Integrated Vision-Physics-Reinforcement Learning Framework for Dynamic Industrial Robot Navigation

Team

Benyamain yacoob
author
yacoobby@udmercy.edu

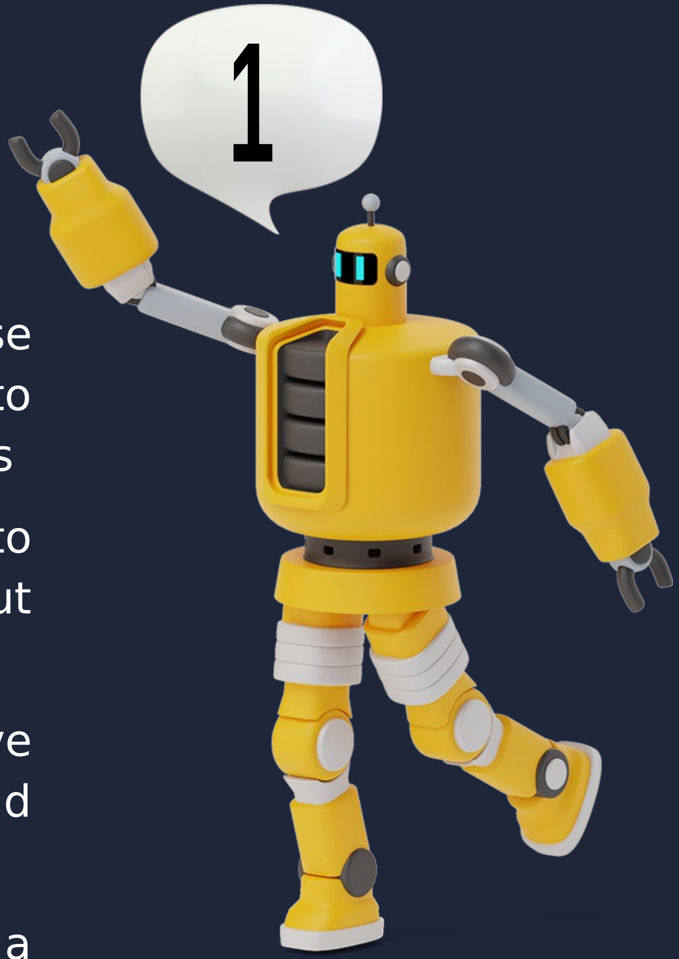
xinyang zhang
Author
zhangxi24@udmercy.edu

Jingyuan wang
Author
wangji15@udmercy.edu

Mina maleki
advisor
malekimi@udmercy.edu

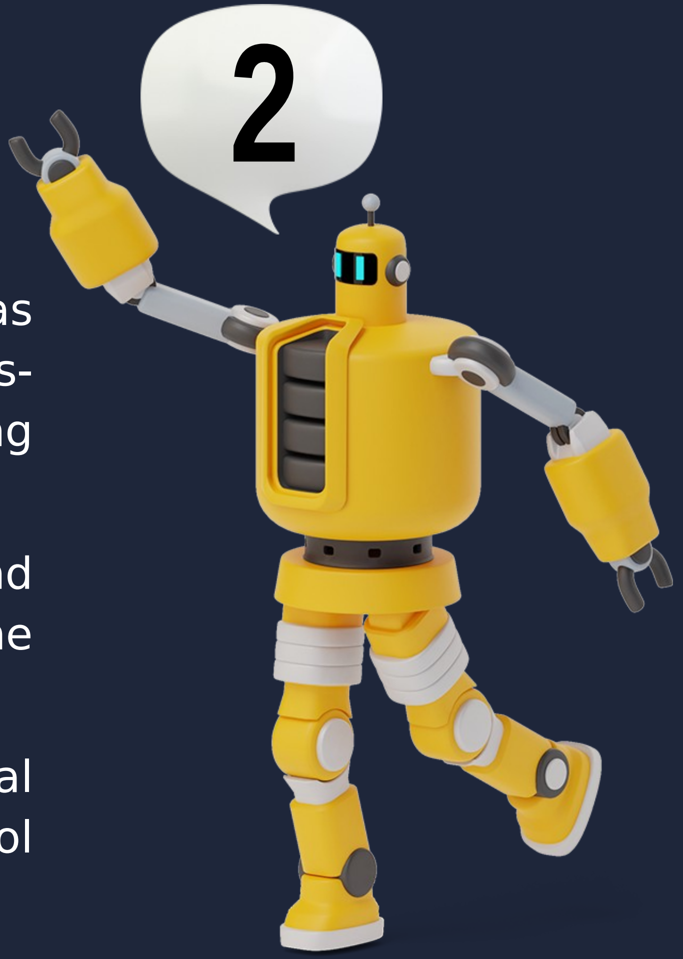
Problem & Motivation

- Dynamic industrial environments pose challenges for mobile robot navigation due to unpredictable obstacles and changing layouts
- Traditional navigation methods struggle to adapt to these dynamic conditions without extensive reprogramming
- Recent research has explored adaptive techniques, integrating vision, physics, and reinforcement learning (RL)
- Our work addresses these challenges with a novel integrated framework combining vision-



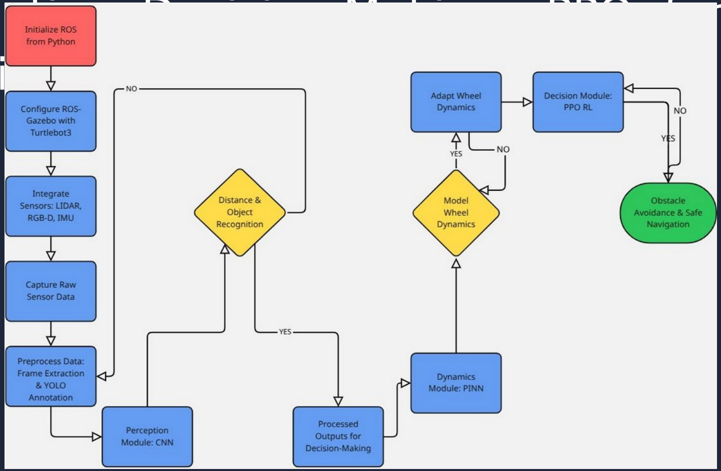
Related literature

- Recent research in robot navigation has focused on vision-based perception, physics-informed control, and reinforcement learning approaches
- Vision-based navigation uses cameras and computer vision algorithms to perceive the environment and make navigation decisions
- Physics-informed control incorporates physical constraints and models into the control system to improve the robot's movement
- Reinforcement learning uses rewards and



Research Focus: CNN-PPO-PINN Architecture

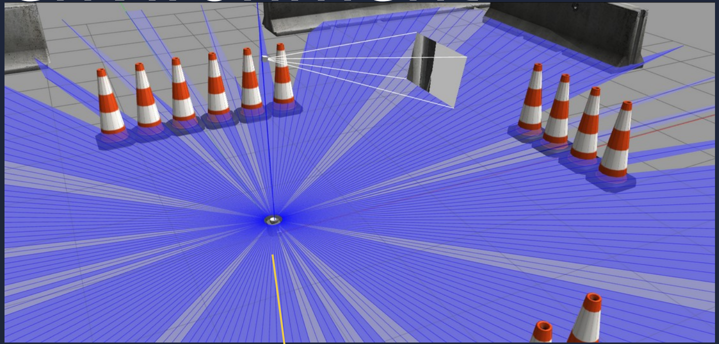
- Vision-Based Perception: CNN for object detection and distance estimation
- Integrated Framework: Combines CNN, PINN, and PPO for enhanced navigation in dynamic environments
- Physics-Informed Control: PINN for precise wheel dynamics control
- Ada real-time deci



