Course Number: 6.4200
Date: 3/16/2025
Lab Briefing: #4

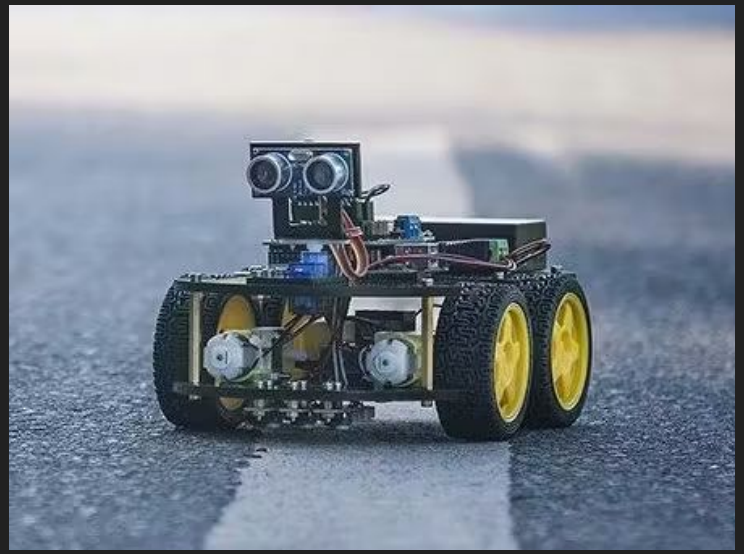
Autonomous Race Car: Visual Servoing

Team 7: Diego Contreras, Kevin Huang,
Nathaniel Morgan, Weiming Zhou, Soe Wai Yan

Overview: Visual Servoing Enables Real-Time Cone Parking and Line Following

Goal: Park in front of a cone. Follow a line.

Modules:
1 Detect orange cone by color
2 Object detection algorithms
3 Pixels to real-world (x, y)
4 Steer and stop at target distance
Synthesis: (line following)



Color Segmentation Can Effectively Detect Orange Cones

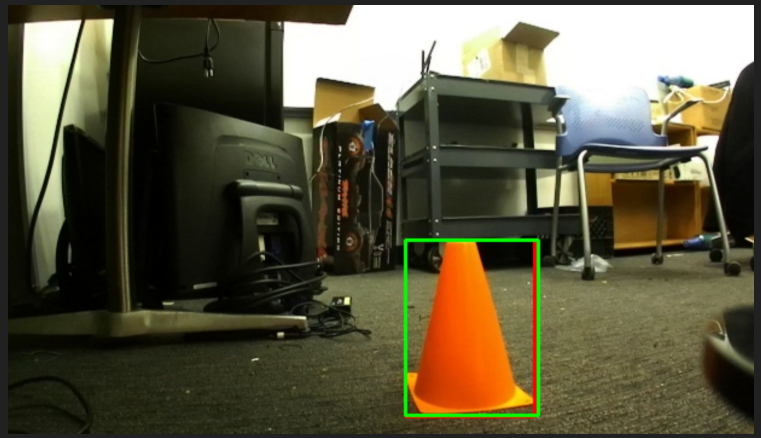
Original

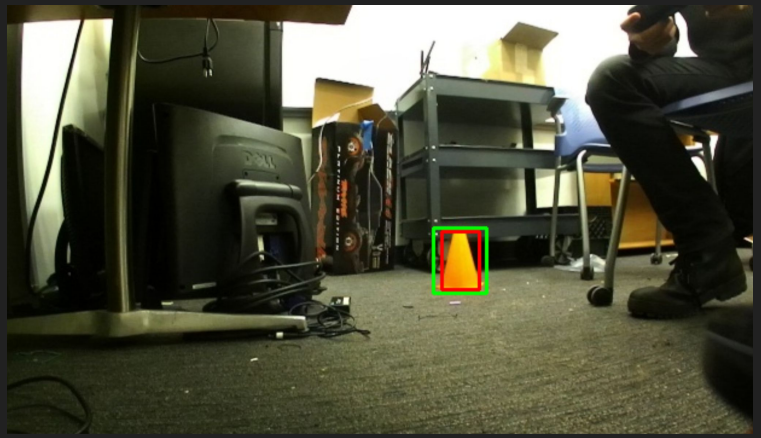
Gaussian Blur

Mask

Bounding Box

Our Color Segmentation Achieves median 0.79 with IQR = 0.18 IOU on the Cone Dataset

