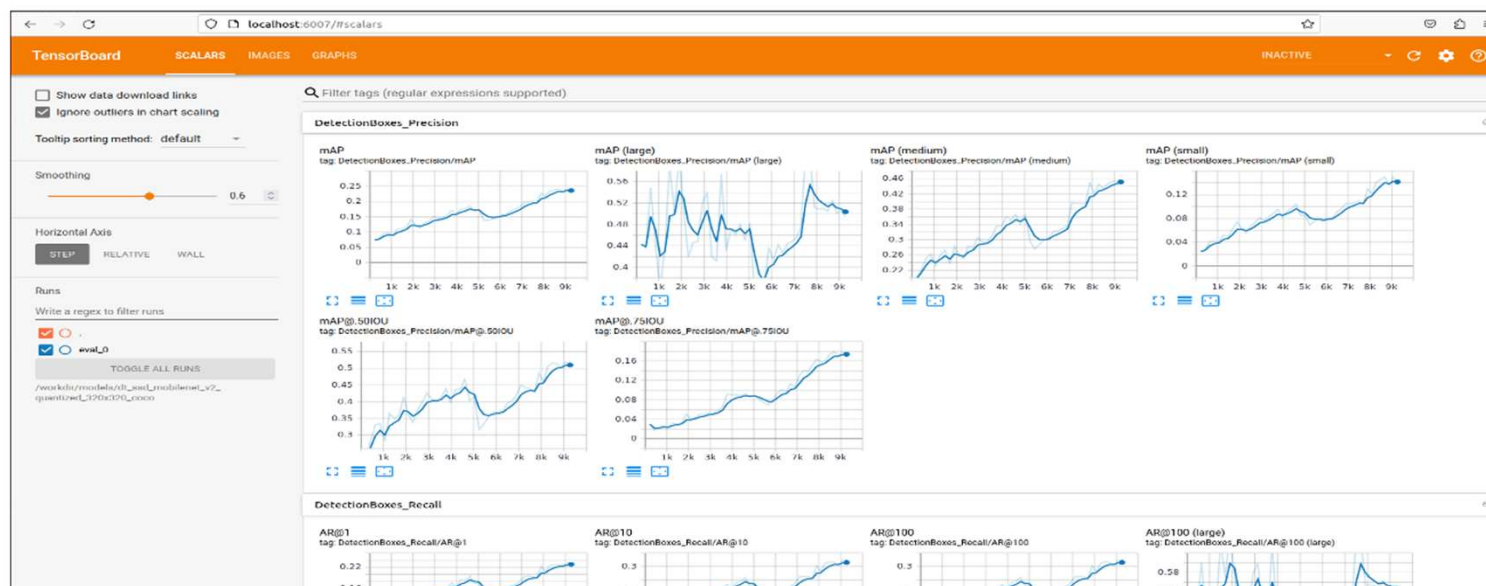


<https://github.com/duckietown-ethz/proj-lfivop-ml>

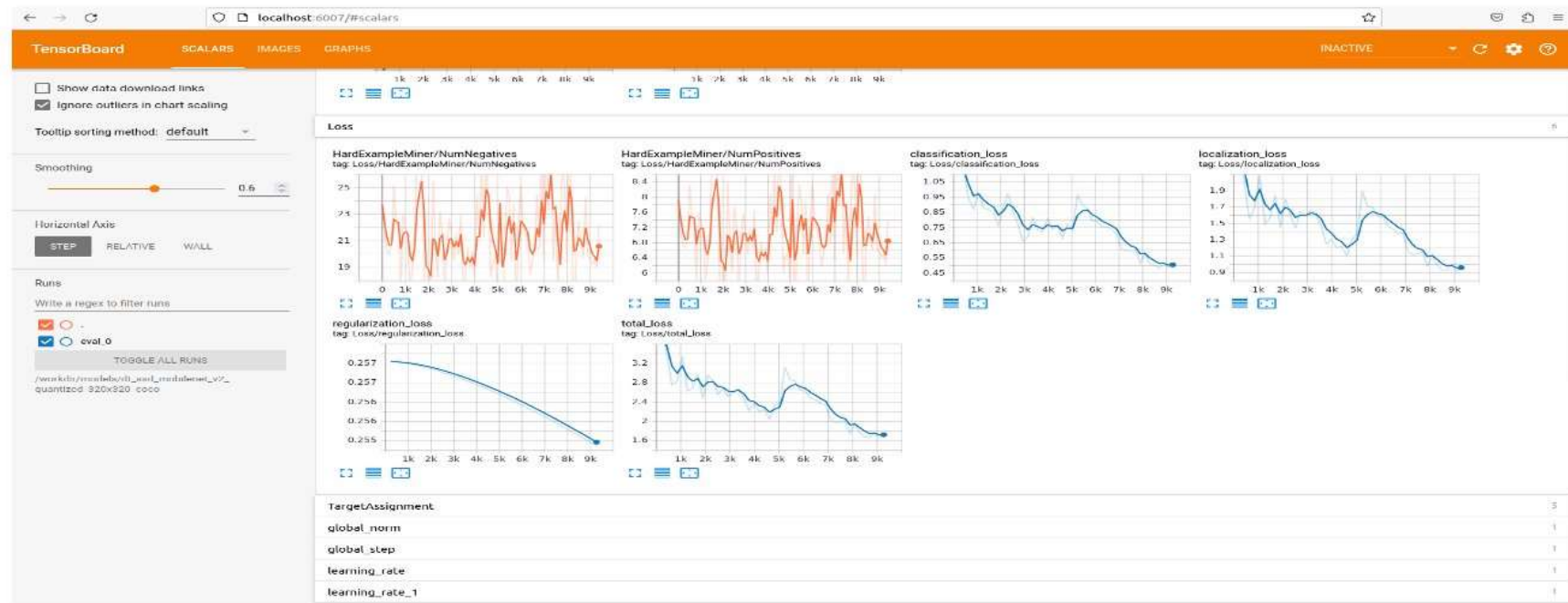
SSD Results: Recall



SSD Results: Precision

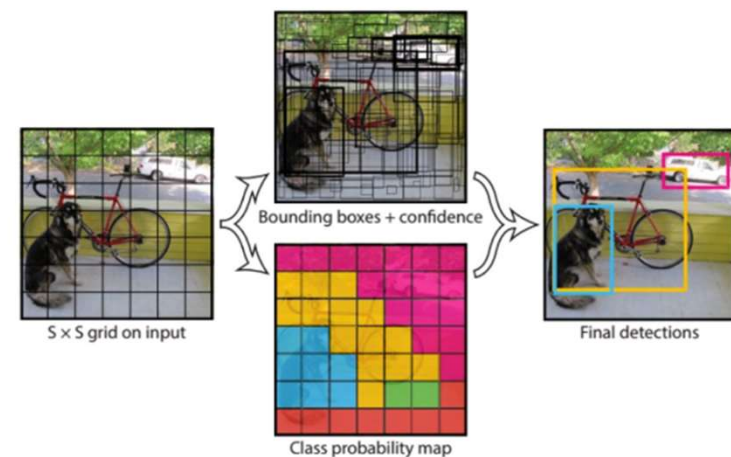


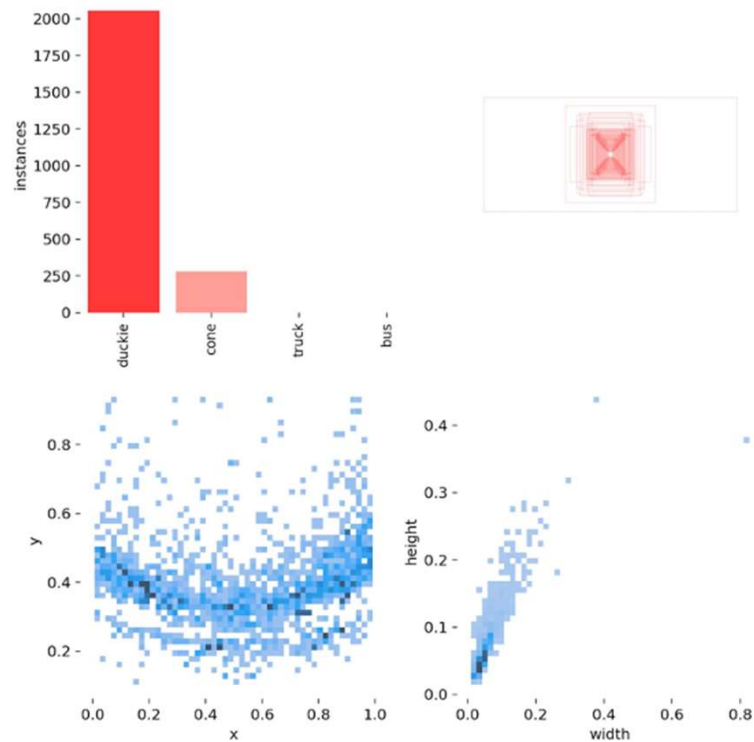
SSD Results: Loss



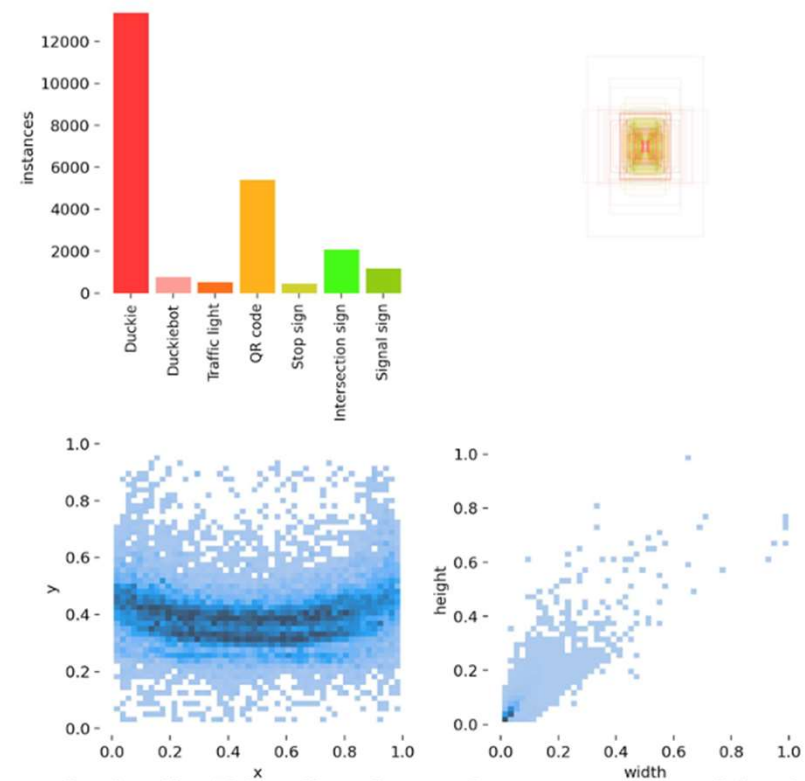
Yolo - You Only Look Once

- Started from scratch again but now using different algorithm
- Our target was to implement end to end feature
- Starting with a Baseline Model
- Transition to an Advanced Model

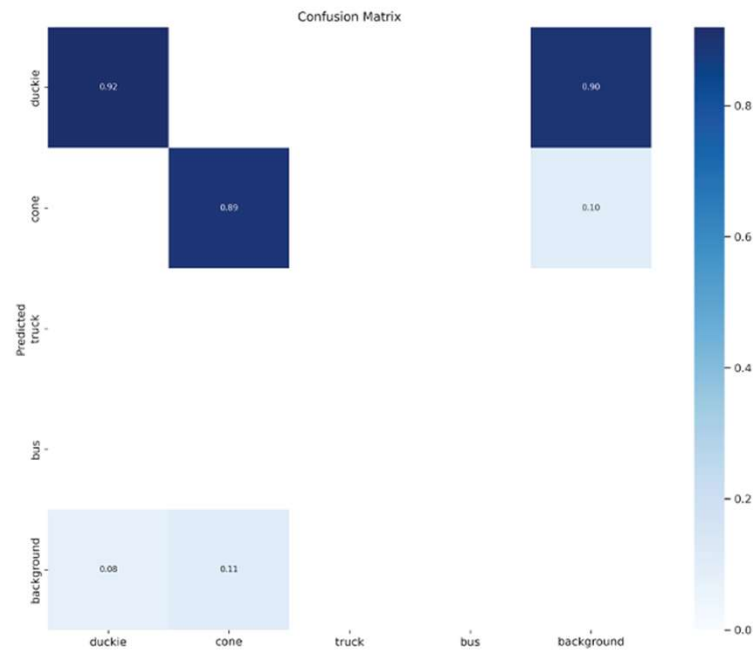




Labels' Distribution of the baseline Model



Label's Distribution of our current Model

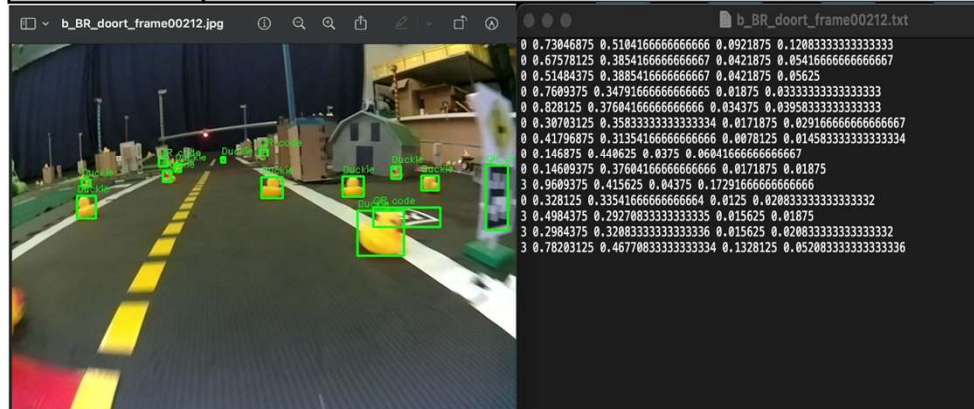


Confusion Matrix of the baseline Model



Confusion Matrix of our current Model

New and improved annotations



Results of our current Model

Collision-Checker

- tried to Implement Collision checker
- No map Definition
- Therefore shifted to ToF Approach

```

collision_checker.py - collision-checker - Visual Studio Code
File Edit Selection View Go Run Terminal Help
EXPLORER
  COLLISION-CHECKER
  > .vscode
  > assets
    > images
      dtlogo.png
      env18-result.png
      env18.png
    ! environment.yaml
  > docs
  > notebooks/01-Collision-Chec...
  > collision_checker.ipynb
  > packages
    > collision_checker
      __init__.py
      collision_checker.py
      environment_mana...
      geometry_utils.py
      simulation_controll...
    > tests
    > README.md
  > OUTLINE
  > TIMELINE
  packages > collision_checker > collision_checker.py > ...
86         raise NotImplementedError("Collision check not implemented for given shape types")
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
def circle_circle_collision(circle1: Circle, circle2: Circle) -> bool:
    dx = circle1.center.x - circle2.center.x
    dy = circle1.center.y - circle2.center.y
    distance = (dx**2 + dy**2)**0.5
    return distance < (circle1.radius + circle2.radius)

def rectangle_circle_collision(rectangle: Rectangle, circle: Circle) -> bool:
    closest_x = max(rectangle.left, min(circle.center.x, rectangle.right))
    closest_y = max(rectangle.bottom, min(circle.center.y, rectangle.top))
    dx = circle.center.x - closest_x
    dy = circle.center.y - closest_y
    return (dx**2 + dy**2) < (circle.radius**2)

def rectangle_rectangle_collision(rect1: Rectangle, rect2: Rectangle) -> bool:
    return (rect1.left < rect2.right and rect1.right > rect2.left and
            rect1.bottom < rect2.top and rect1.top > rect2.bottom)

def apply_rototranslation(primitive, pose):
    # Apply rototranslation based on the type of primitive
    if isinstance(primitive, Circle):
        # Assuming Circle has a center attribute of type Point
        new_center = rotate_and_translate_point(primitive.center, pose)
        return Circle(center=new_center, radius=primitive.radius)
    elif isinstance(primitive, Rectangle):
        # Assuming Rectangle has a center attribute of type Point
        new_center = rotate_and_translate_point(primitive.center, pose)
        new_angle = primitive.angle + pose.angle # Adjust if Rectangle has an angle attribute
        return Rectangle(center=new_center, width=primitive.width, height=primitive.height, angle=new_angle)
    else:
        raise NotImplementedError("Primitive type not supported for rototranslation.")

def rotate_and_translate_point(point, pose):
    # Rotate a point around the origin (0, 0) by an angle and then translate it
    new_x = math.cos(pose.angle) * point.x - math.sin(pose.angle) * point.y + pose.x
    new_y = math.sin(pose.angle) * point.x + math.cos(pose.angle) * point.y + pose.y
    return Point(new_x, new_y) # Use the Point class from dt_protocols.collision_protocol
  
```


Collision-Checker

- Implemented the **publish-subscribe** pattern using **ROS Publishers** and **ROS Subscribers**
- Succeeded but sometimes not receiving incorrect data

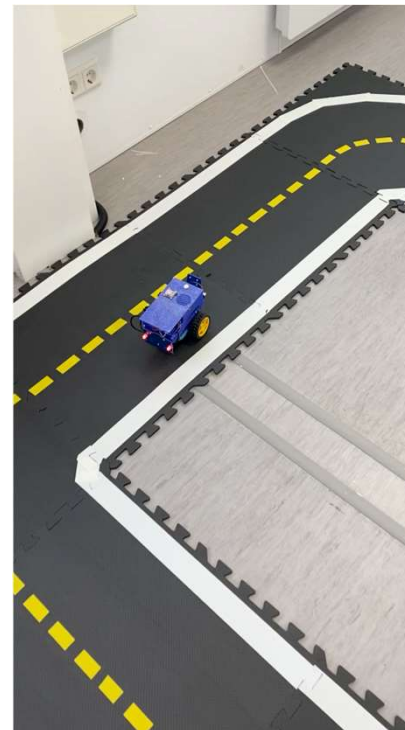


Demo video



Demo Video

**Previously in
Duckietown...**



Demo Video

And **now** in
Duckietown...



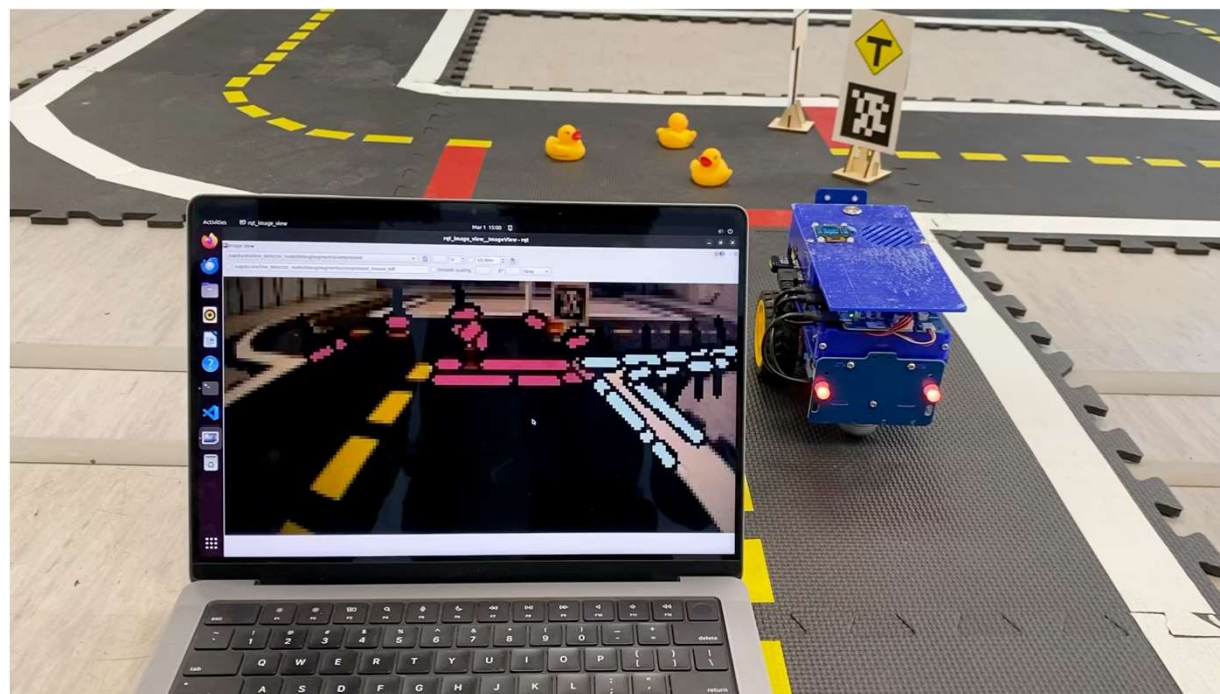
Demo Video

And **NOW** in
Duckietown...

```

/sapduckie/apriltag_detector_node/detections/image/compressed
/sapduckie/camera_node/image/compressed
/sapduckie/camera_node/image/raw
/sapduckie/ground_projection_node/debug/ground_projection_image/compressed
/sapduckie/line_detector_node/debug/edges/compressed
/sapduckie/line_detector_node/debug/maps/compressed
/sapduckie/line_detector_node/debug/ranges_H5
/sapduckie/line_detector_node/debug/ranges_HV
/sapduckie/line_detector_node/debug/ranges_SV
/sapduckie/line_detector_node/debug/segments/compressed
/sapduckie/rectifier_node/image/compressed
/sapduckie/vehicle_detection_node/debug/detection_image/compressed