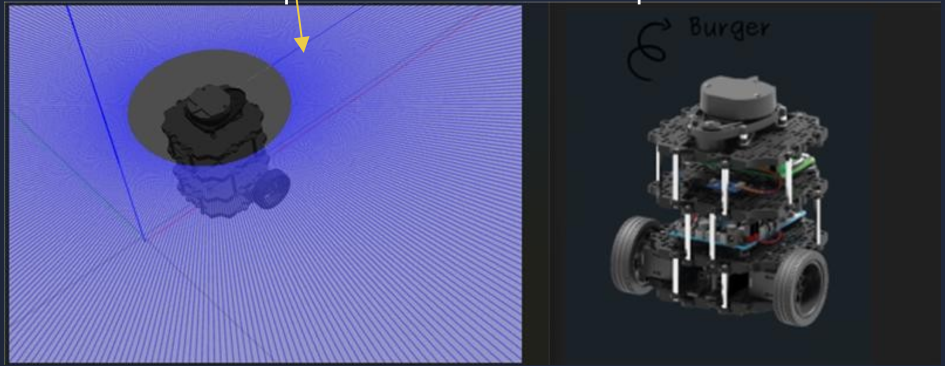
Methodology: Integrated Vision, Physics, and RL Framework

- CNN: Detects target objects and estimates distances from RGB images and depth maps
- PINN: Controls wheel torque via physically-constrained dynamics
- PPO: Learns optimal navigation strategies through continuous state-action feedback

experimental environment

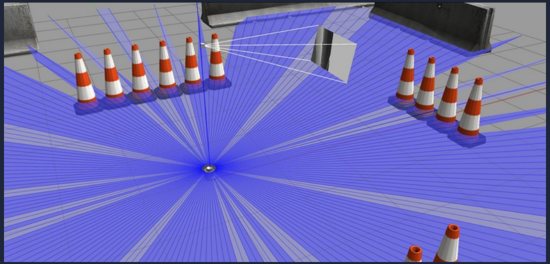


TurtleBot3 robotic platform used in our experiments



Gazebo: These components are demonstrated working in concert within a ROS2 Humble and Gazebo Classic simulation environment, showcasing the framework'

data Format



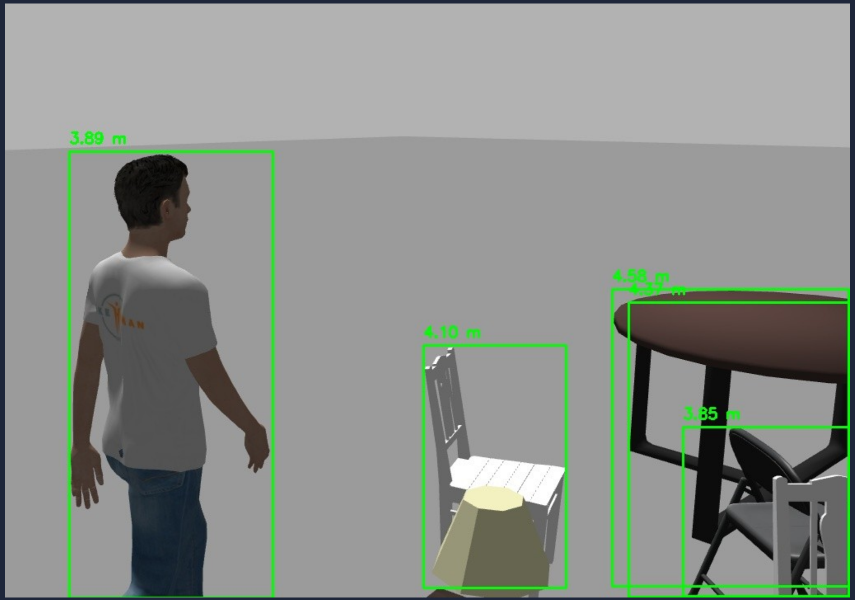
The dataset includes sensor readings LiDAR, RGB/Depth camera images. Using YOLOv5l6 model to generate bounding boxes and categories, and extended the label format to include target distances

3	0.322656	0.720833	0.107813	0.297222	4.69
3	0.897656	0.374306	0.087500	0.206944	10.00
1	0.392969	0.397917	0.093750	0.387500	7.55
3	0.254688	0.621528	0.037500	0.215278	5.89
2	0.127734	0.713889	0.253906	0.547222	4.50
1	0.666797	0.346528	0.030469	0.223611	10.00
1	0.809766	0.329861	0.067969	0.279167	10.00
4	0.916016	0.929861	0.166406	0.131944	3.58

<class_id> <x_center> <y_center> <width> <heig nce>

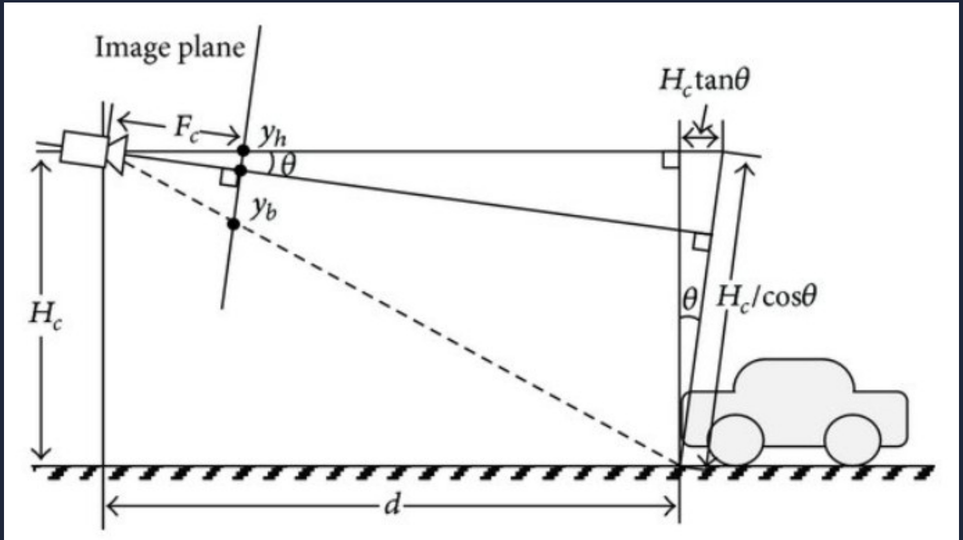
- All values except class_id are normalized to [0, 1]
- Distance is in meters, representing camera-to-obje enter
- Format supports both object detection and distance imation

Problems with distance information processing

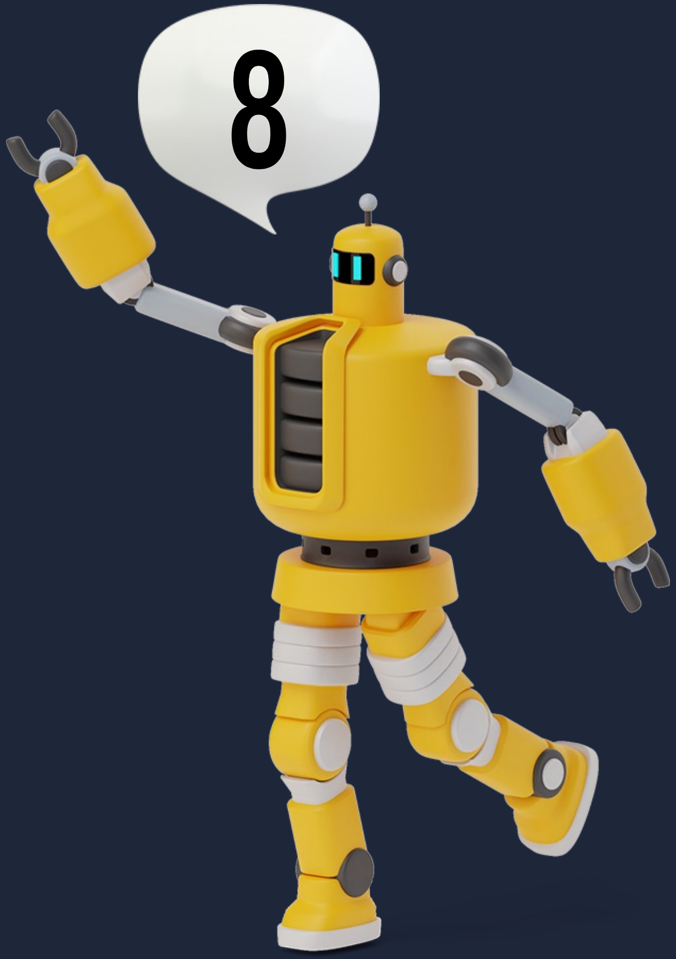


Direct depth estimation challenges showing inaccurate distance measurement (4.1 m) due to inclusion of multiple objects within the bounding box

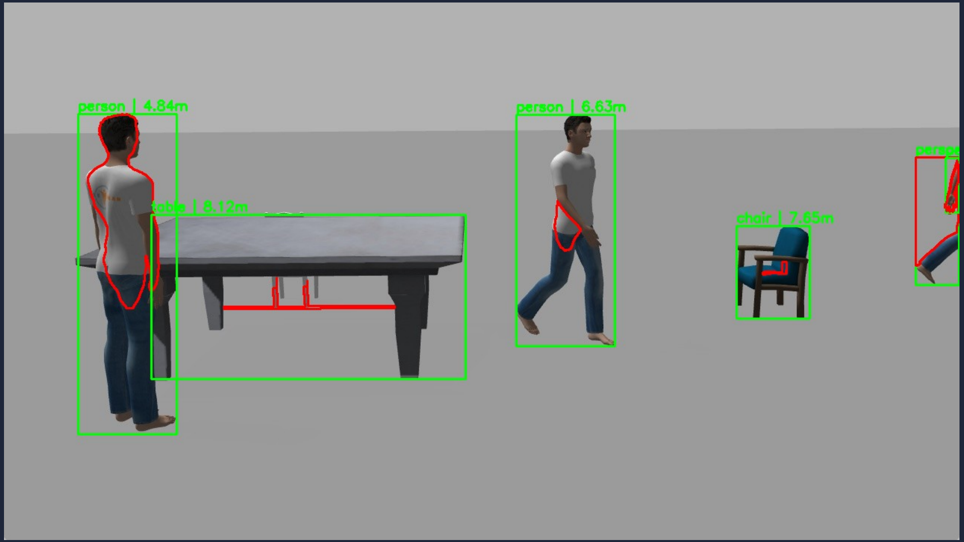
Camera Top Angle Processing



Schematic diagram of the monocular camera ranging algorithm showing the pinhole imaging principle

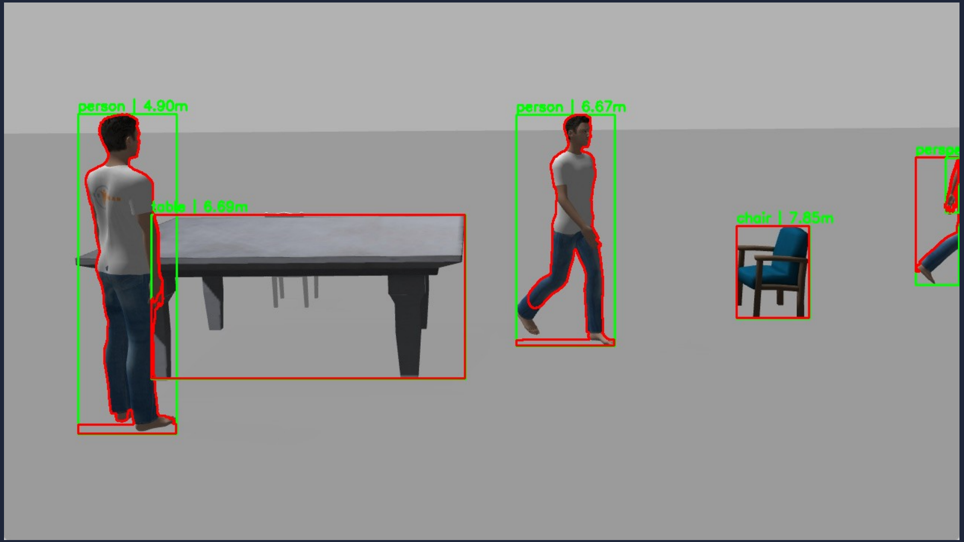


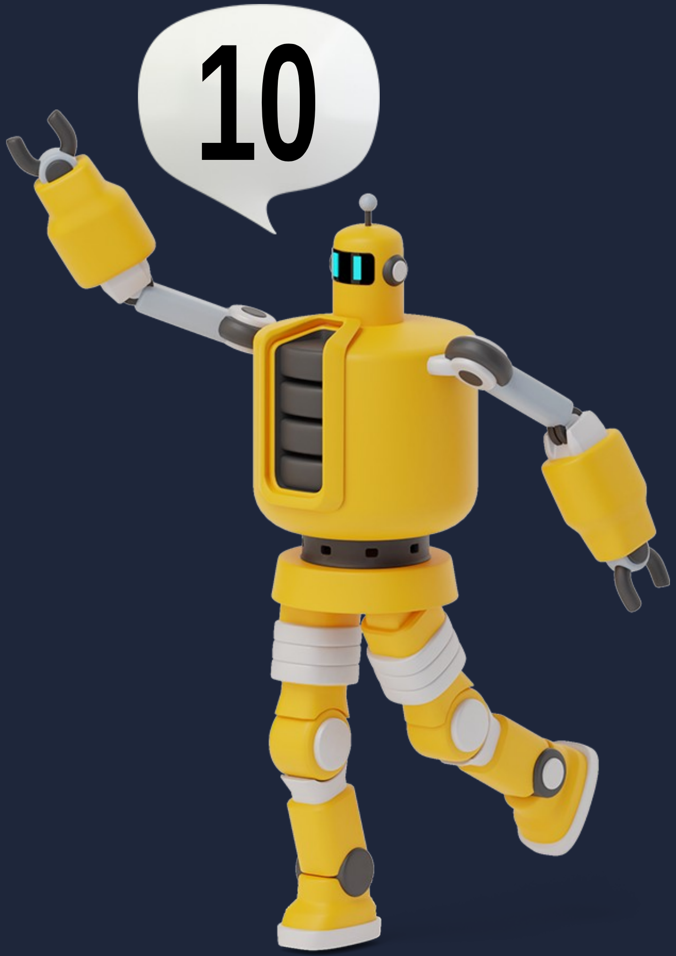
Cross-validation of distance information



Region growing segmentation results using uniform threshold, showing limitations in object boundary detection

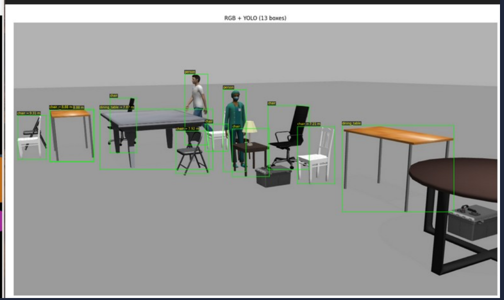
Cross-validation of distance information