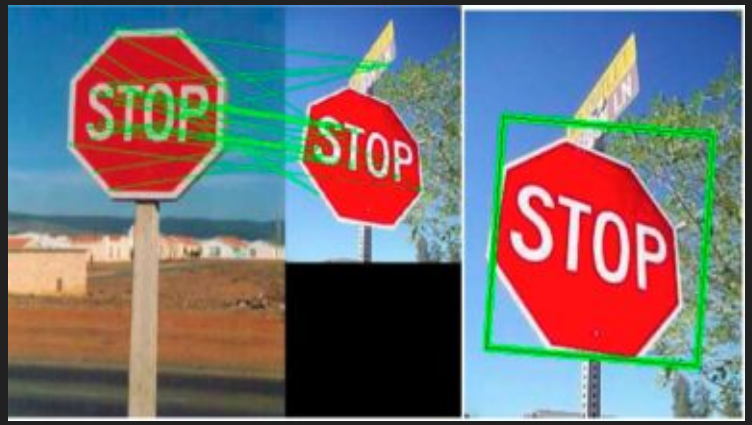




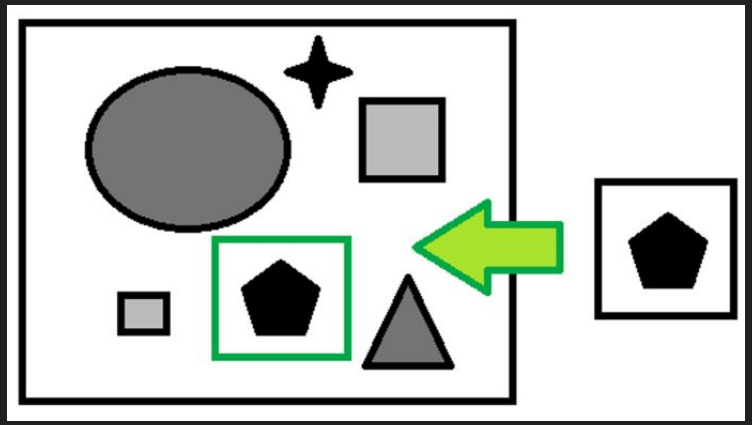
Ground truth bounding box in green, predicted bounding box in red. Test 1 (left) has an IOU score of 0.97; test 7 (right) has an IOU score of 0.63.

Object Detection Algorithms

Sift



Template Matching

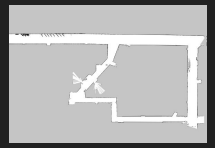


Sift Detection Results

```
~/visual_servoing/visual_servoing/visual_servoing/computer_vision~> python3 cv_test.py citgo sift
[SIFT] not enough matches; matches: 2
('./test_images_citgo/citgo1.jpeg', 0.7735889702714347)
('./test_images_citgo/citgo2.jpeg', 0.6287878787878788)
('./test_images_citgo/citgo3.jpeg', 0.7710727969348659)
('./test_images_citgo/citgo4.jpeg', 0.0)
('./test_images_citgo/citgo5.jpeg', 0.0)
('./test_images_citgo/citgo6.jpeg', 0.640294388224471)
('./test_images_citgo/citgo7.jpeg', 0.6195652173913043)
('./test_images_citgo/citgo8.jpeg', 0.765843179377014)
('./test_images_citgo/citgo9.jpeg', 0.7003787878787879)
('./test_images_citgo/citgo10.jpeg', 0.7740384615384616)
('./test_images_citgo/citgo11.jpeg', 0.9089433826181031)
('./test_images_citgo/citgo12.jpeg', 0.0)
('./test_images_citgo/citgo13.jpeg', 0.8351648351648352)
('./test_images_citgo/citgo14.jpeg', 0.8659638554216867)
```

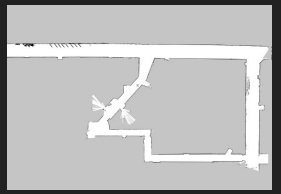
Citgo Data set – Works well

Stata Map – Works Poorly



```
~/visual_servoing/visual_servoing/visual_servoing/computer_vision~> python3 cv_test.py map sift
[SIFT] not enough matches; matches: 0
[SIFT] not enough matches; matches: 0
[SIFT] not enough matches; matches: 0
[SIFT] not enough matches; matches: 1
[SIFT] not enough matches; matches: 0
[SIFT] not enough matches; matches: 5
[SIFT] not enough matches; matches: 1
[SIFT] not enough matches; matches: 0
[SIFT] not enough matches; matches: 1
('./test_images_localization/map_scrap1.png', 0.0)
('./test_images_localization/map_scrap2.png', 0.0)
('./test_images_localization/map_scrap3.png', 0.0)
('./test_images_localization/map_scrap4.png', 0.0)
('./test_images_localization/map_scrap5.png', 0.0)
('./test_images_localization/map_scrap6.png', 0.0)
('./test_images_localization/map_scrap7.png', 0.0)
('./test_images_localization/map_scrap8.png', 0.0)
('./test_images_localization/map_scrap9.png', 0.0)
```

Template Matching Results

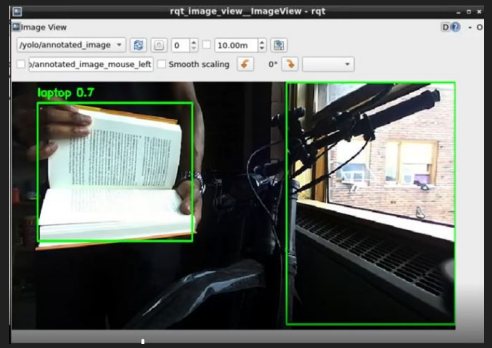


```
~/visual_servoing/visual_servoing/visual_servoing/computer_vision~> python3 cv_test.py map template
('./test_images_localization/map_scrap1.png', 0.6035714285714285)
('./test_images_localization/map_scrap2.png', 0.481651376146789)
('./test_images_localization/map_scrap3.png', 0.905727923627685)
('./test_images_localization/map_scrap4.png', 0.8822371012209531)
('./test_images_localization/map_scrap5.png', 0.7962413452027696)
('./test_images_localization/map_scrap6.png', 0.868421052631579)
('./test_images_localization/map_scrap7.png', 0.5555555555555556)
('./test_images_localization/map_scrap8.png', 0.0)
('./test_images_localization/map_scrap9.png', 0.9003831417624522)
```