```

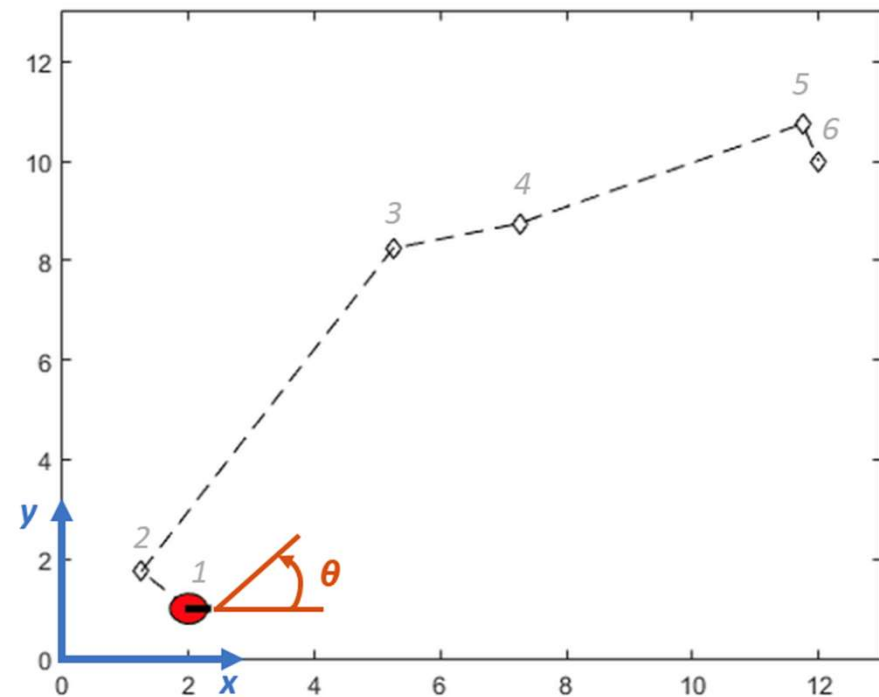


Lane Following

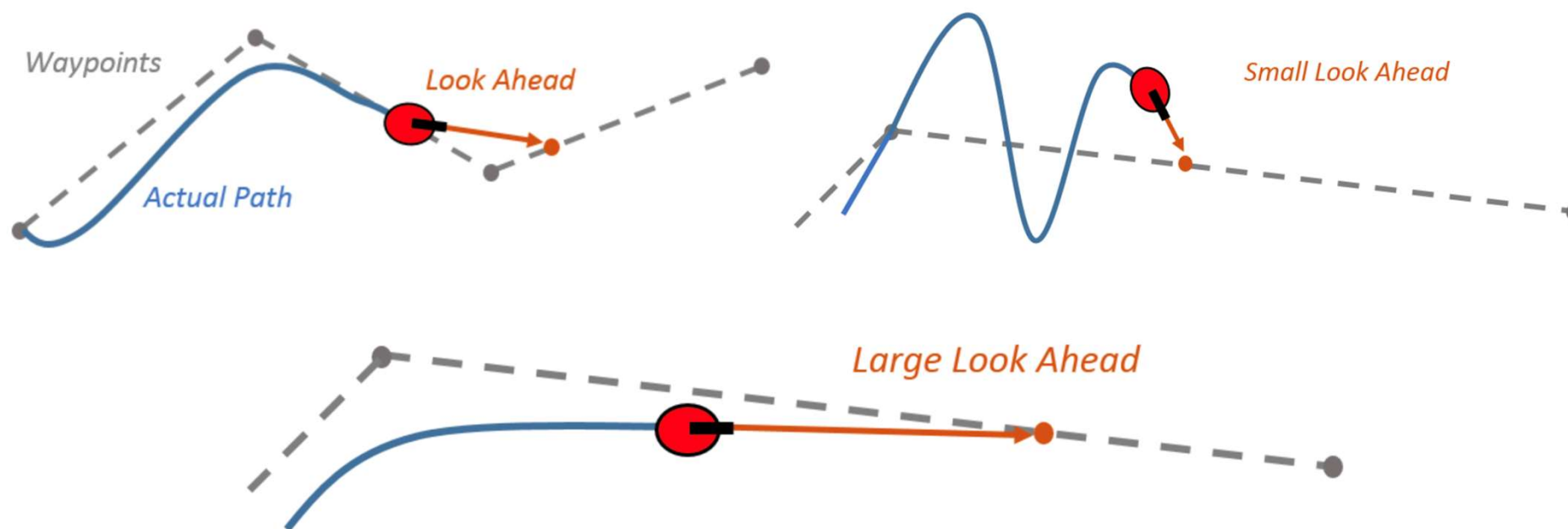


Lane Following

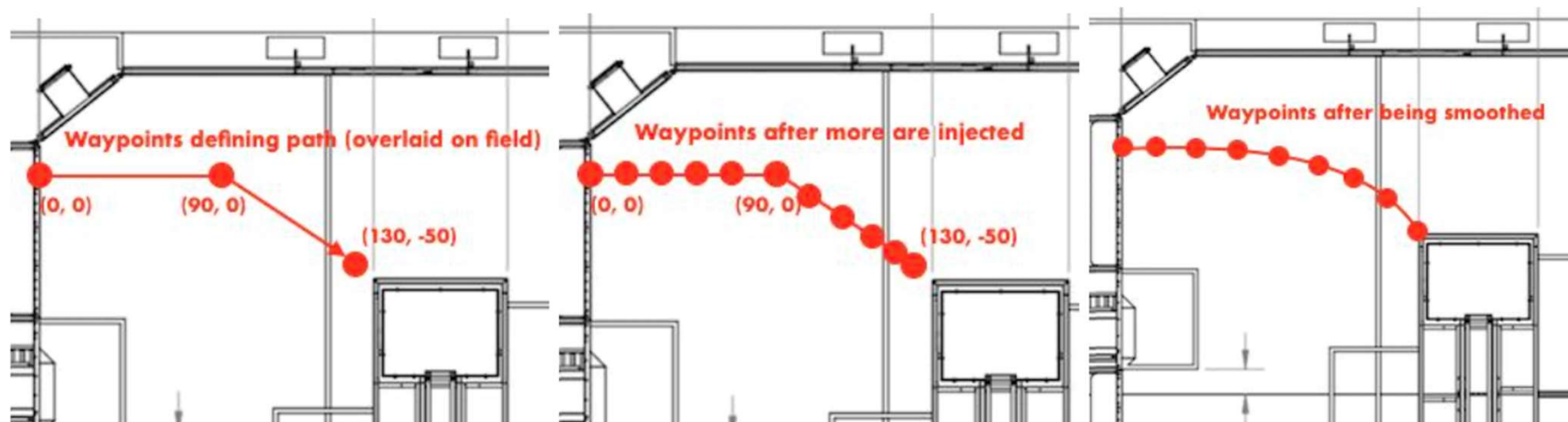
- Pure Pursuit Algorithm
- General purpose tracking algorithm



Lane Following



Lane Following



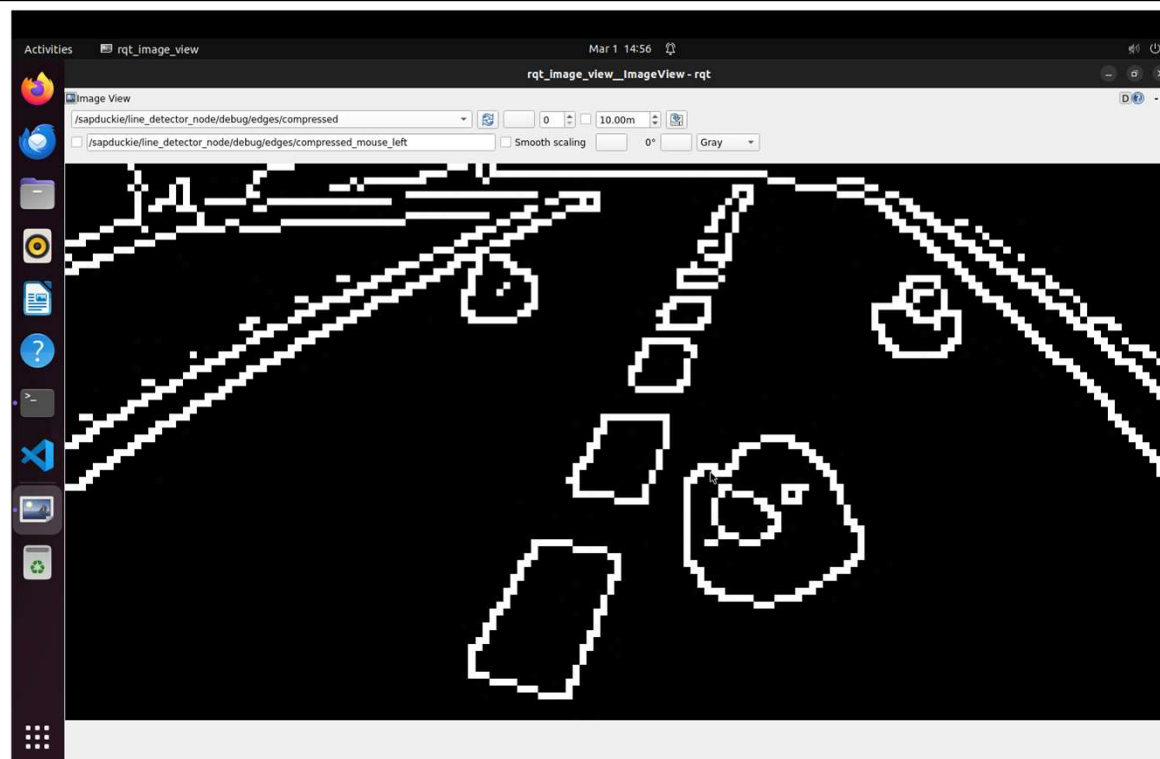
Lane Following

Modification in Pure Pursuit Algorithm:

