Shivam Prakash Jha

219619

Task: Detect the color of the Traffic light and Provide appropriate indication to driver by displaying message.

Algorithm:

1. Main Function

1.1. Read the input image from the specified file path. 1.2. Check if the image was successfully loaded. 1.3. Call the function to detect and display traffic lights. 1.4. Return from the main function.

2. Detect and Display Traffic Lights Function

2.1. Preprocess the input image to convert it to the HSV color space. 2.2. Get the bounding boxes of potential traffic lights based on color detection. 2.3. Iterate over each bounding box: 2.3.1. Draw a rectangle around the detected traffic light. 2.3.2. Determine the color of the traffic light. 2.3.3. Print the detected color to the console. 2.4. If no traffic lights are detected, print a message to the console. 2.5. Display the original image with the bounding boxes drawn. 2.6. Wait for a key press to close the display window.

3. Preprocess Image Function

3.1. Convert the input image from BGR to HSV color space. 3.2. Return the HSV image.

4. Get Traffic Light Bounds Function

4.1. Detect red color in the HSV image using two ranges (for different shades of red). 4.2. Detect yellow color in the HSV image. 4.3. Detect green color in the HSV image. 4.4. Combine the masks for red, yellow, and green colors. 4.5. Find contours in the combined mask. 4.6. Iterate over each contour: 4.6.1. Get the bounding rectangle for the contour. 4.6.2. Filter out small contours that are unlikely to be traffic lights. 4.6.3. Store the bounding box of the detected traffic light. 4.7. Return the bounding boxes of all detected traffic lights.

5. Get Traffic Light Color Function

5.1. Extract the region of the HSV image corresponding to the traffic light bounding box. 5.2. Initialize counters for red, yellow, and green pixel counts. 5.3. Detect red color in the traffic light region using two ranges (for different shades of red) and update the red counter. 5.4. Detect yellow color in the traffic light region and update the yellow counter. 5.5. Detect green color in the traffic light region and update the green counter. 5.6. Compare the pixel counts for red, yellow, and green: 5.6.1. Return "Red" if red pixel count is the highest. 5.6.2. Return "Yellow" if yellow pixel count is the highest. 5.6.3. Return "Green" if green pixel count is the highest. 5.6.4. Return "Unknown" if no color is dominant.

Code:

```
#include <opencv2/opencv.hpp>
```

```
#include <iostream>
#include <vector>

using namespace cv;
using namespace std;

// Function declarations
void detectAndDisplayTrafficLights(const Mat& image);
Mat preprocessImage(const Mat& image);
vector<Rect> getTrafficLightBounds(const Mat& hsvImage);
string getTrafficLightColor(const Mat& hsvImage, const Rect& trafficLightBound);

// Main function
int main() {

Mat image = imread("/home/kpit/build/3rd_half/OneDrive_1_04-06-2024/traffic_light4.png");

if (image.empty()) {
cout << "Could not open or find the image!" << endl;
return -1;
}

// Detect and display the traffic lights
detectAndDisplayTrafficLights(image);

return 0;
}

// Function to detect and display the traffic lights
void detectAndDisplayTrafficLights(const Mat& image) {
Mat hsvImage = preprocessImage(image);
vector<Rect> trafficLightBounds = getTrafficLightBounds(hsvImage);
```

```
for (const Rect& bound : trafficLightBounds) {
rectangle(image, bound, Scalar(0, 255, 0), 2);
string color = getTrafficLightColor(hsvImage, bound);
cout << "Traffic Light Color: " << color << endl;
}

if (trafficLightBounds.empty()) {
cout << "No traffic lights detected." << endl;
}

// Show the original image with the bounding boxes
imshow("Traffic Light Detection", image);
waitKey(0);
}

// Function to preprocess the image (convert to HSV color space)
Mat preprocessImage(const Mat& image) {
Mat hsvImage;
cvtColor(image, hsvImage, COLOR_BGR2HSV);
return hsvImage;
}

// Function to get the bounding boxes of the traffic lights based on color detection
vector<Rect> getTrafficLightBounds(const Mat& hsvImage) {
Mat redMask1, redMask2, yellowMask, greenMask, combinedMask;
vector<vector<Point>> contours;
vector<Vec4i> hierarchy;

// Detect red color (two ranges)
inRange(hsvImage, Scalar(0, 100, 100), Scalar(10, 255, 255), redMask1);
inRange(hsvImage, Scalar(160, 100, 100), Scalar(180, 255, 255), redMask2);
```

```
Mat redMask = redMask1 | redMask2;

// Detect yellow color
inRange(hsvImage, Scalar(20, 100, 100), Scalar(30, 255, 255), yellowMask);

// Detect green color
inRange(hsvImage, Scalar(40, 100, 100), Scalar(70, 255, 255), greenMask);

// Combine masks for contours detection
combinedMask = redMask | yellowMask | greenMask;

// Find contours
findContours(combinedMask, contours, hierarchy, RETR_TREE, CHAIN_APPROX_SIMPLE);

// Store bounding boxes of all detected traffic lights
vector<Rect> trafficLightBounds;
for (const auto& contour : contours) {
Rect boundingBox = boundingRect(contour);
// Filter out small contours that are unlikely to be traffic lights
if (boundingBox.width > 20 && boundingBox.height > 20) {
trafficLightBounds.push_back(boundingBox);
}
}

return trafficLightBounds;
}

// Function to determine the color of the traffic light
string getTrafficLightColor(const Mat& hsvImage, const Rect& trafficLightBound) {
Mat redMask1, redMask2, yellowMask, greenMask;
Mat trafficLightRegion = hsvImage(trafficLightBound);
```

```
int redCount = 0, yellowCount = 0, greenCount = 0;

// Detect red color
inRange(trafficLightRegion, Scalar(0, 100, 100), Scalar(10, 255, 255), redMask1);
inRange(trafficLightRegion, Scalar(160, 100, 100), Scalar(180, 255, 255), redMask2);
redCount = countNonZero(redMask1) + countNonZero(redMask2);

// Detect yellow color
inRange(trafficLightRegion, Scalar(20, 100, 100), Scalar(30, 255, 255), yellowMask);
yellowCount = countNonZero(yellowMask);

// Detect green color
inRange(trafficLightRegion, Scalar(40, 100, 100), Scalar(70, 255, 255), greenMask);
greenCount = countNonZero(greenMask);

if (redCount > yellowCount && redCount > greenCount) {
return "Red";
} else if (yellowCount > redCount && yellowCount > greenCount) {
return "Yellow";
} else if (greenCount > redCount && greenCount > yellowCount) {
return "Green";
} else {
return "Unknown";
}
}
```

Sample Input/ Output Image: