Lab 8 - Computer Vision

Overview

This lab introduces the topic of computer vision and image processing. You will use a USB camera with the SCUTTLE to capture images, create a "mask" with a color range filter, and find the centroid of the masked area to estimate the target's position in the image. This information can finally be used to control the robot's heading and speed for following the target.

Resources:
The following are links to useful resources for this lab:

- SCUTTLE Homepage
- SCUTTLE GitHub
- Guides and resources
- RGB and HSV color picker

Software Needed:

- Python scripts:
  - L3_image_filter.py
  - L3_color_tracking.py
- NodeRED flows:
  - nodered_color_tracking.json
- Other:
  - start_mjpg_streamer_pi.sh

Run MJPG Streamer and NodeRED Flow

To handle the output of the camera, we use MJPG-Streamer. This is a command line application which will stream the frames from the camera to another application. It can be used to stream a camera to a browser, VLC, or other software capable of receiving MJPG streams. Since it is written for use on devices with limited resources (RAM and CPU) it is ideal for a small mobile robot.

This camera stream will be sent to a NodeRED flow, which shows the raw images, filtered images, and provides an interface for adjusting the filter thresholds during runtime. In this lab, image filtering is done using HSV (hue-saturation-value) thresholds. This is a more useful metric than RGB (red-green-blue) because real objects don't typically change their color, but lighting can influence the brightness and saturation of the object perceived by the camera. HSV filtering improves color tracking performance by taking advantage of this.

1. Plug the camera into the Pi and use the command `lsusb` to verify it is detected. Pic1

2. Now you will start the streamer and the filter script.
    a. In your basics folder, create a new directory called *computer_vision* using the *mkdir* command.
    b. Download *start_mjpg_streamer_pi.sh* and *L3_image_filter.py* to this directory.
    c. Run this command to run the filter script with the streamer as an input: Pic2 terminal
    `bash start_mjpg_streamer_pi.sh L3_image_filter.py`
3. Next the NodeRED flow for viewing the filter needs to be deployed.
    a. Download the NodeRED flow *nodered_color_tracking.json* and import it to NodeRED.
    b. Find the blue node at the top of the flow labeled "Stream" and open it. The menu should look like figure 1. Change the IP address at the beginning of the line to match the one on your SCUTTLE's display. This needs to be done for any new network you connect to, however in the lab it only needs to be done once since your Pi uses a static IP. **DO NOT** change port number.
    c. Verify that the deployed dashboard looks like figure 2. Pic3 dashboard

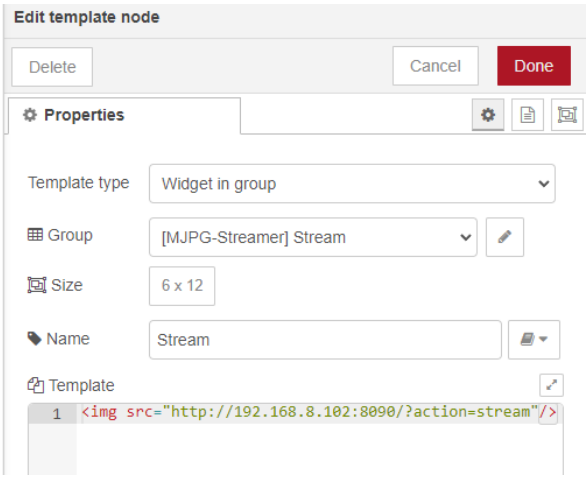

Figure 1: NodeRED Stream template

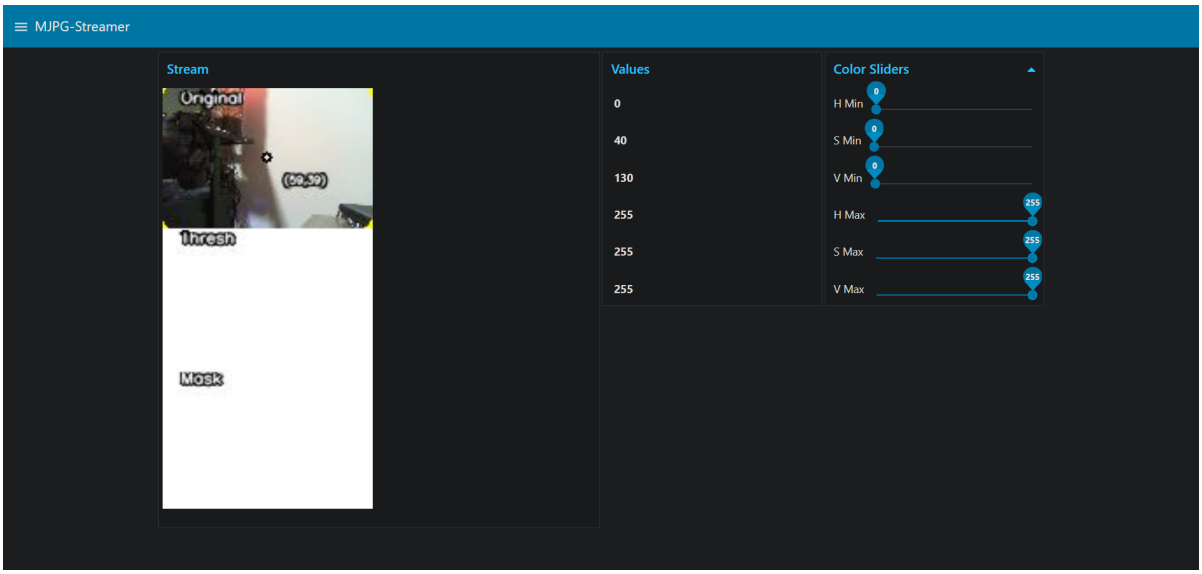

Figure 2: Color tracking NodeRED dashboard

Tuning the HSV Filter

Now, the HSV filter must be tuned to extract the desired information from the image. Remember that the HSV color space measures color, intensity, and brightness. Hue should be a small range of values, saturation will vary, and value should be a larger range to account for different brightnesses caused by changing lighting conditions.