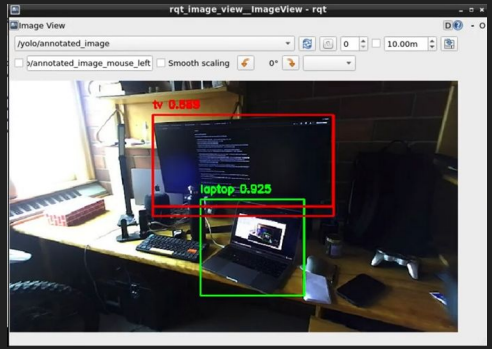
Stata Map – Works Well

Method	Best Use Case
SIFT	Landmark Localization
Template Matching	Map Localization

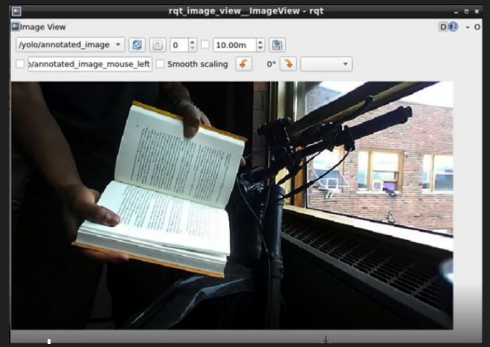
YOLO Detects Objects on the Live ZED Feed with Tunable Confidence and IOU Thresholds



