MXET 300
Mechatronics I

Multidisciplinary
Engineering Technology
COLLEGE OF ENGINEERING

LABORATORY # 8

Computer Vision

Submission Date: 04/07/2025

Name: Kyle Rex and Rex Worley
Section: 501
UIN: 932008894 and 432006442

Introduction

This lab introduces critical principles of computer vision and how each is utilized in robotics to detect and track objects solely based on color. Live image data was obtained and integrated into the SCUTTLE system through a USB camera connected to the Raspberry Pi onboard; this data was processed through an HSV color filter to isolate a predetermined range of colors identifying the target object. The central positioning of the object within the camera's vision parameters was done by locating the filtered region's centroid and estimating the object's position relative to the camera frame. Specifics for the robot's heading and velocity were internally calculated from the camera vision data which allowed the SCUTTLE to follow the object in real time. Overall, the usefulness of image processing and color-based image tracking was demonstrated through the execution of this laboratory experiment.

Procedures and Lab Results

The lab begins with the assembly of the SCUTTLE robot, ensuring that all components, including the Raspberry Pi, battery pack, and motor driver bracket, are securely placed on the DIN rail, before wiring, following the same procedure for mounting these parts that is described in the Lab 3 manual. Placing the parts first, before wiring, prevents potential damage to connectors. Once all the parts are placed on the DIN rail the connections between the Raspberry Pi and Battery Pack can be completed following the same procedure for connecting these two parts described in the Lab 2 manual. Additional connections to the motors and L298N H-bridge are required on the SCUTTLE and can be completed following the procedure that is described in the Lab 4 manual. Two more connections are required between the absolute rotary encoders and the Raspberry Pi and can be done following the procedure described in the Lab 5 manual. These connection schemes can be seen in Figure 1, Figure 2, and Figure 3.

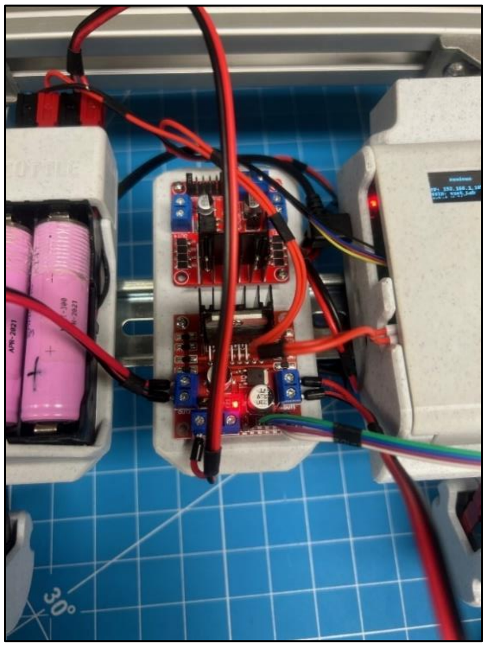


Figure 1: H-Bridge Connections

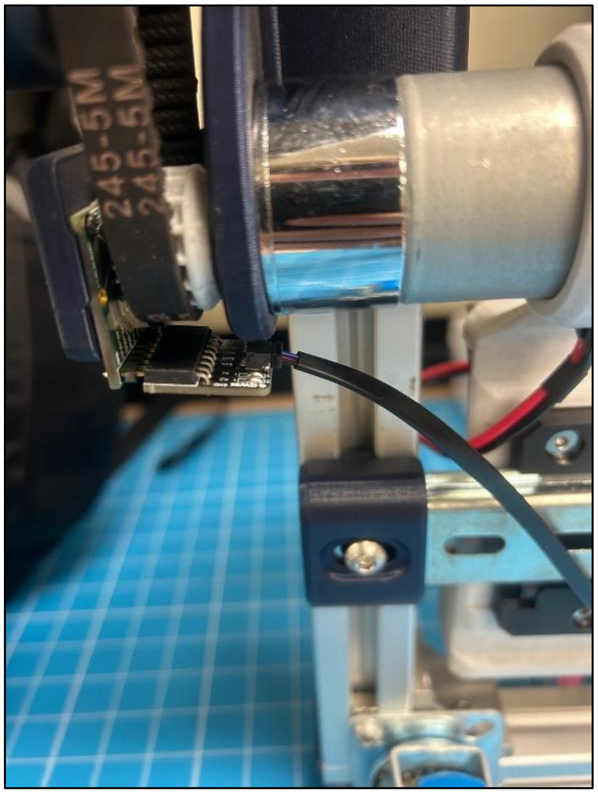


Figure 2: Encoder Connection (Same on Both Sides)

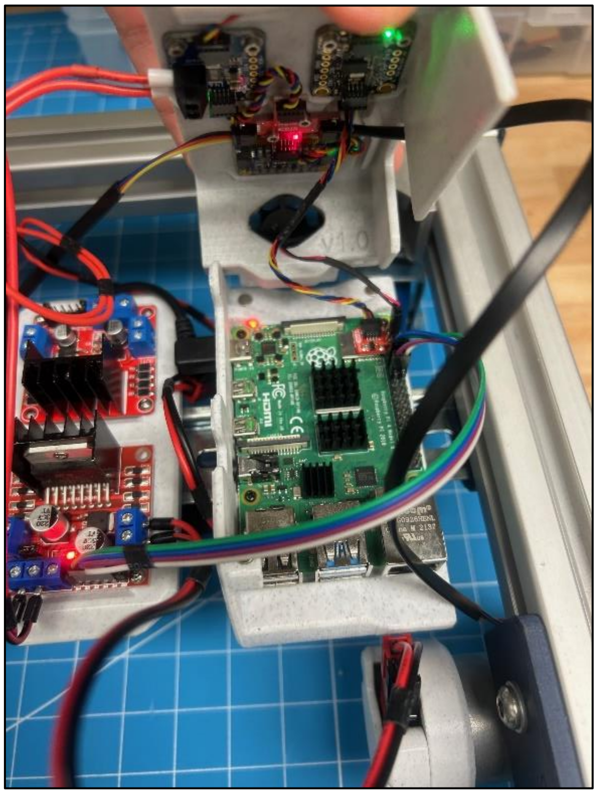


Figure 3: Raspberry Pi Enclosure Connections

For this lab, an additional camera is required to be set up and plugged into the Raspberry Pi. To process the camera's output, the MJPG-Streamer utility is utilized. This lightweight command-line tool streams camera frames to compatible applications, such as web browsers, VLC media player, or any software that supports MJPG streams. Its efficiency and low resource usage make it particularly suitable for embedded systems or mobile robotics platforms with limited CPU and memory capacity. In this experiment, the video feed is directed into a NodeRED workflow, which displays both the raw and processed images. Additionally, the interface allows for real-time adjustment of image filter parameters. The filtering process is based on HSV (Hue, Saturation, Value) color space, which offers advantages over the traditional RGB (Red, Green, Blue) model. HSV filtering enhances tracking accuracy by focusing on hue, which remains relatively constant under different lighting conditions, whereas RGB values can vary significantly with changes in illumination. To mount the sensor to the front of the SCUTTLE a hex tool must be used to loosen the hex screws and allow the connectors to slide into a slot in the SCUTTLE body ensuring that the sensor itself is facing forward. Once in the slot the hex screws can be tightened until the camera is securely connected to the SCUTTLE body. The final setup for the SCUTTLE with the camera can be seen in Figure 4.

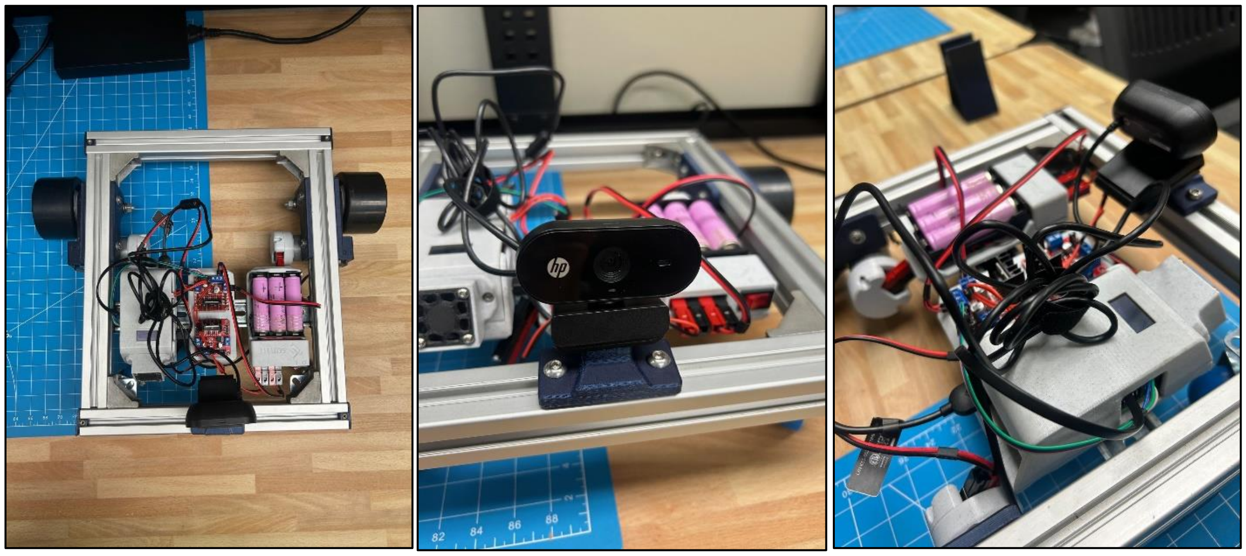
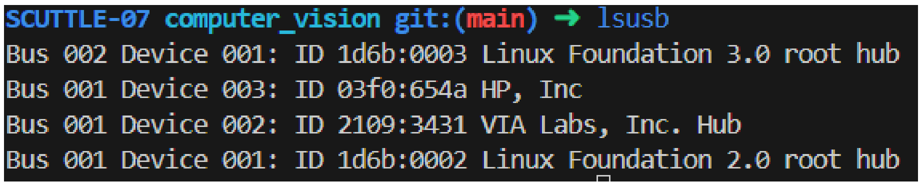


Figure 4: Complete SCUTTLE Physical Setup with All Connections

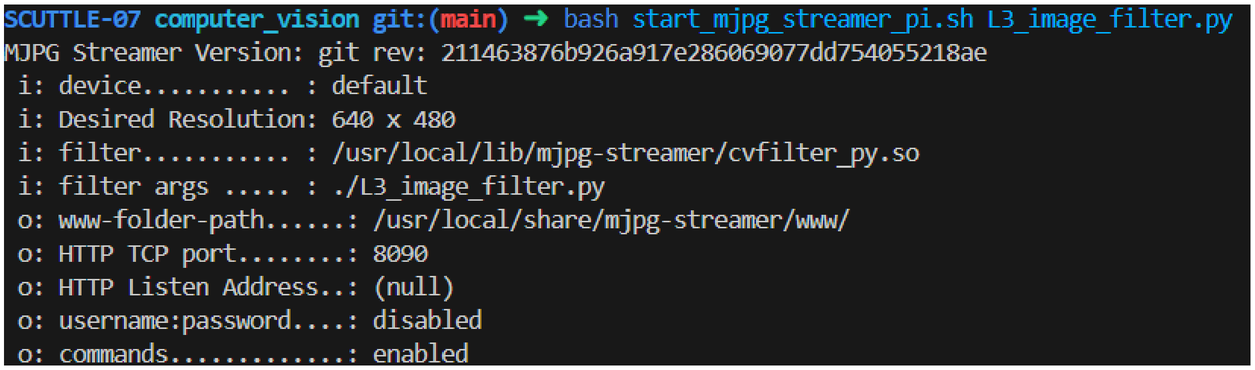
After checking the wiring and setup are correct the next step is to complete the booting and connecting procedure that are described in the Lab 1 manual. This will allow the Raspberry Pi to be accessed through Visual Code Studio's terminal over a wireless network. When the pi powers on connect to the network and open the VS code. In the VS code terminal window enter the "lsusb" command. This command lists all USB devices detected by the Raspberry Pi. While the camera is connected it will display the terminal output that can be seen in Figure 5. The camera must show up on this list to continue with the next parts of the lab.



```
SCUTTLE-07 computer_vision git:(main) ➜ lsusb
Bus 002 Device 001: ID 1d6b:0003 Linux Foundation 3.0 root hub
Bus 001 Device 003: ID 03f0:654a HP, Inc
Bus 001 Device 002: ID 2109:3431 VIA Labs, Inc. Hub
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
```

Figure 5: USB Devices Detected by the Pi with Camera Connected

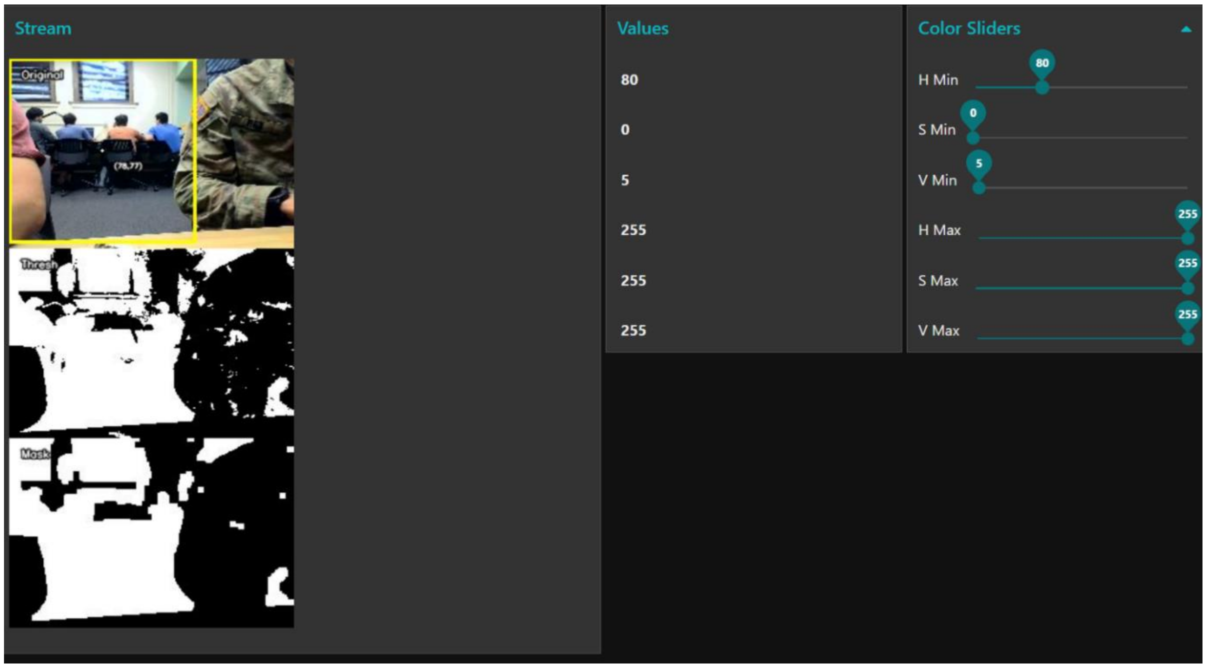
Based on Figure 5, it can be inferred that device 003 with the display name "HP, Inc" is the camera. With an operational camera, the next step involves creating a new directory in the basics folder using the "mkdir" command. A few files must be imported into this directory. To do this use the "cd" command to go into the basics directory and then the new directory. Once in this directory the "wget" command can be used, following the procedure in the Lab 2 manual, in the MXET300-SCUTTLE GitHub directory, to download and import start_mjpg_streamer_pi.sh and L3_image_filter.py. Follow this same procedure to import L3_color_tracking.py and nodered_color_tracking.json that that will be used later in the lab. Running these newly imported scripts with the command "bash start_mjpg_streamer_pi.sh L3_image_filter.py" will result in the terminal output seen in Figure 6.



```
SCUTTLE-07 computer_vision git:(main) ➜ bash start_mjpg_streamer_pi.sh L3_image_filter.py
MJPG Streamer Version: git rev: 211463876b926a917e286069077dd754055218ae
 i: device........... : default
 i: Desired Resolution: 640 x 480
 i: filter........... : /usr/local/lib/mjpg-streamer/cvfilter_py.so
 i: filter args ..... : ./L3_image_filter.py
 o: www-folder-path......: /usr/local/share/mjpg-streamer/www/
 o: HTTP TCP port........: 8090
 o: HTTP Listen Address..: (null)
 o: username:password....: disabled
 o: commands.............: enabled
```

Figure 6: Terminal Output for "bash start_mjpg_streamer_pi.sh L3_image_filter.py"