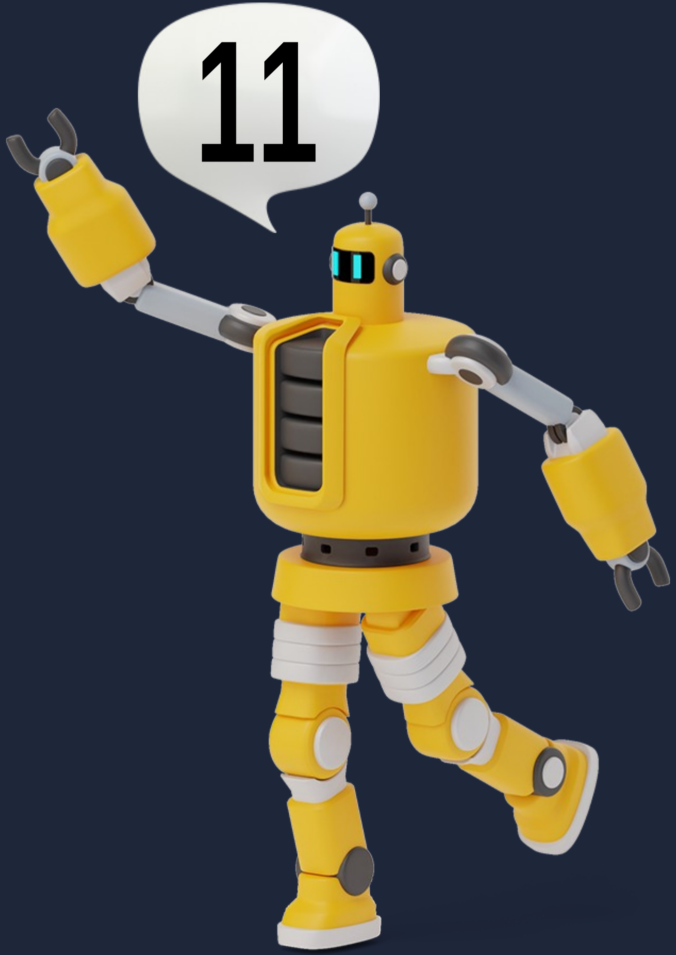


Improved region growing segmentation using category-specific thresholds, demonstrating better object boundary definition

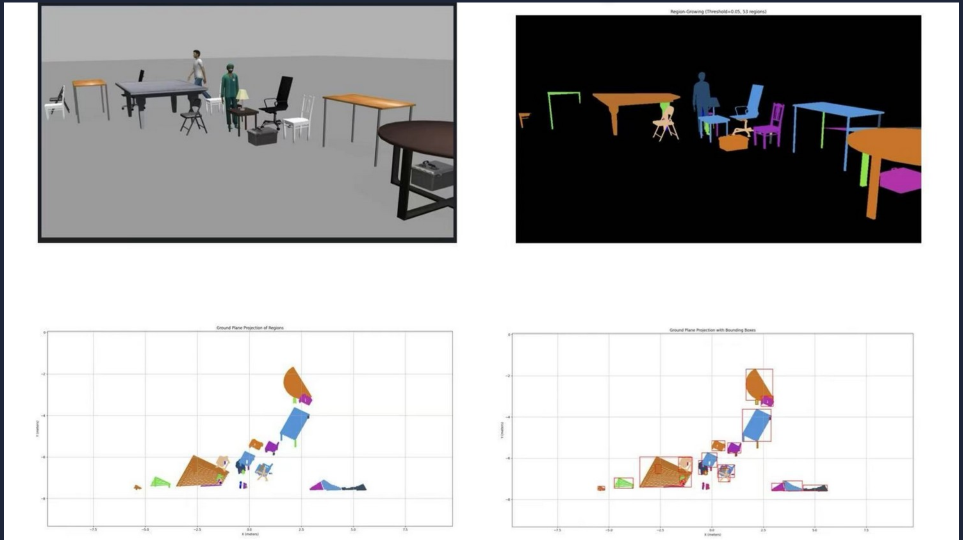
Distance information



Visualization of raw depth image data used for distance estimation

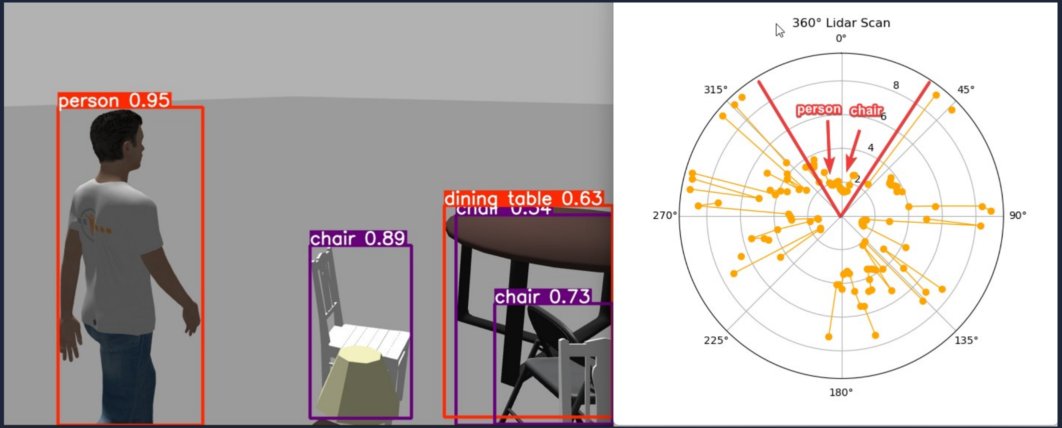


Depth camera mapping attempt



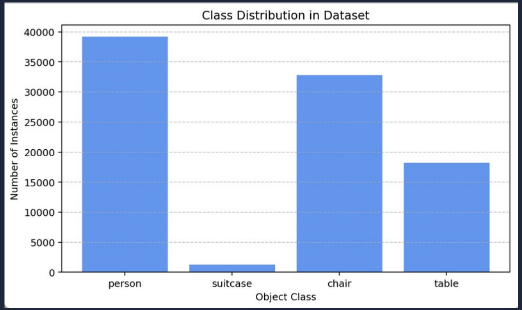
Processing pipeline showing: (a) Original RGB image, (b) Segmented depth image, (c) Horizontal projection, and (d) Bounding box detection

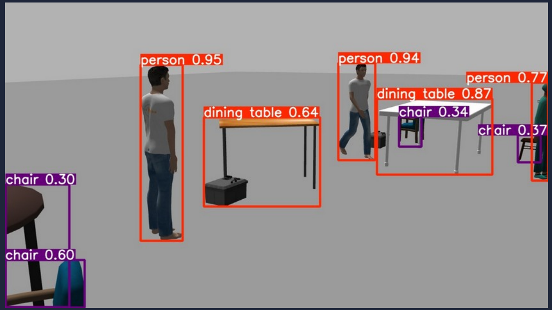
Cross-validation of distance information

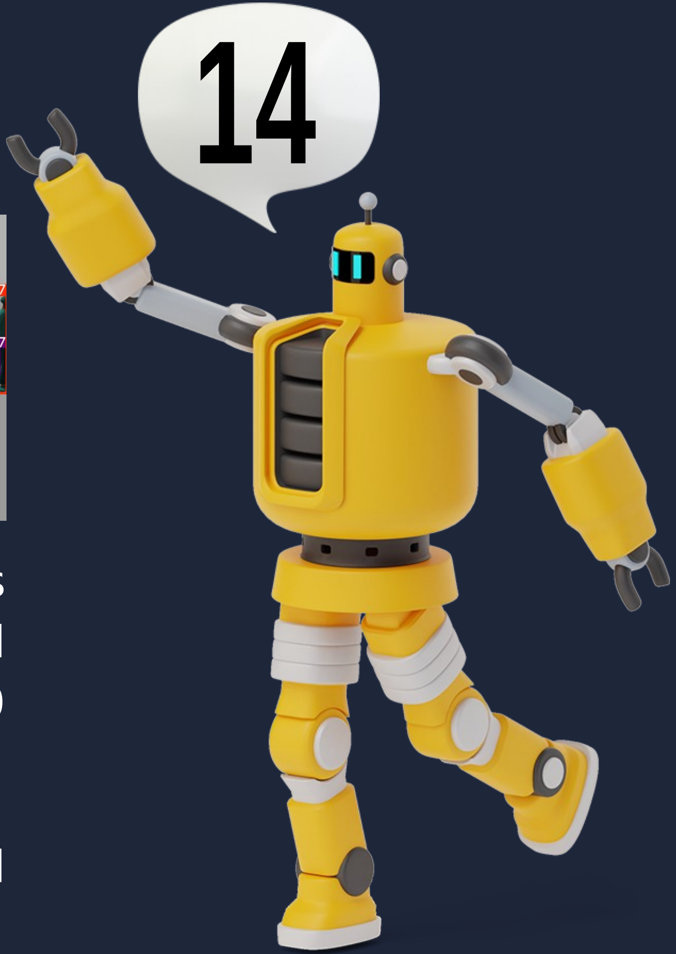


Evaluating LiDAR recognition for target depth distance variation

dataset preprocessing





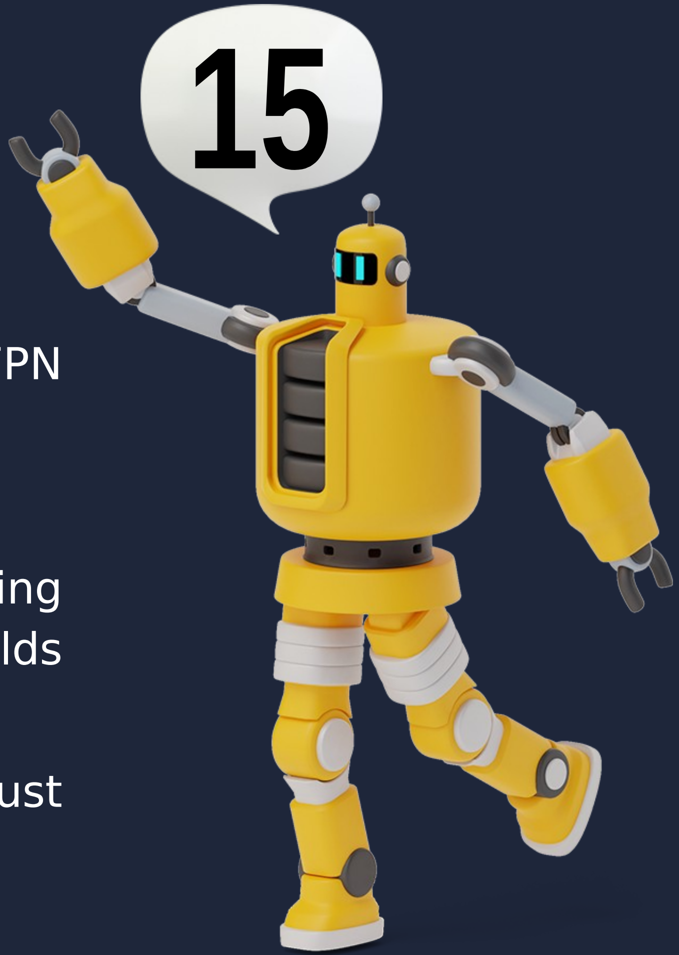


The training dataset comprises 10,077 images split in an 8:2 ratio between training and validation sets, with a batch size of 4 over 50 epochs

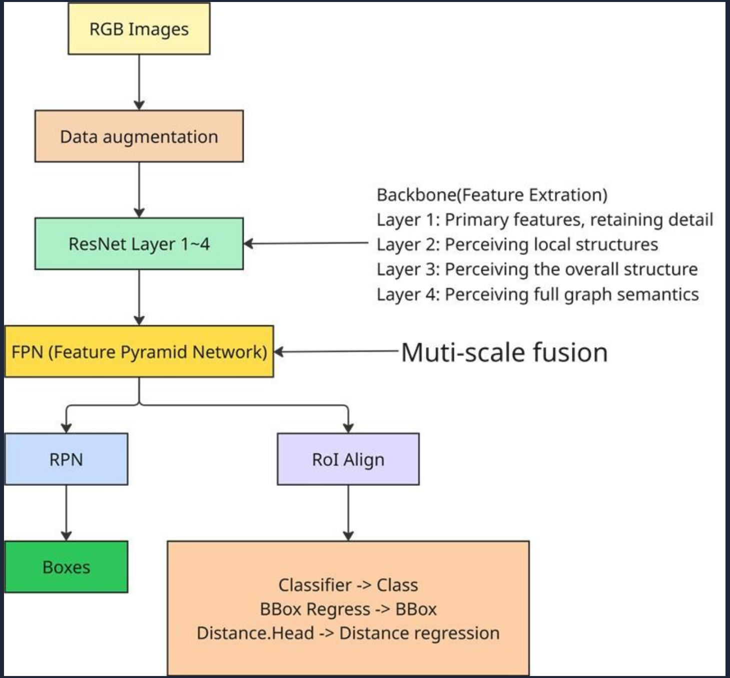
Preprocessed with Python scripts for format and class checks

Vision-Based Perception (CNN)

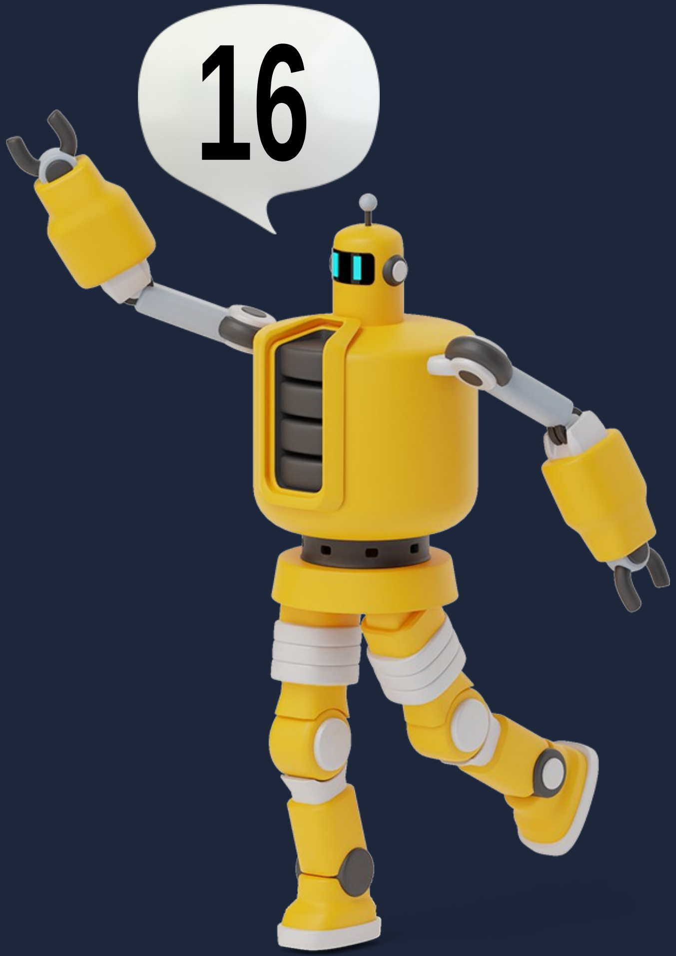
- Modified Faster R-CNN with ResNet101 + FPN backbone
- Joint object detection & distance estimation
- Depth refinement using region-growing segmentation with class-specific thresholds for accurate distance measurement
- RGB + LiDAR data integrated for robust environmental understanding



Faster R-CNN framework



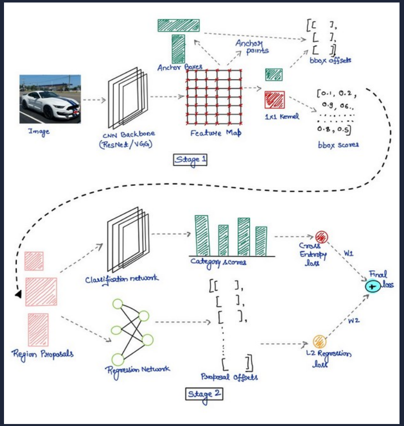
Faster R-CNN framework showing the network architecture and processing pipeline



feature extraction

Faster R-CNN architecture (input 3 × 720 × 1280)

Stages	Roles	Feature map changes (input 3×720×1280)
Conv1+bn+relu	Initial convolution + activation	[64, 360, 640]
Maxpool	Downsampling	[64, 180, 320]
Layer1	Residual module × 3	[256, 180, 320]
Layer2	Residual module × 4 + down-sampling	[512, 90, 160]
Layer3	Residual module × 23 + down-sampling	[1024, 45, 80]
Layer4	Residual module × 3 + down-sampling	[2048, 23, 40]
Avgpool	Pooling to 1x1	[2048, 1, 1]
FC	Classification output	[1000]



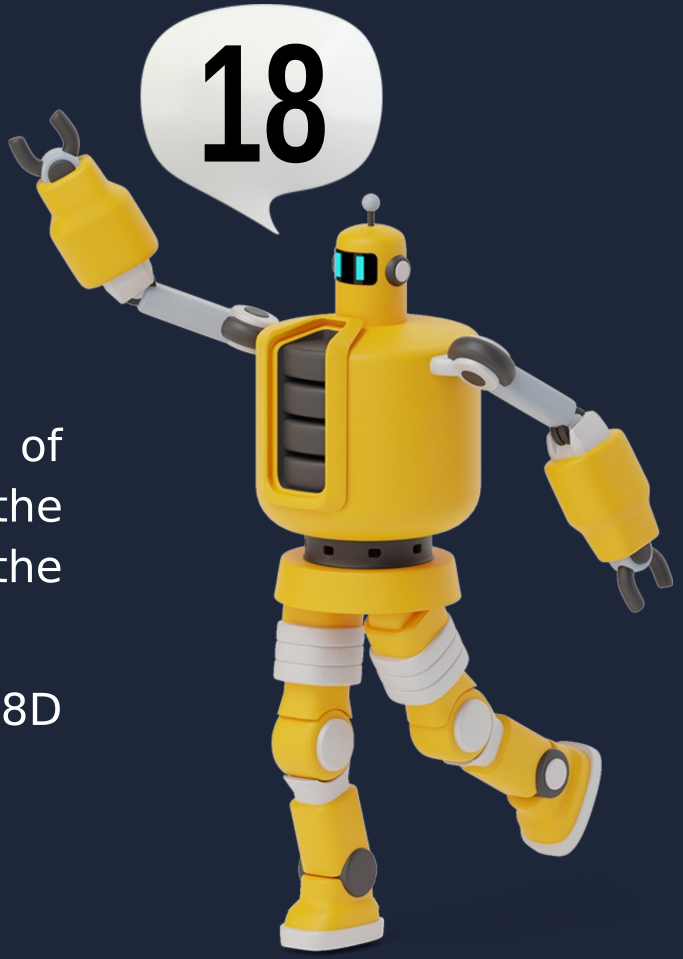
$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{cls}} + \mathcal{L}_{\text{bbox}} + \alpha \cdot \mathcal{L}_{\text{dist}}$$

$$\mathcal{L}_{\text{dist}} = \frac{1}{N} \sum_{i=1}^{N} \left| d_i^{\text{pred}} - d_i^{\text{gt}} \right|$$

Our model is based on Faster R-CNN with a ResNet-FPN backbone. The standard classification and bounding box heads are retained, using cross-entropy and Smooth L1 loss respectively. We additionally introduce a custom distance regression head that outputs the estimated object-camera

Physics-Informed Control (PINN)

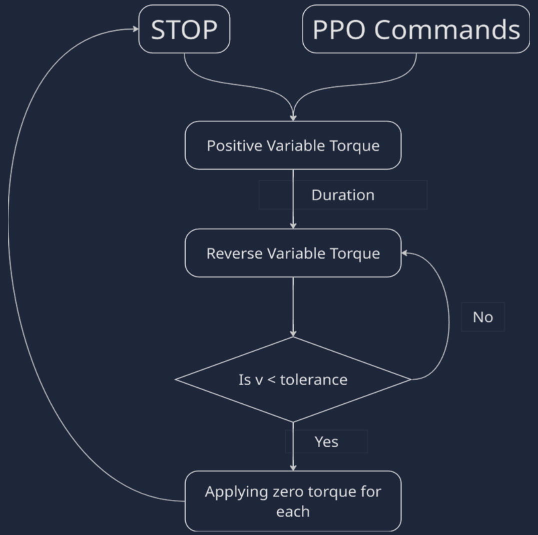
- Neural network embeds wheel dynamics: $J\ddot{\theta} + b\dot{\theta} = \tau_m - \tau_l$
- where J represents the wheel's moment of inertia, b is the damping coefficient, θ is the wheel angle, τ is the motor torque and the load torque
- Inputs: Wheel states, torques, timestep (8D input vector)



Physics-Informed Control (PINN)

- [64, 32] hidden layers with ReLU activations
- Multi-threaded ROS2 services for real-time torque updates at 30 Hz
- Replaces speed-control plugins with reverse torque strategy for direct control of wheel acceleration

PINN framework



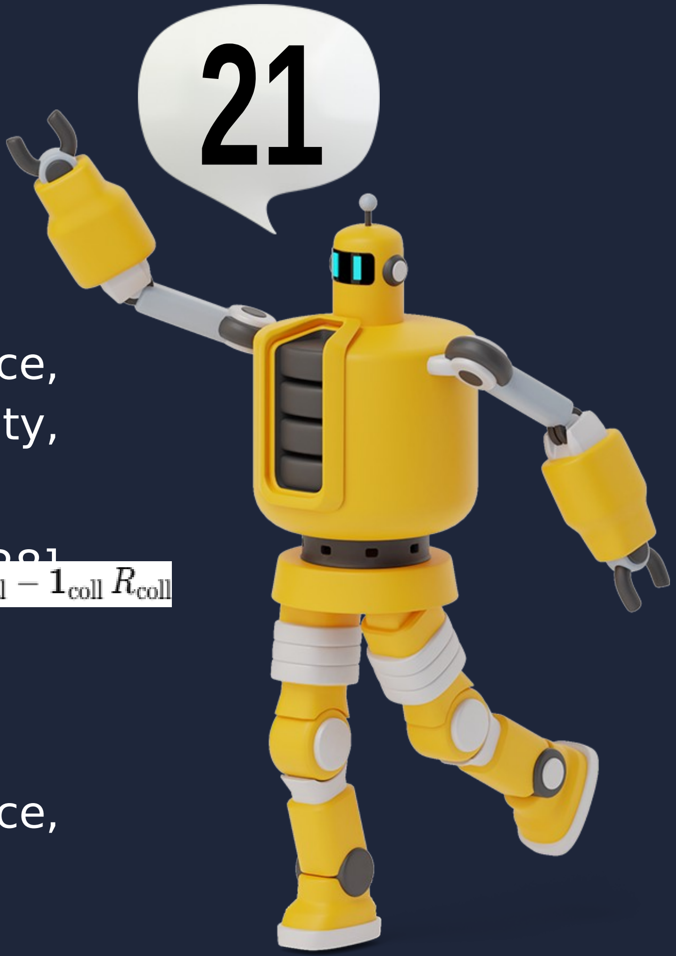
PINN torque control flowchart showing the multi-threaded control architecture

Reinforcement Learning (PPO)

- Agent Input (6D state): Target distance, obstacle distance, linear & angular velocity, visibility flag, normalized timestep
- Policy & Critic Networks: Dual [256, 128] layers, ReLU activations

$$R_t = k_p\,(d_{t-1} - d_t) + k_v\,v_t - k_\omega\,\omega_t^2 + k_s\,\max\big(0,\ s_{\min,t} - s_{\text{th}}\big) + \mathbf{1}_{\text{goal}}\,R_{\text{goal}} - \mathbf{1}_{\text{coll}}\,R_{\text{coll}}$$