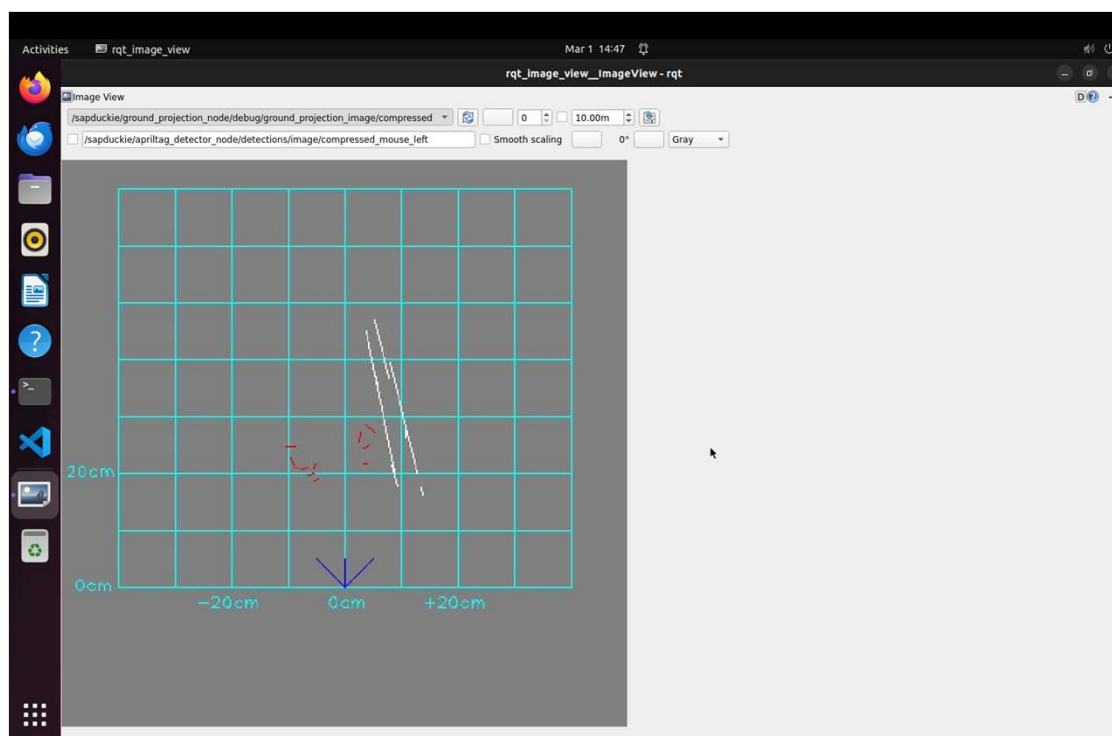
- Yellow lane points to the right
- If yellow lane not visible then offset ground-projected white lane to the left
- We then take average of this offset



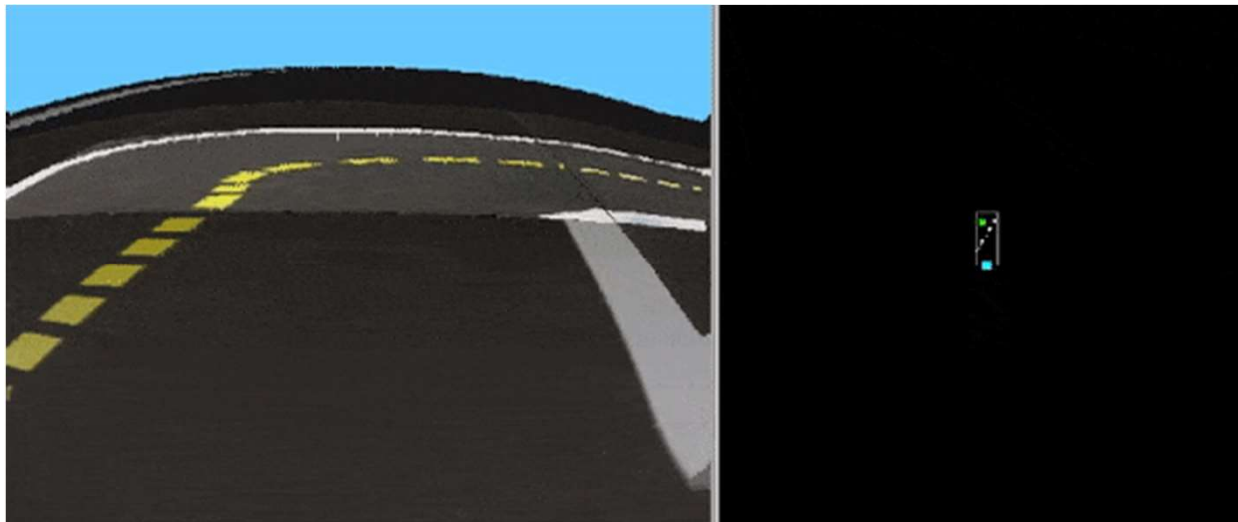
Lane Edge Detection



Ground Projection



Lane Following



Lane Following

Modification in Pure Pursuit Algorithm:

- Varying speeds and Omega gains
- Gradual speed-up occurs on straight paths
- Slows down at turns for smoother turns



Lane Following

Modification in Pure Pursuit Algorithm:

- Modified Lane Filter
 - At each update step, we calculate the time elapsed since the last update.
 - Dynamically scale the variance of the Gaussian used for smoothing
- Adaptive Scaling



Lane Following

Limitation:

- The controller cannot exactly follow direct paths between waypoints
- Parameters must be tuned to optimize the performance and to converge to the path over time.
- This pure pursuit algorithm does not stabilize the robot at a point.
- In our application, a distance threshold for a goal location should be applied to stop the robot near the desired goal.

Stop Line Detection



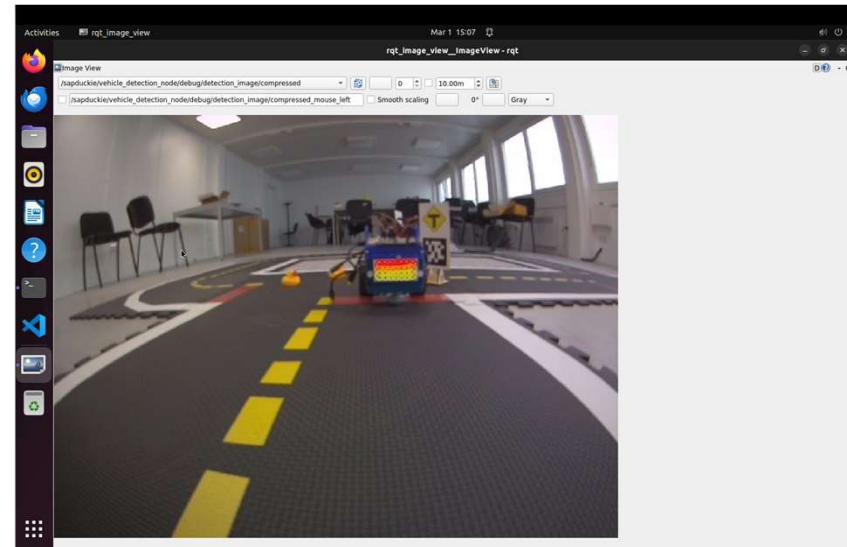
Stop Line Detection

- Identifying the colour coordinates of the line ahead of the camera,
- If we see **red line**, then we should stop at the safe distance.
- Stop is really import for car to take time to determine turns.



Obstacle-Vehicle Detection

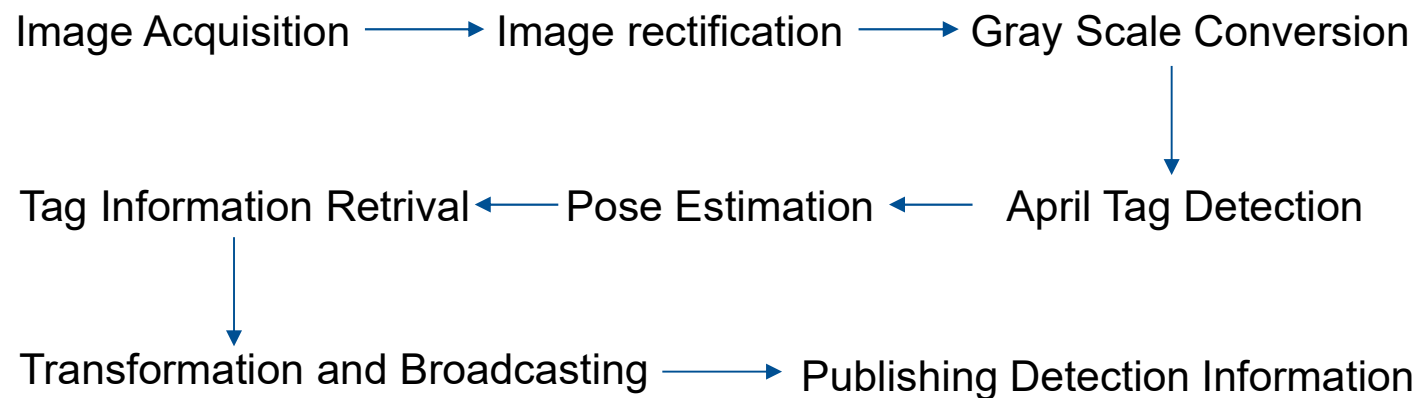
- Similarly, we need to stop when we have a bot Infront of us to avoid colliding with that, so bot detection is also very crucial for that.