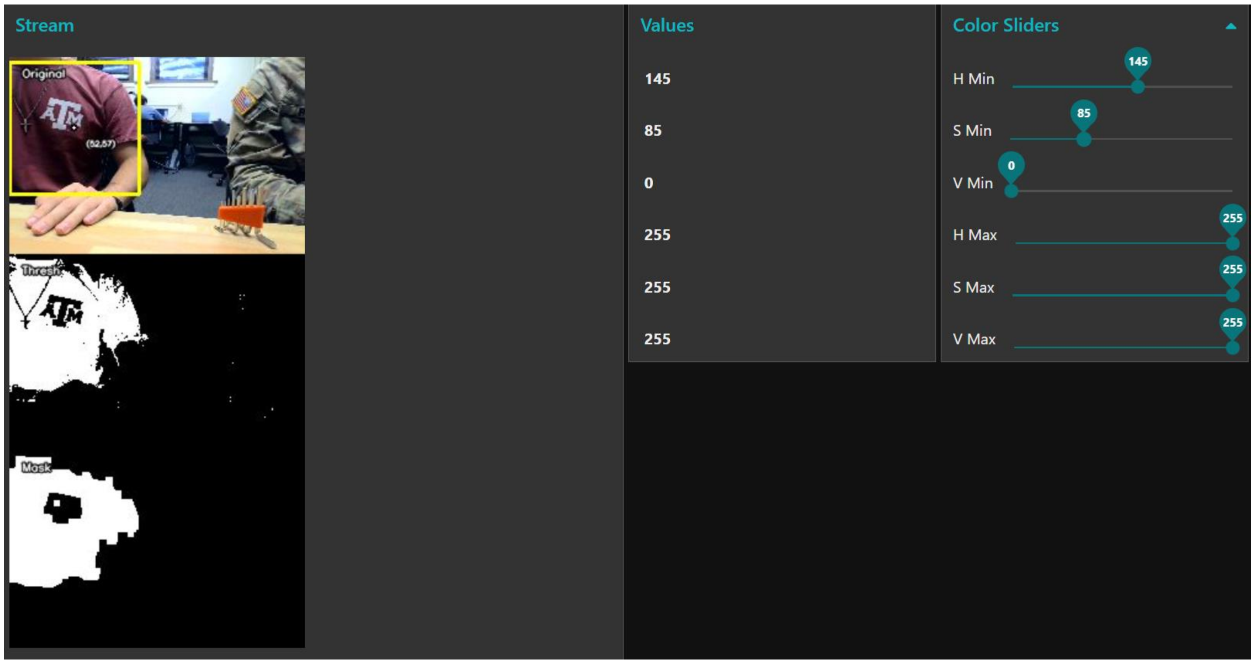
With this step complete the next step is to import and deploy the NodeRED flow for viewing the filter using the previously downloaded nodered_color_tracking.json file. Its important while doing this to match the IP address being used in the flow to that of the SCUTTLE's display. It should now display, on the dashboard, what can be seen in Figure 7.



H Min	80
S Min	0
V Min	5
H Max	255
S Max	255
V Max	255

Figure 7: Color Tracking NodeRED dashboard

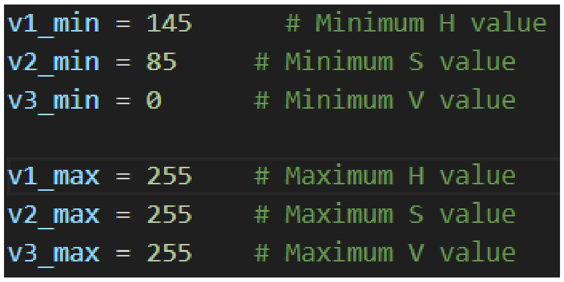
To effectively isolate the desired features in the camera feed, the HSV filter must be carefully tuned. The HSV (Hue, Saturation, Value) color space is particularly useful for image processing because it separates color information from intensity and brightness. Hue corresponds to the actual color, and is typically set to a narrow range when targeting specific objects. In contrast, saturation and value should be set to broader ranges to account for variations in lighting and reflectivity, which can affect how colors appear on camera. Initially, the filter settings may not exclude any part of the image, as indicated by a fully white output in the filtered view, meaning that all pixels are currently within the specified thresholds. To begin calibration, vivid, easily recognizable objects can be placed in front of the camera to act as visual targets. A red t-shirt was chosen as the object for this lab. The HSV threshold sliders are then adjusted until only the object of interest remains visible in the filtered output. This tuning process is largely iterative and may require several adjustments to achieve optimal results. The goal is to create a filtered image where the target appears mostly solid with only minimal distortion around the edges. Figure 8 demonstrates a successful outcome of this filtering process, where the target is clearly identified and isolated from the background using well-adjusted HSV parameters.