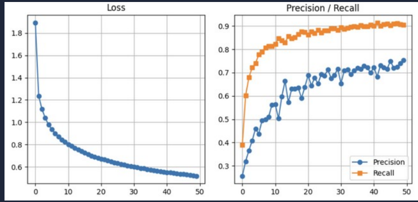
- Balances goal-seeking, obstacle avoidance, smooth motion, and energy usage



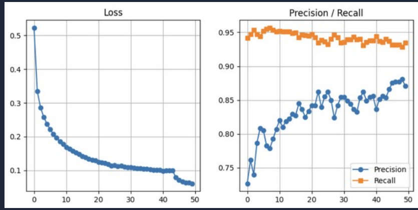
Results & discussion

- exhibited effective navigation in dynamic environments, with the PPO algorithm successfully learning to combine information from the CNN and PINN components to generate appropriate navigation commands
- Performance evaluation showed consistently high success rates in reaching target locations while maintaining smooth trajectory generation
- The system demonstrated reliable obstacle avoidance capabilities and effective speed

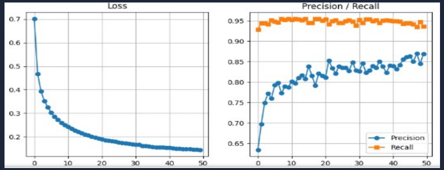
Optimizer Comparison



SGD



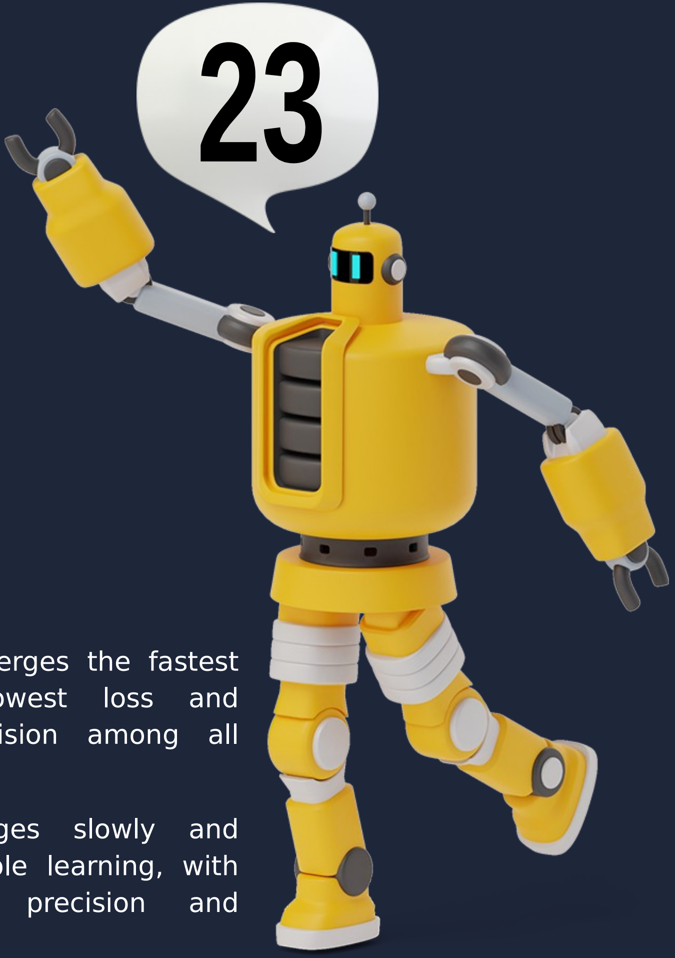
Adam



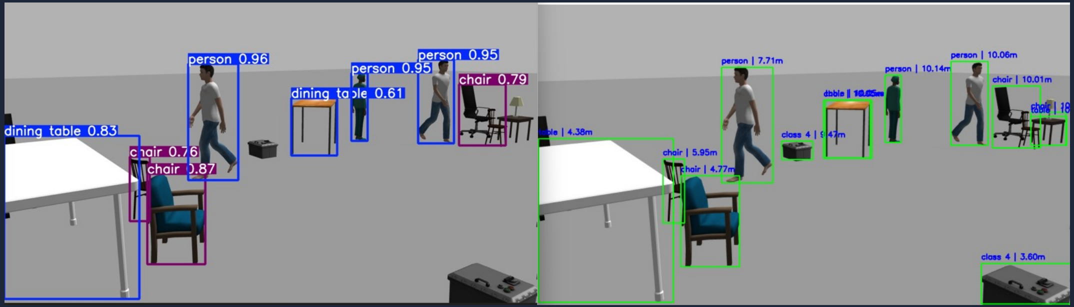
SGD with Momentum

	AdamW	SGD	SGD + Momentum
Avg Loss	0.0612	0.5181	0.1442
Precision	0.8711	0.7536	0.8689
Recall	0.9348	0.9054	0.9369
Remarks	Fast convergence, strong on regression	Slow, weak performance	Balanced performance, strong generalization

- AdamW converges the fastest with the lowest loss and highest precision among all optimizers.
- SGD converges slowly and shows unstable learning, with the lowest precision and highest loss.
- SGD with Momentum achieves

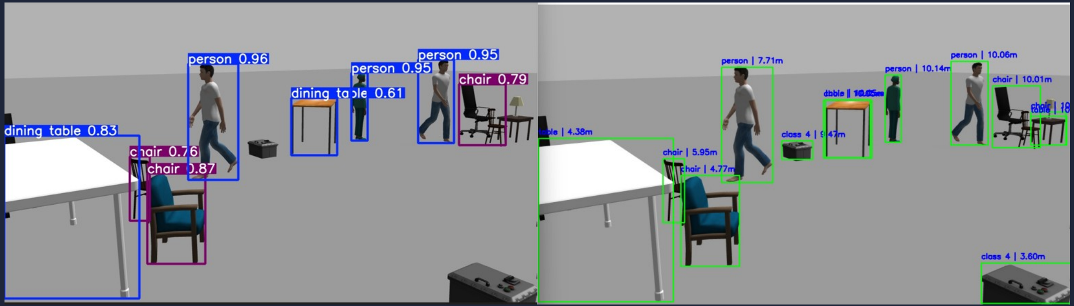


Yolov5l6 vs Faster RCNN (YOLOV5l6)



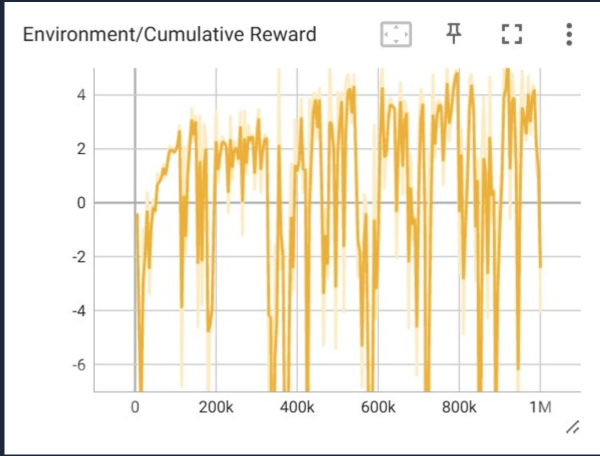
- One-stage model, very fast
- Bounding boxes less accurate, sometimes overlapping
- No distance estimation
- Good for real-time tasks, but less reliable in complex scenes

Yolov5l6 vs Faster RCNN (faster rcnn)



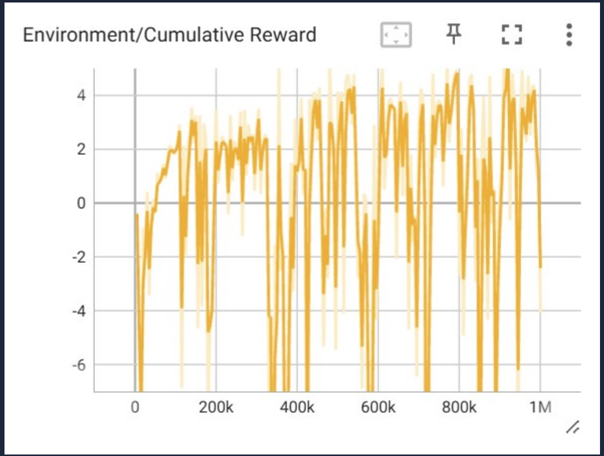
- Two-stage model, more accurate
- Boxes are tighter and more precise
- Supports real-world distance output
- Better at detecting occluded or cluttered objects