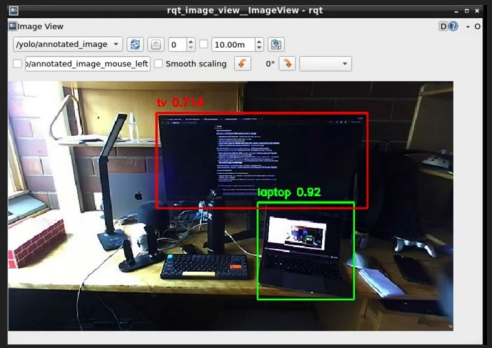
C: .2



IOU: .9

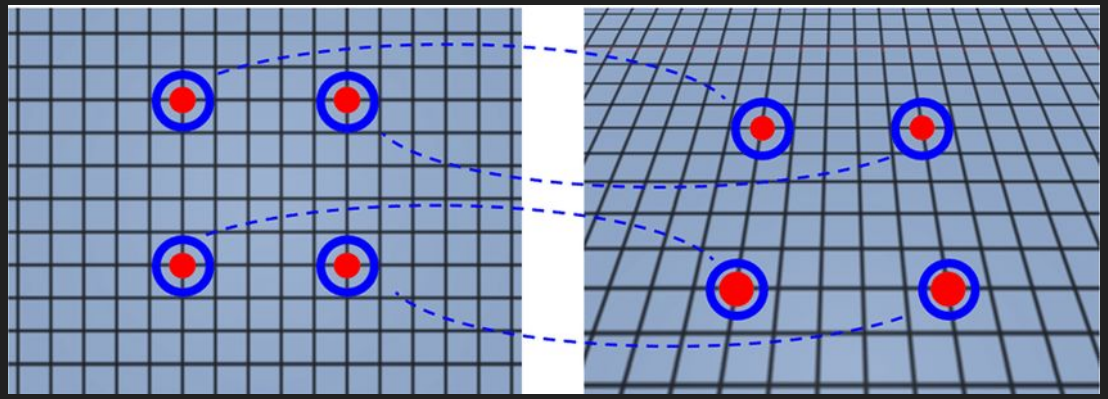


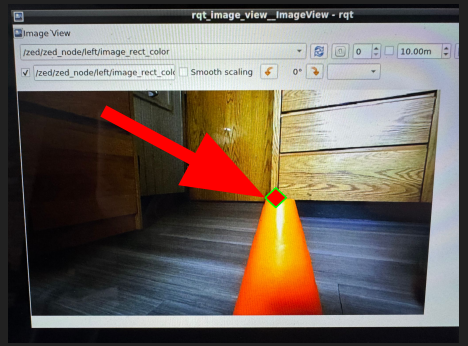
C: .9



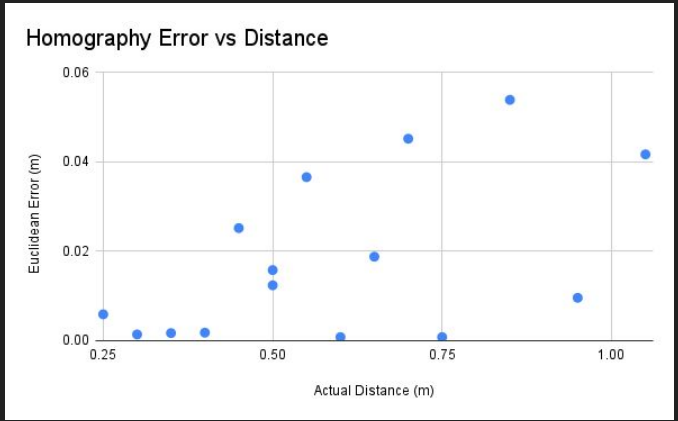
IOU: .2

Homography Transforms Pixel Coordinates to Robot Frame with 1.5 cm Mean Error

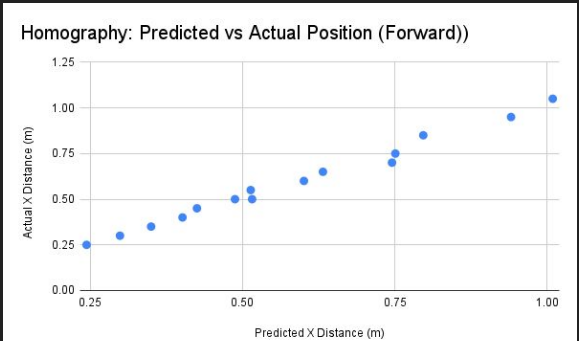


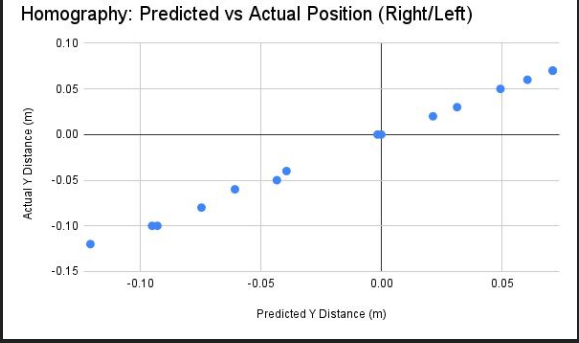


Homography Transforms Pixel Coordinates to Robot Frame with 1.5 cm Mean Error



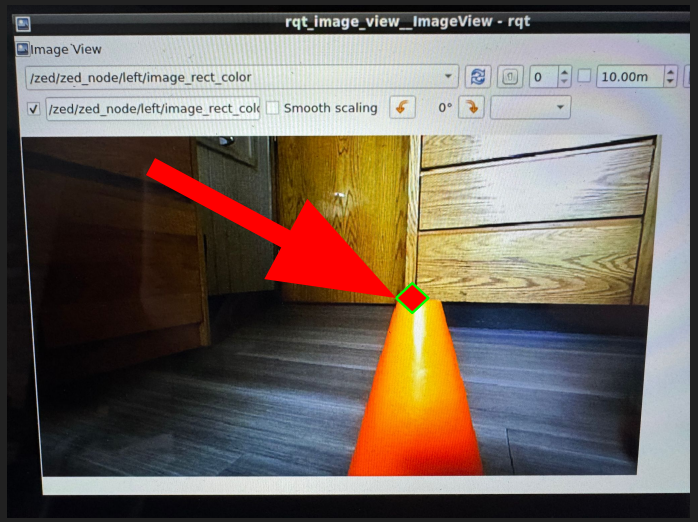
Mean: 1.5cm
Std. Dev: 1.7cm

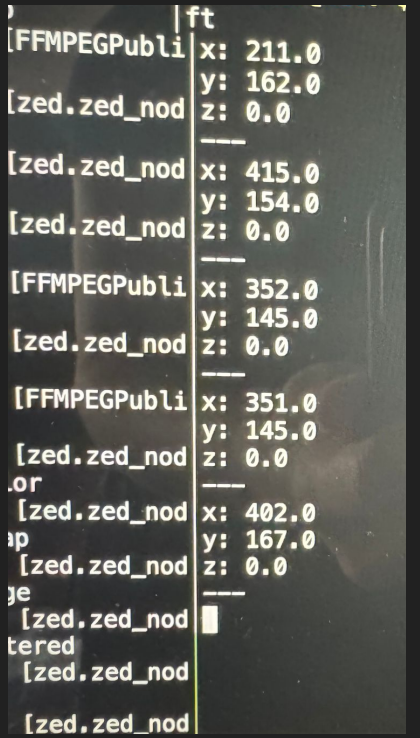




Mainly Forward Error

Manual Calibration with rqt_image_view



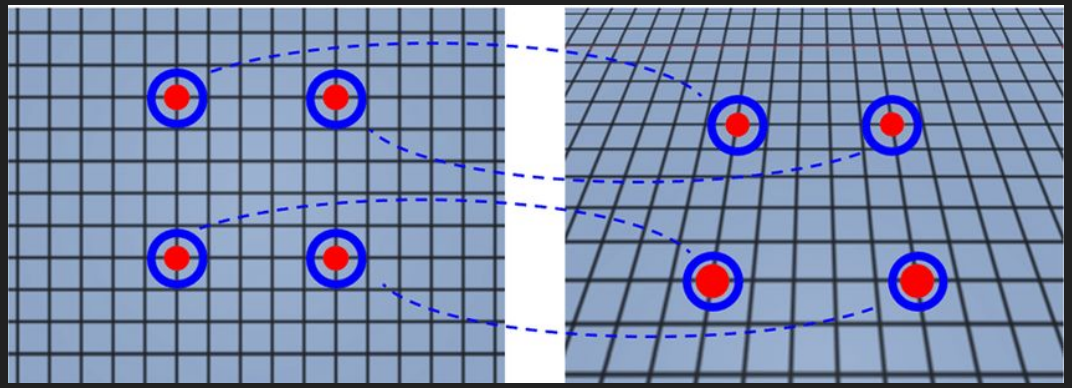


Homography

Point	Pixel (u, v)	Ground (x cm, y cm)
1	(211, 162)	(30.48, 7.62)
2	(415, 154)	(46.99, -12.70)
3	(351, 145)	(109.22, -13.97)
4	(402, 167)	(22.86, -6.35)

cv2.findHomography()

$$s \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & 1 \end{bmatrix} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}$$