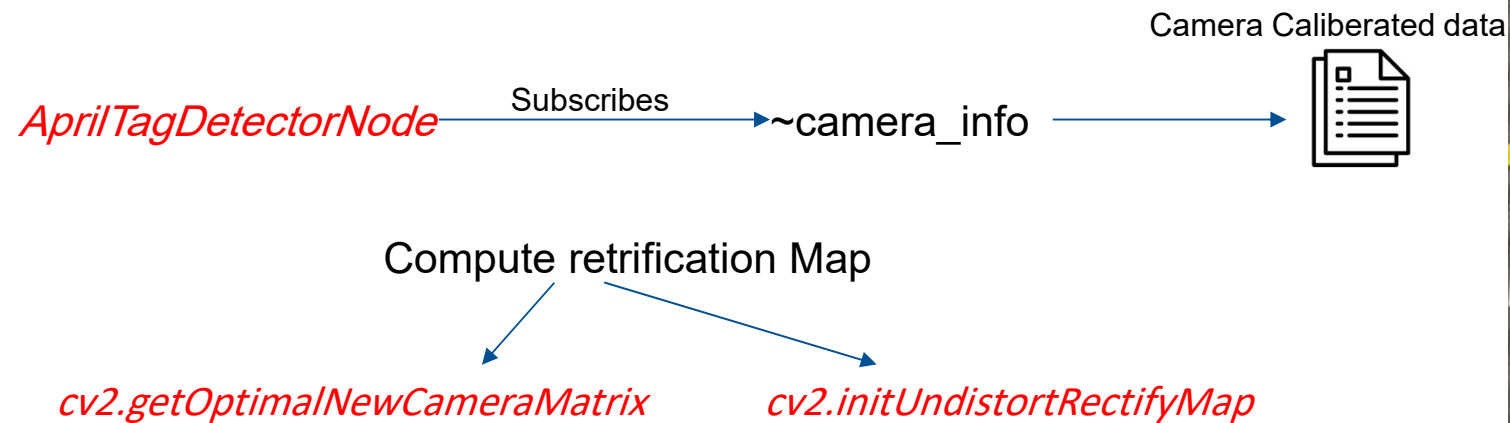
April - Tags



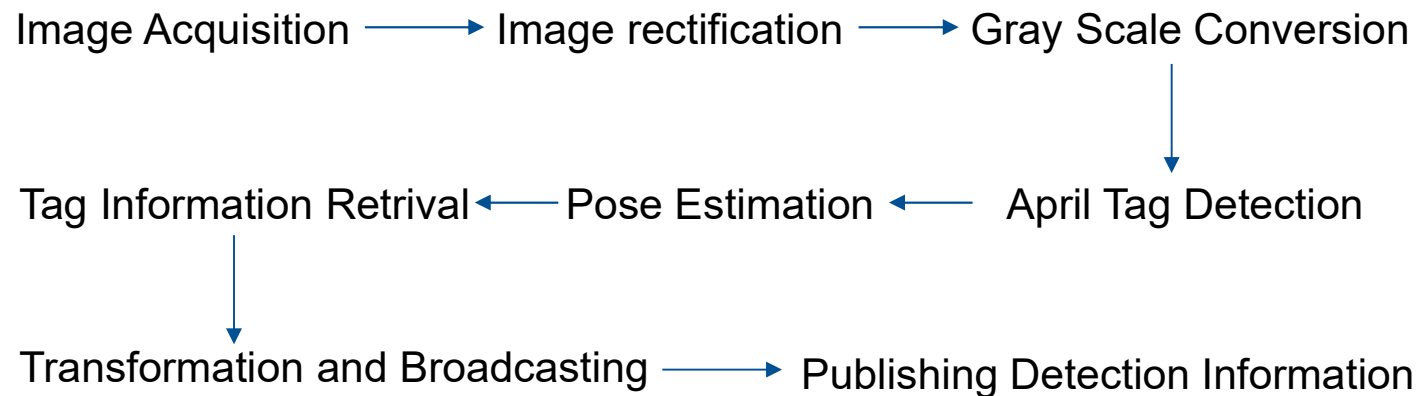
How April Tag works?



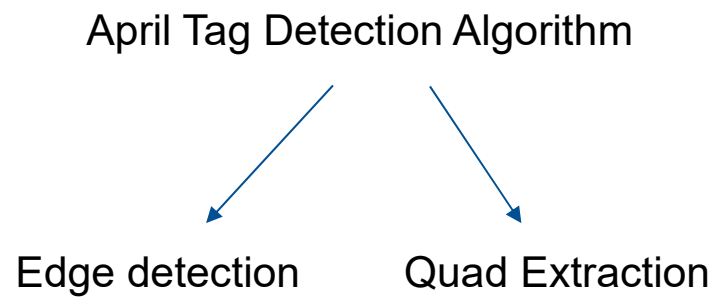
How April Tag works?



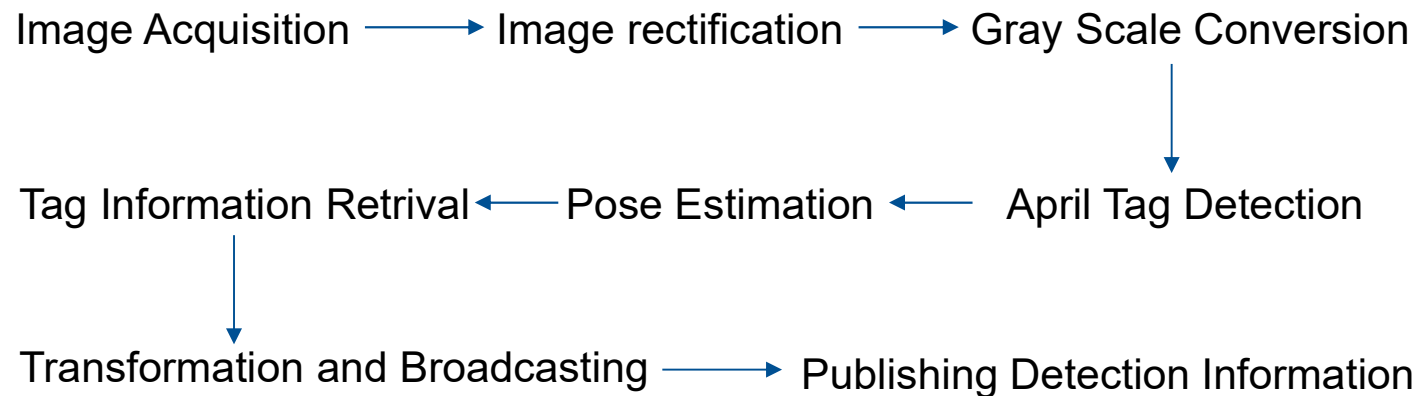
How April Tag works?



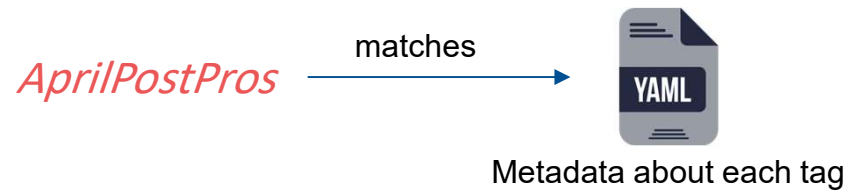
How April Tag works?