Color Sliders	Values
H Min	145
S Min	85
V Min	0
H Max	255
S Max	255
V Max	255

Figure 8: HSV Filtered Image Color Tracking NodeRED Dashboard

The next portion of this lab involves the SCUTTLE robot using image data to actively track and follow a target. This demonstrates a fundamental application of computer vision in mobile robotics. The control algorithm identifies the largest detected object in the camera feed and calculates its center coordinate. Based on the horizontal position of this coordinate relative to the center of the image frame, the robot determines whether to turn left or right to align itself with the target. When the object is close to the center of the image, the robot then estimates the target's distance using the object's apparent area in the frame. By monitoring this area, the system attempts to maintain a consistent distance from the target, adjusting motor outputs accordingly. This method enables the robot to follow a moving object with basic visual feedback and proportional motor control. After opening the previously downloaded L3_color_tracking.py file, the lines that control the color range, described in HSV, can be updated with the values found in the previous part of the lab. These updated values in the code can be seen in Figure 9.

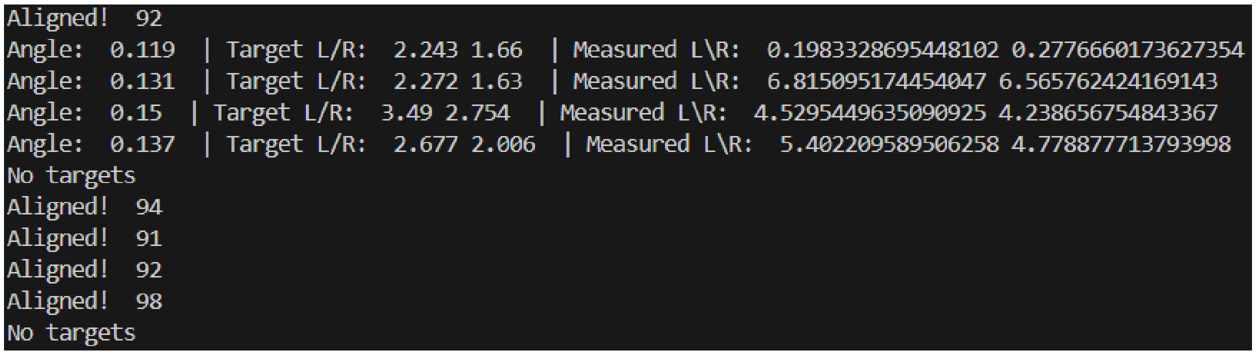


```
v1_min = 145      # Minimum H value
v2_min = 85      # Minimum S value
v3_min = 0       # Minimum V value

v1_max = 255     # Maximum H value
v2_max = 255     # Maximum S value
v3_max = 255     # Maximum V value
```

Figure 9: New HSV Minimum and Maximum Values

With the new values set and ready to be used the next step is to return to the original terminal window the MJPG-Streamer script was ran in order to end it with Ctrl+C. Run this script with the updated values using the "sudo python3 L3_color_tracking.py" command in the terminal window. Doing so will prompt the robot to begin following the obstacle and will in return produce a terminal output similar to that which can be seen in Figure 10.



```
Aligned!  92
Angle:  0.119  | Target L/R:  2.243 1.66  | Measured L\R:  0.1983328695448102 0.2776660173627354
Angle:  0.131  | Target L/R:  2.272 1.63  | Measured L\R:  6.815095174454047 6.565762424169143
Angle:  0.15  | Target L/R:  3.49 2.754  | Measured L\R:  4.5295449635090925 4.238656754843367
Angle:  0.137  | Target L/R:  2.677 2.006  | Measured L\R:  5.402209589506258 4.778877713793998
No targets
Aligned!  94
Aligned!  91
Aligned!  92
Aligned!  98
No targets
```

Figure 10: Terminal Output for "sudo python3 L3_color_tracking.py"

Conclusion

This lab demonstrated the integration of computer vision into mobile robotics through the use of HSV filtering to enable the SCUTTLE robot to detect and follow a target object based on color. The experiment highlighted the effectiveness of real-time image processing and visual feedback for autonomous navigation. Key takeaways included the importance of selecting the appropriate color space for tracking, with HSV proving more reliable than RGB under varying lighting conditions. Challenges encountered included initial camera recognition issues and the iterative nature of tuning HSV values for accurate filtering, both of which emphasized the need for precision in sensor configuration and environmental control. Overall, the lab reinforced core concepts in robotics related to image-based object tracking and control, providing practical experience in applying vision systems to autonomous platforms.