X, Y

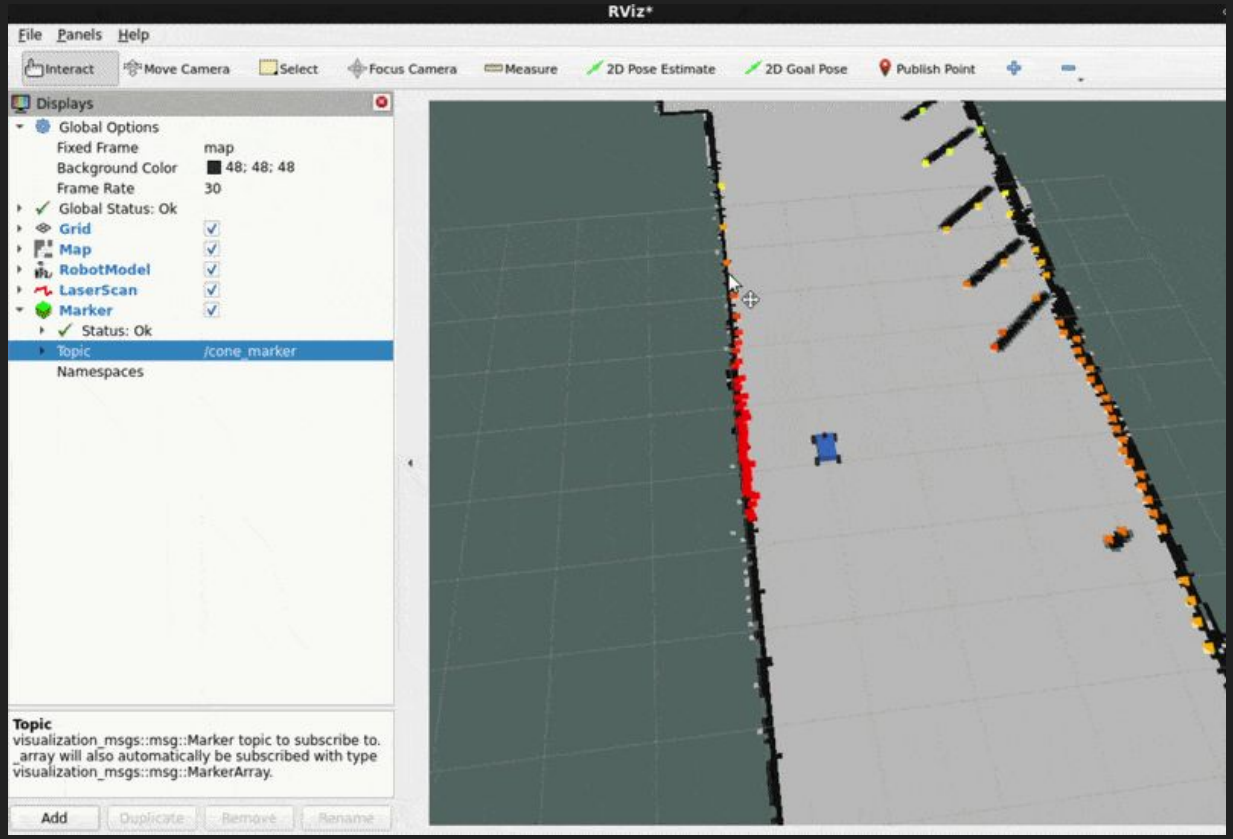
U, V

Our Parking Controller Converges to the Target Distance Across Multiple Trials

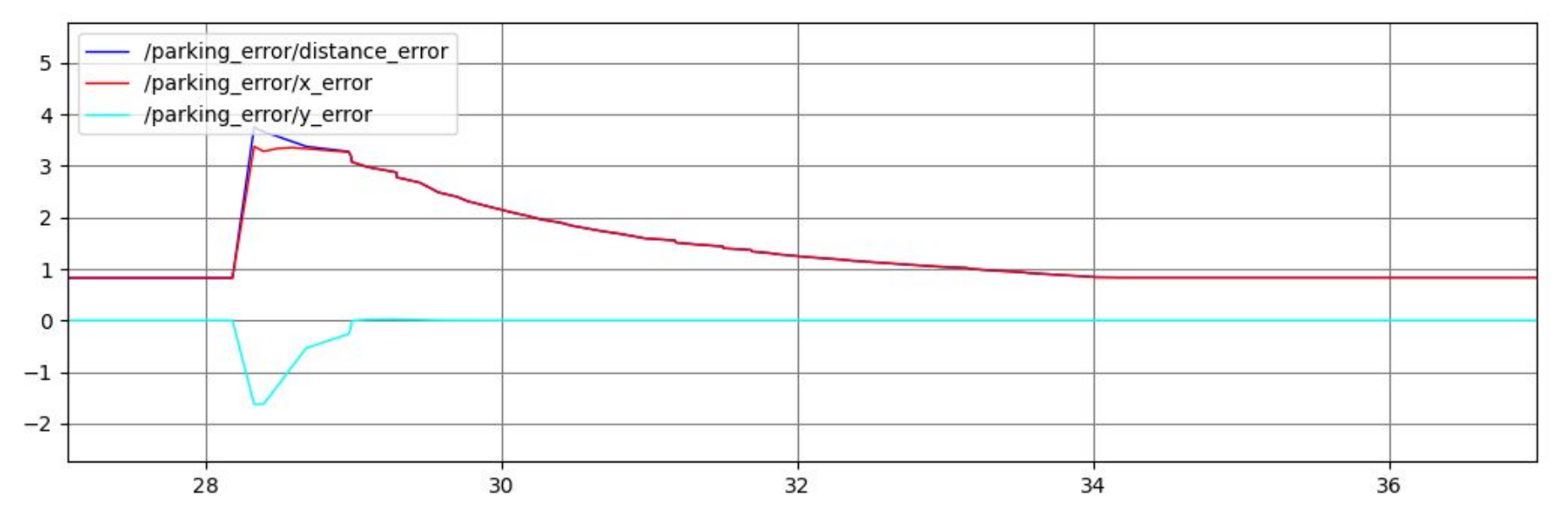
Controller Design:
- If cone is far: drive closer
- If cone is too close: reverse further
- If cone is off-center: steer to align
- Desired parking distance: .75 meters
- Parameters: Kp = 1.0 | Kd = 0.1

Performance (N = 9 trials):
- Final x_error: .75 m | Final distance_error: 0 m

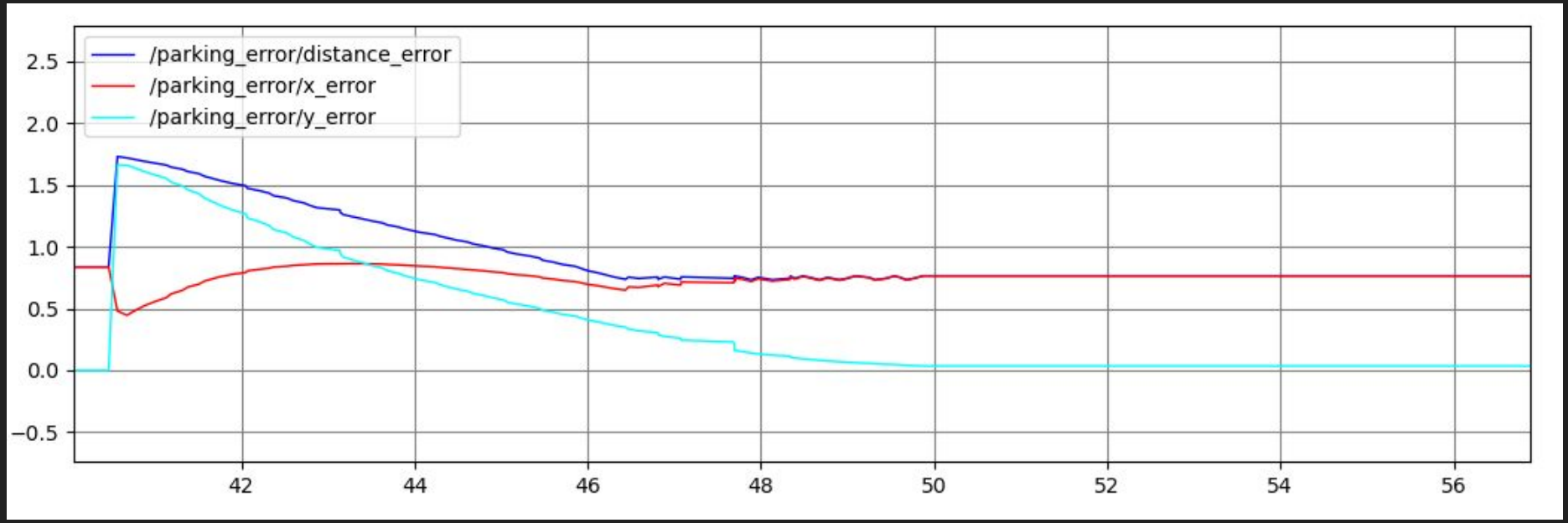
Simulation Tests



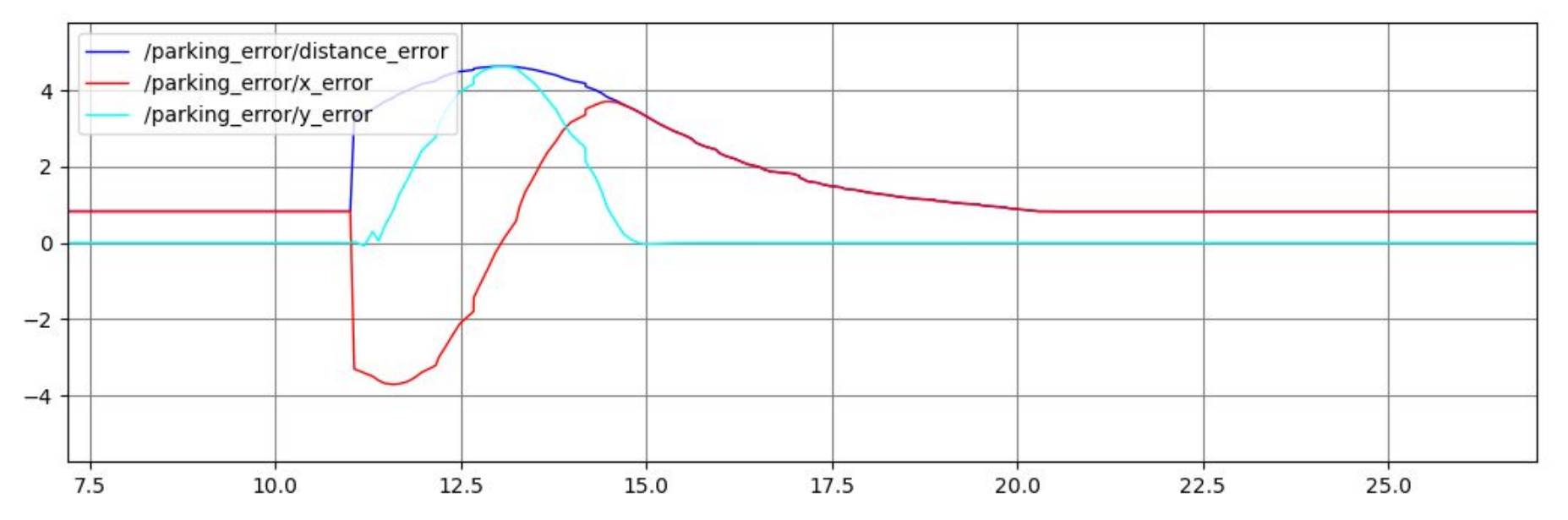
Sim Test 1: Cone in front



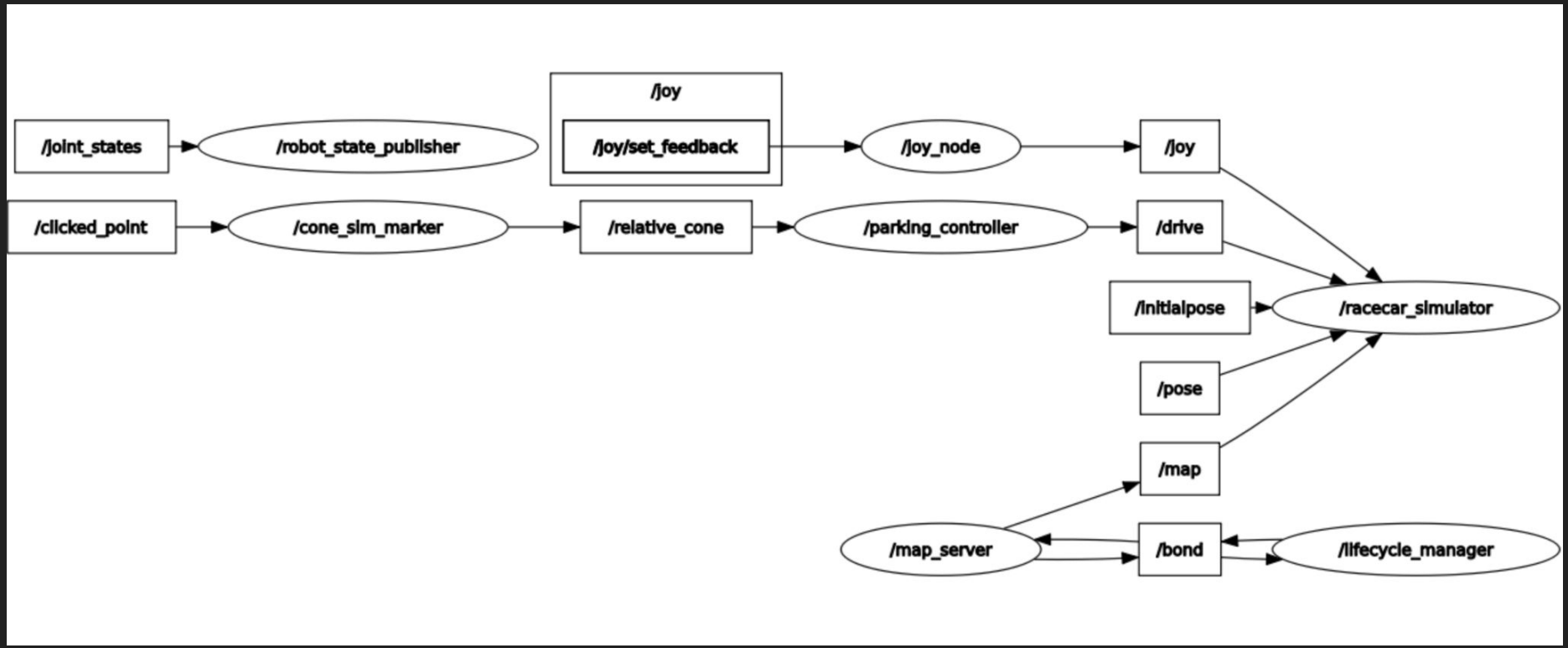
Sim Test 2: Cone to Side



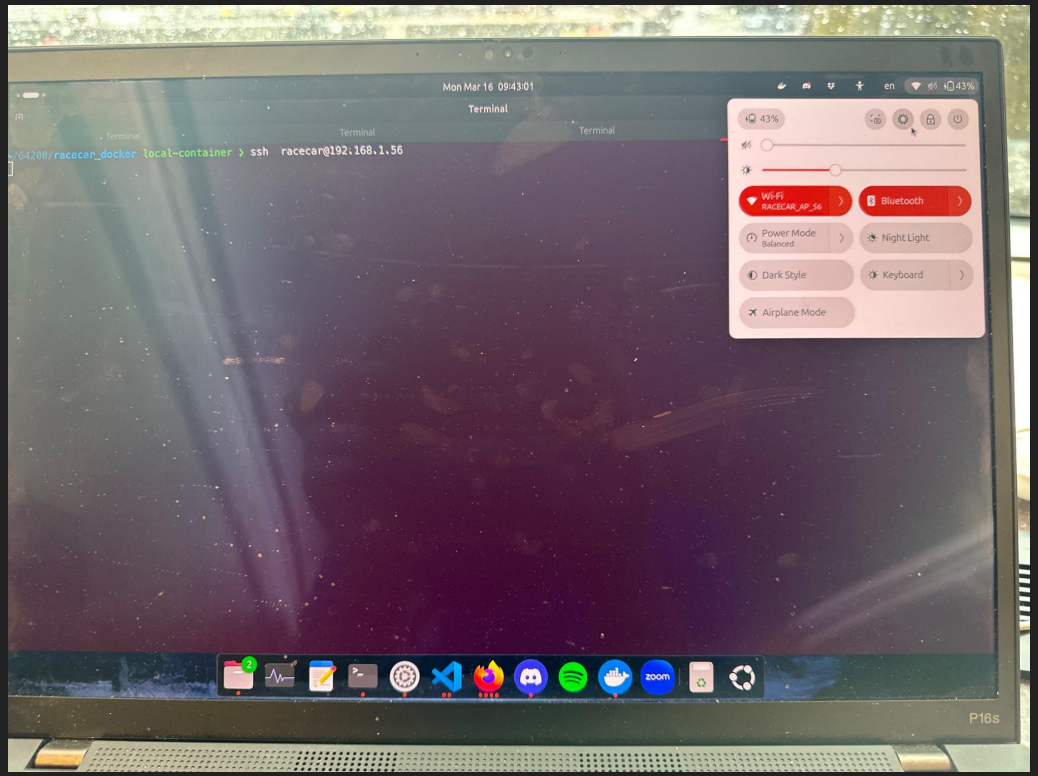
Sim Test 3: Cone Behind

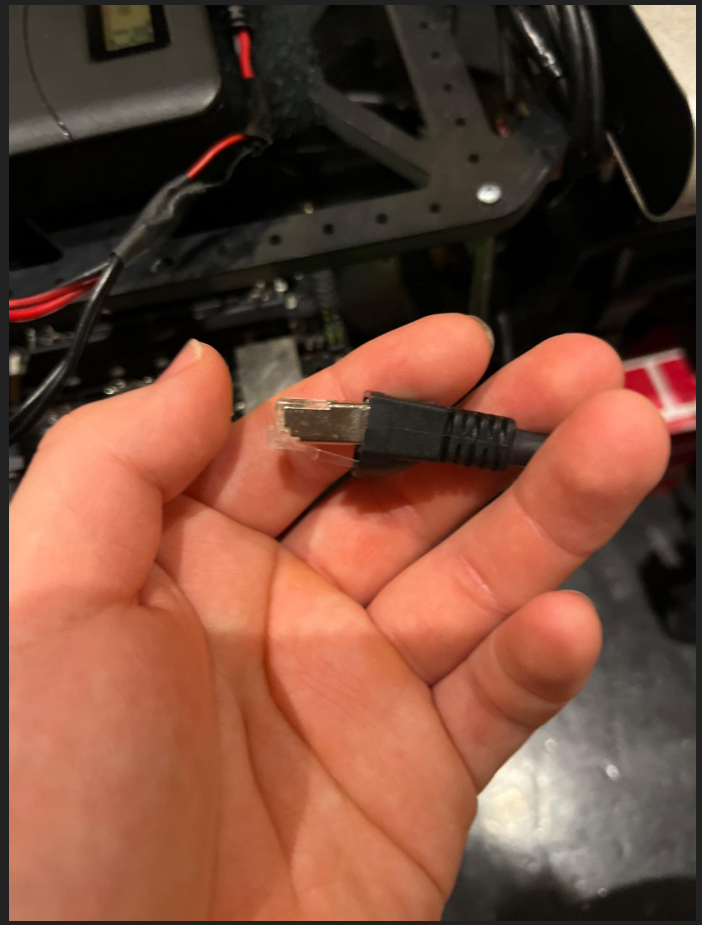


ROS Node Graph for the Parking Pipeline



Robot Issues





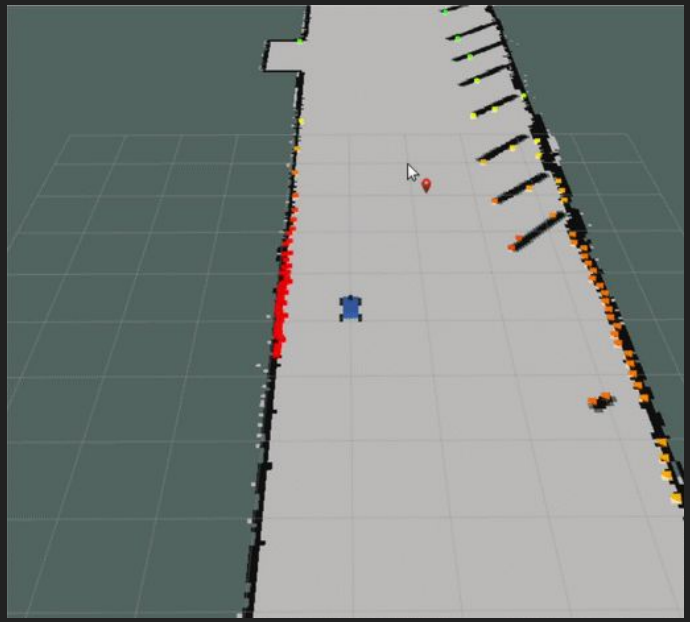
Conclusion

Visual Servoing Enables Reliable Cone Parking in Simulation

Results Summary:
- Color segmentation IOU on cone dataset: [value]
- Homography error: [value] cm
- Parking final distance error: N/A
- Max line following speed: N/A

Lab Goals - Status:
[✔️] Object detection algorithms implemented and evaluated
[✔️] Homography computed and validated with error metric
[❌] Parking controller deployed on real robot
[❌] Line following demonstrated on track

Future Work: Deploy on Robot



Citations

[1] OpenCV Documentation. cv2.findHomography. https://docs.opencv.org/4.x/d9/d0c/group__calib3d.html
[2] Ultralytics. YOLOv8 Documentation. https://docs.ultralytics.com
[3] MIT RSS Lab 4: Visual Servoing. https://github.com/mit-rss/visual_servoing
[4] A. Shwaiheen, "Line Follower Robot - Very Fast Using Port Manipulation," Hackster.io, Jan. 3, 2020. [Online]. Available: https://www.hackster.io/Laheeb/line-follower-robot-very-fast-using-port-manipulation-c87639