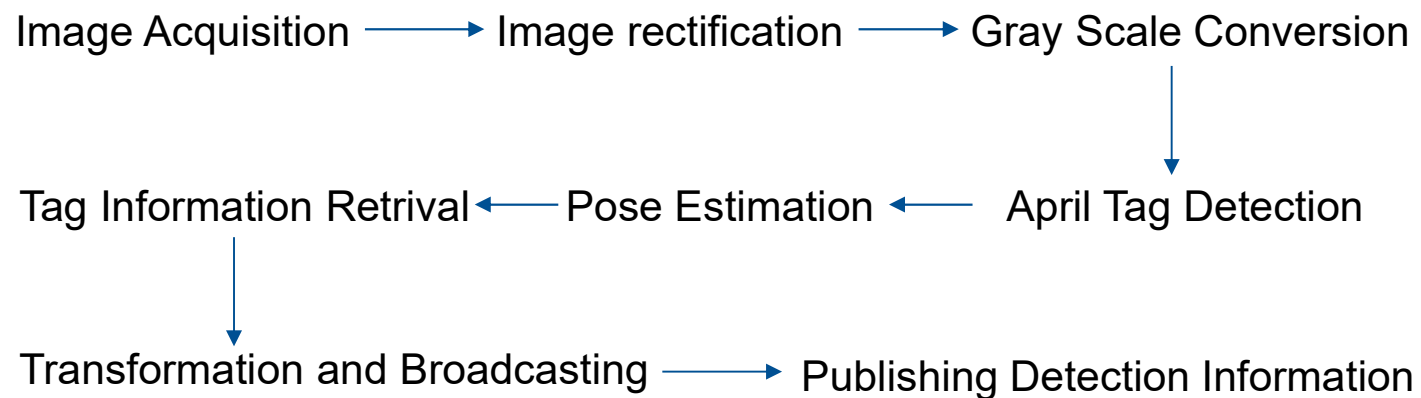
How April Tag works?



How April Tag works?



How April Tag works?

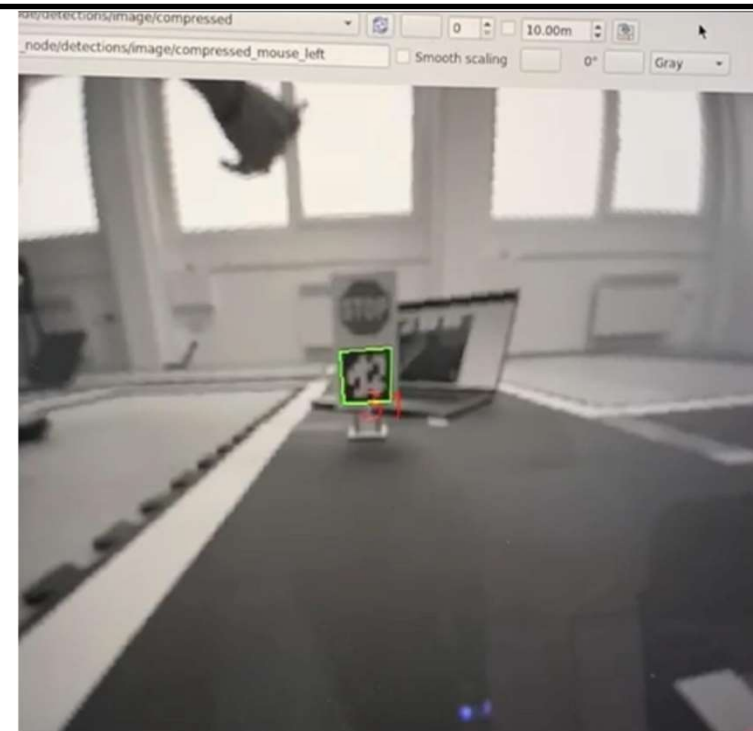


Intersection Navigation



Intersection Detection

- April Tags
- April Tags IDs
 - 0 for Left
 - 1 for Straight
 - 2 for Right



Intersection Detection

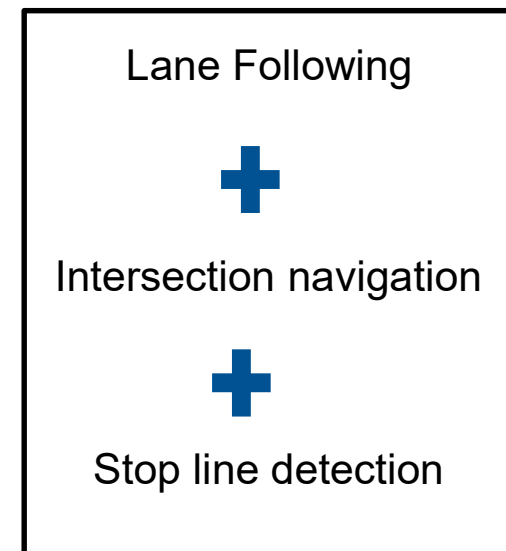


Harmonizing all the modules

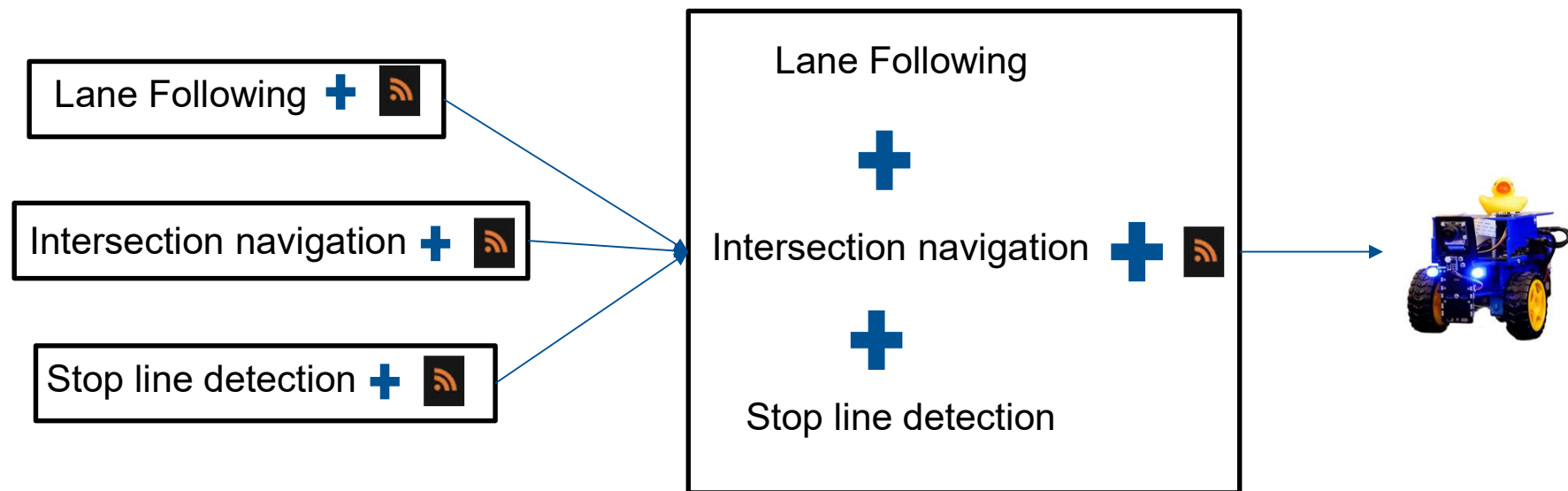


Harmonizing all the modules

Real Time Synchronisation
Seamless Collaboration



Harmonizing all the modules



 = respective launch file

Observations



Observation

- When running all the modules together, we hit the maximum CPU usage.
- After we tried a lot of things, memory usage of the bot was over 90%.
- We could not utilize GPU capacity of the Jetson.
- Lidar sensor we wanted to use didn't had proper port to connect.