Post-Lab Questions

1. How can computer vision be used in robotics? Give three examples with at least one industrial application.

Computer vision improves robots' capacity to understand and engage with their surroundings. Autonomous navigation systems, quality inspection, and object detection are three of the most common applications. Autonomous Navigation Robots utilize visual input to trace lines or objects, steer clear of obstacles, and create environmental maps using image data. Quality Inspection in Manufacturing utilizes quick cameras and vision systems to identify defects in production on production lines, ensuring uniform quality and minimizing the need for human inspections. Object Detection and Sorting uses robotic arms fitted with cameras to recognize objects based on physical attributes like color, shape, or size and sort them accordingly. This practice is commonly seen in recycling facilities or package fulfillment centers such as Amazon warehouses.

2. Explain what color space is and why we use HSV instead of RGB for color tracking.

Color space is the specific organization of colors used in combination with color profiling by various physical devices (Wikimedia Foundation). HSV was chosen over RGB as the hue directly correlates to the specific color allowing the isolation of a desired color and shade to be found promptly. The saturation and value separate the brightness and color intensity from the hue further implementing the ability to quickly identify various shades of the same base color in different environments. While RGB is sensitive to changes in light, the HSV filtering provides a larger scope of comprehension based on changes in lighting to identify the same color.

Appendix

Lab 8 Commands Used:

- Insert Text Here
- python3 lsusb
- mkdir computer_vision
- wget start_mjpg_streamer_pi.sh
- wget L3_image_filter.py
- bash start_mjpg_streamer_pi.sh L3_image_filter.py
- wget nodered_color_tracking.json
- sudo python3 L3_color_tracking.py

Lab 7 Commands Used:

- python3 L1_motors.py
- python3 L1_encoder.py
- wget Lab7Template.py
- python3 L2_speed_control.py
- sudo mv /tmp/excel_data.csv

Lab 6 Commands Used:

- wget L1_lidar.py
- sudo python3 L1_lidar.py
- wget L2_vector.py
- sudo python3 L2_vector.py
- touch L3_logs

- log.tmpFile(voltage, "voltage_log.txt")
- log.tmpFile(B[0], "L2_dist.txt")
- log.tmpFile(B[1], "L2_angle.txt")
- log.tmpFile(C[0], "L2_x.txt")
- log.tmpFile(C[1], "L2_y.txt")
- git add .
- git commit -m “lab6”
- git push

Lab 5 Commands Used:
- sudo watch -n0.2 i2cdetect -y -r 1
- wget
- python3 L1_encoder.py
- wget
- python3 L2_kinematics.py
- python3 L1_motor.py
- ctrl + c
- log.tmpFile(voltage, "voltage_log.txt")
- log.tmpFile(C[0], "xdot.txt")
- log.tmpFile(C[1], "thetadot.txt")
- log.tmpFile(B[0], "pdl.txt")
- log.tmpFile(B[1], "pdr.txt")
- python3 L3_log_speeds.py
- ctrl + c
- python3 L3_log_speeds.py
- python3 L3_path_template.py

Lab 4 Commands Used:
- pwd
- python3 L3_path_template.py
- python3 L3_noderedControl.py
- git add .
- git commit -m “Lab 4”
- git push