Ppo solution



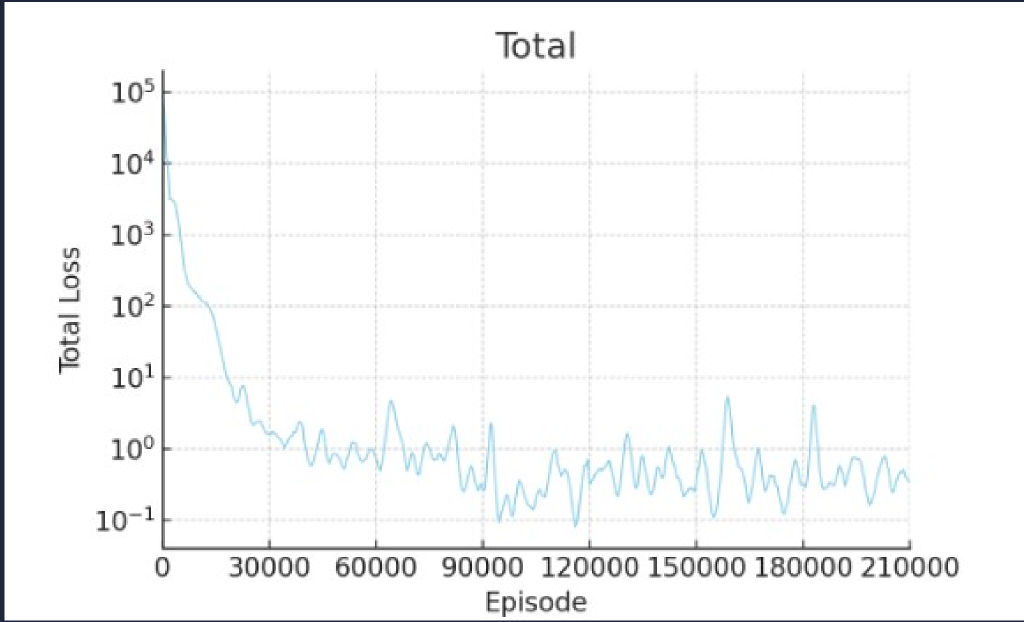
- Policy collapse due to unstable updates
- Delay or noise in visual perception (e.g., Model/Camera latency)
- Sparse or overly complex reward design

Ppo solution



- Improve reward shaping and feedback clarity
- Integrate more stable perception models
- Fine-tune PPO hyperparameters (learning rate, entropy, batch size)

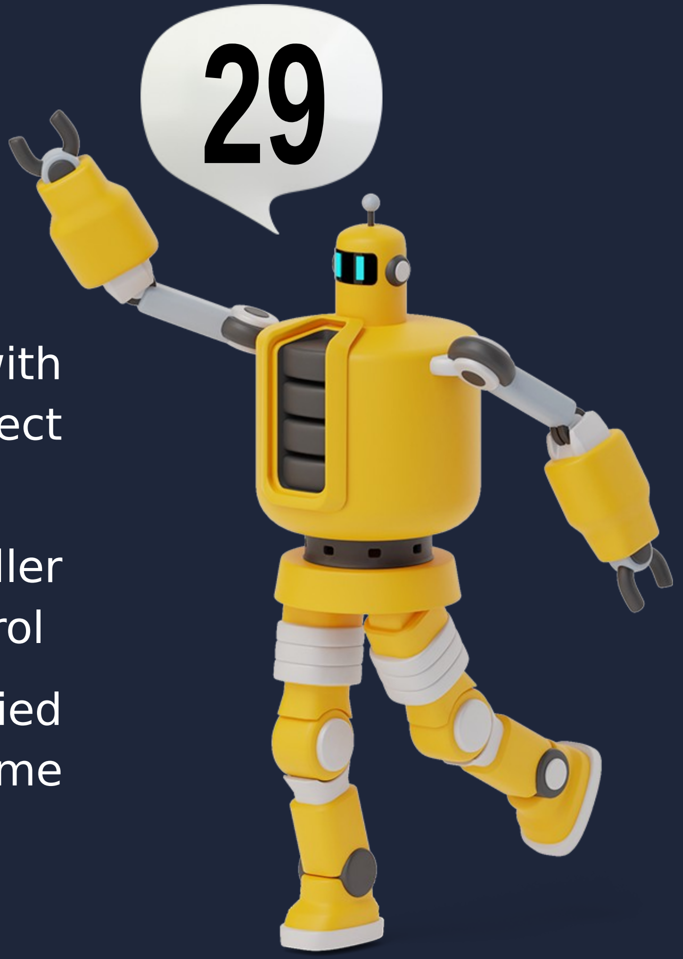
PINN result


Total
Total Loss
10^5
10^4
10^3
10^2
10^1
10^0
10^{-1}
0
30000
60000
90000
120000
150000
180000
210000
Episode


28

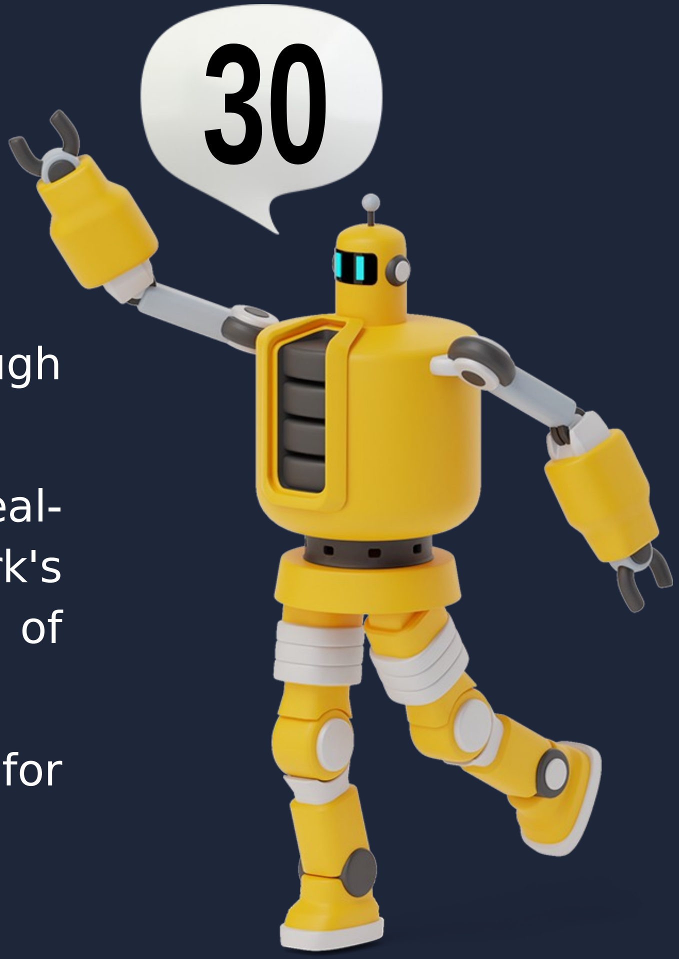
Conclusion

- a modified Faster R-CNN architecture with ResNet101 backbone for effective object detection and distance estimation
- a PINN-based wheel dynamics controller operating at 30 Hz with precise torque control
- a PPO implementation with verified hyperparameters managing real-time navigation decisions



Future work

- focus on bridging the sim-to-real gap through improved physics modeling
- optimizing computational efficiency for real-time operation, expanding the framework's capabilities to support a broader range of environments
- incorporating additional sensor modalities for enhanced perception



questions?