1. With the current settings, the filter is not masking any pixels from the image. This is indicated by the completely white images below the unfiltered view, which means that every pixel falls within the threshold.
2. Use any vivid colored objects to use as vision targets.
3. Place the target in front of the camera and adjust the sliders until only the target is visible.
    d. This process is somewhat trial and error. Tune the values until the object is mostly solid with slight distortion around the edges.
    e. Remember that this is an HSV filter, where hue corresponds to the "color" and saturation/value describe the "amount of color" and brightness.
    f. Figure 3 shows some acceptable results for the filtered images.

Pic4

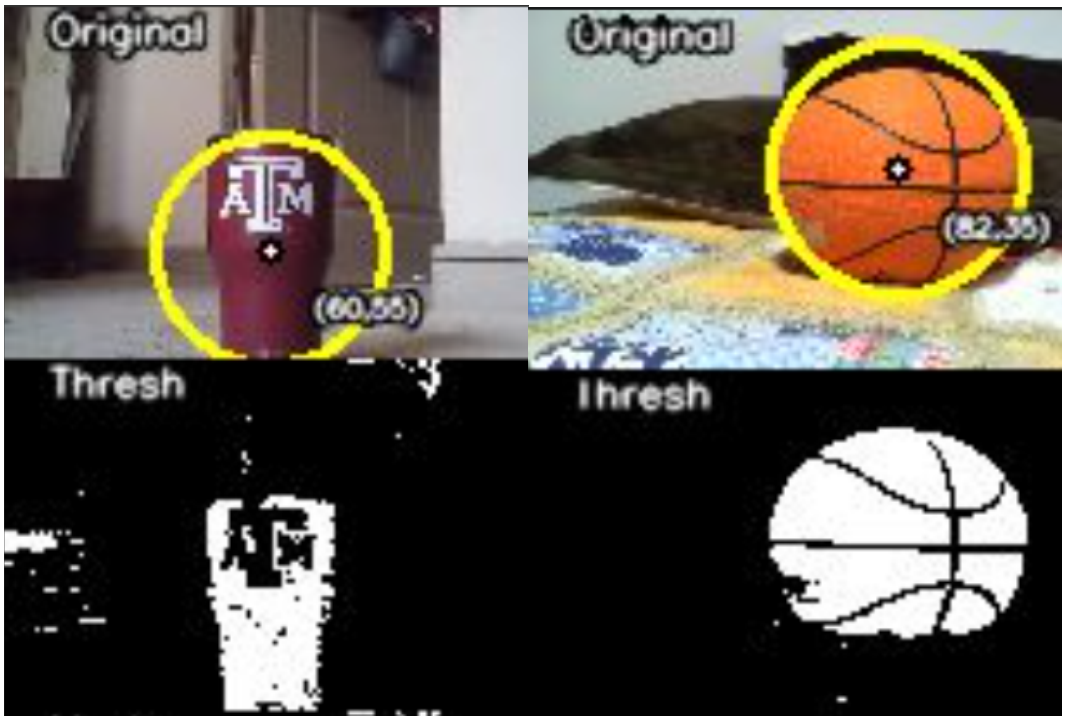

Figure 3: HSV filtered image examples

Control SCUTTLE with Color Tracking

Finally, your SCUTTLE will track its target in the image and actuate its motors to follow the target around. This is a simple application of computer vision in a mobile robot. The program will find the center coordinate of the largest detected area in the image and turn the SCUTTLE left or right depending on which side of the image the coordinate is on. If the coordinate is within a certain range of the center, it roughly estimates distance using the area of the object and tries to keep it at a certain range.

```
#    Color Range, described in HSV

v1_min = 0     # Minimum H value
v2_min = 0     # 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 4: Default HSV minima and maxima

1. Download the *L3_color_tracking.py* script from GitHub and find the lines shown in figure 4.
2. Input the range of values you determined in the previous exercise to track the target. Pic6 NEW VALUES
3. Find the terminal window where you originally ran the MJPG-Streamer script and use Ctrl+C to stop it.
4. Run the newly downloaded script using `sudo python3 L3_color_tracking.py` Pic7 terminal
5. Your SCUTTLE should attempt to follow the obstacle.
    a. You can adjust the variable ‘target_width’ to vary the target distance for the obstacle. It is the width in pixels of the detected object, which relates to distance, that the robot will try to maintain. Change this to keep the target at varying distances.
    b. Variables ‘angle_margin’ and ‘width_margin’ are the margins that define when an object is centered or at the correct distance in radians and pixels, respectively. These will need to be tuned for your chosen target.
6. Finally, save your work and commit the changes. Then push the changes to your remote repository along with the NodeRED flow used for visualizing the filter.

Post-Lab Questions

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

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