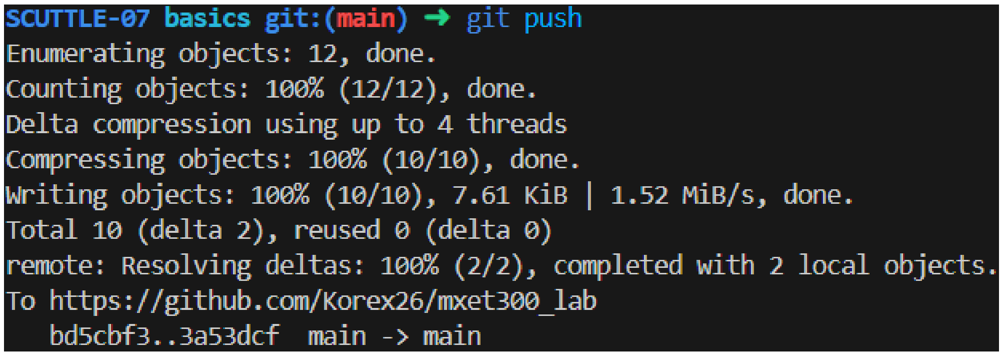
Lab 3 Commands Used:
- cd basics/
- wget
- python3 L2_compass_heading.py
- python3 L3_Compass.py
- tmpFile
- stringTmpFile
- git add .
- git commit -m “Lab 3”

- git push

Lab 2 Commands Used:
- cd
- mkdir basics
- pwd
- wget
- python3 L1_ina.py
- sudo systemctl stop oled.service
- sudo mv oled.py/usr/share/pyshared/oled.py
- git status
- git add .
- git commit -m
- git push

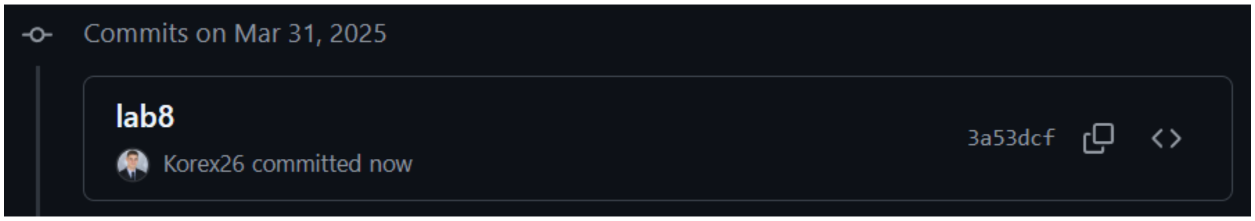
GitHub Submission

All required files were pushed to the team's repository for review and backup as seen in the GitHub Submission Figure 1 and GitHub Submission Figure 2. The repository can be found using the following link: https://github.com/Korex26/mxet300_lab.



```
SCUTTLE-07 basics git:(main) ➜ git push
Enumerating objects: 12, done.
Counting objects: 100% (12/12), done.
Delta compression using up to 4 threads
Compressing objects: 100% (10/10), done.
Writing objects: 100% (10/10), 7.61 KiB | 1.52 MiB/s, done.
Total 10 (delta 2), reused 0 (delta 0)
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To https://github.com/Korex26/mxet300_lab
   bd5cbf3..3a53dcf  main -> main
```

GitHub Submission Figure 1: Commit and Push to GitHub on Visual Studio Code



Commits on Mar 31, 2025

lab8

Korex26 committed now

3a53dcf

GitHub Submission Figure 2: Commit and Push to GitHub on GitHub

References

Wikimedia Foundation. (2025b, March 14). *Color space*. Wikipedia. https://en.wikipedia.org/wiki/Color_space

K. Rex, R. Worley, *MXET 300 Lab Report 7*, 2025.

K. Rex, R. Worley, *MXET 300 Lab Report 6*, 2025.

K. Rex, R. Worley, *MXET 300 Lab Report 5*, 2025.

K. Rex, R. Worley, *MXET 300 Lab Report 4*, 2025.

K. Rex, R. Worley, *MXET 300 Lab Report 3*, 2025.

K. Rex, R. Worley, *MXET 300 Lab Report 2*, 2025.

X. Song, *Lab 8 Manual* - *Computer Vision*, 2025.

X. Song, *Lab 7 Manual* - *Closed-Loop Control*, 2025.

X. Song, *Lab 6 Manual* - *LIDAR and Obstacle Detection*, 2025.

X. Song, *Lab 5 Manual* - *Encoders _ Forward Kinematics*, 2025.

X. Song, *Lab 4 Manual* - *Motor Drivers _ Inverse Kinematics*, 2025.

X. Song, *Lab 3* - *Compass Calibration*, 2025.

X. Song, *Lab 2 Manual* - *Displays and GUI*, 2025.

X. Song, *Lab 1 Manual* - *Pi Setup*, 2025.