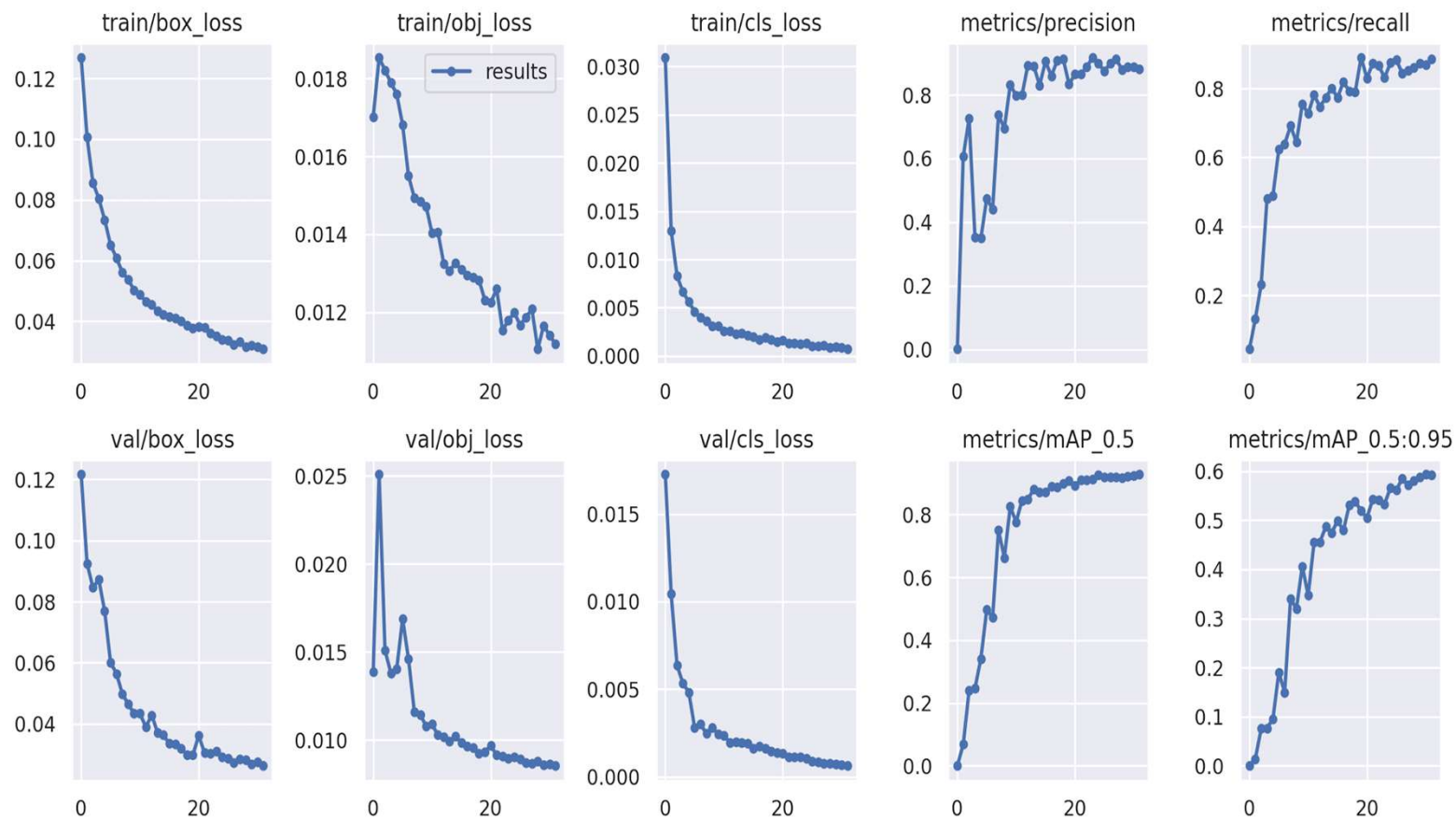




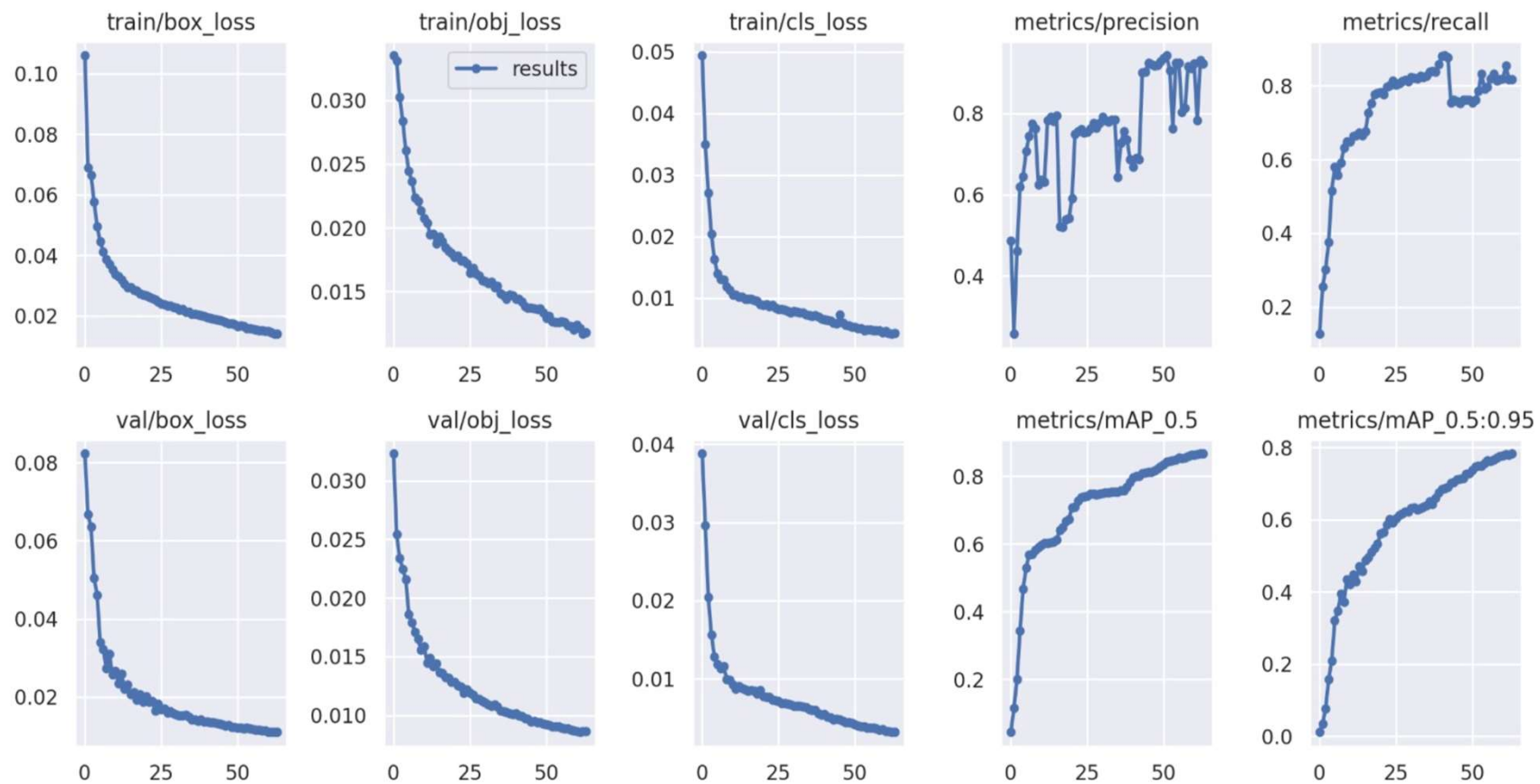
Thank You

Back-up Slides





Results of our baseline Model



Results of our current Model

Validating runs/train/exp/weights/best.pt...

Fusing layers...

YOLOv5n summary: 157 layers, 1768636 parameters, 0 gradients, 4.2 GFLOPs

Class	Images	Instances	P	R	mAP50	mAP50-95: 100% 11/11
all	641	5912	0.924	0.818	0.867	0.783
Duckie	641	3318	0.971	0.978	0.993	0.885
Duckiebot	641	193	0.979	0.995	0.995	0.96
Traffic light	641	140	0.953	0.957	0.983	0.831
QR code	641	1353	0.975	0.988	0.993	0.897
Stop sign	641	112	0.975	0.0357	0.303	0.275
Intersection sign	641	503	0.886	0.974	0.981	0.89
Signal sign	641	293	0.726	0.797	0.819	0.743

Results saved to runs/train/exp

Collision-Checker

- Implemented the publish-subscribe pattern using **ROS Publishers** and **ROS Subscribers**
- Succeeded but sometimes not receiving incorrect data



Future Plans

- Quickly finish basic functionalities such as:
- State Estimation and localization
- Planning and Control
- Agent
- Research and implementation of our final goal from next year.

References

- <https://www.mathworks.com/help/nav/ug/pure-pursuit-controller.html>