Traffic Light Detection

Karthik Challa, E.C.E.
Paul Huch, M.E
Isha Kulkarni, B.M.E.
Zhiming Qian, E.C.E.

Overview

Roles & Responsibilities

Approach

Working Code

Simulation

Q&A

Project Overview

System programmed to detect the color of a traffic light and perform the appropriate control of the vehicle

Roles & Responsibilities

- Karthik Challa, E.C.E.
  - System
- Paul Huch, E.C.E.
  - Control
- Isha Kulkarni, B.M.E.
  - Introduction and overview
- Zhiming Qian, E.C.E.
  - Color detection

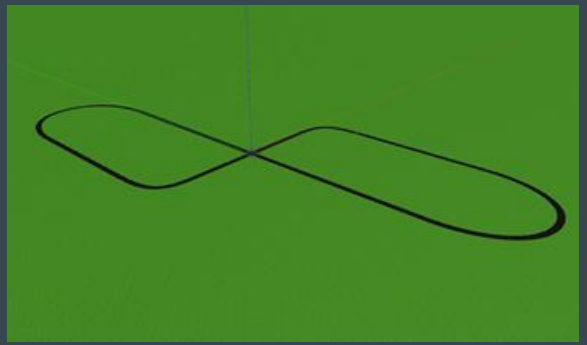
Approach

Simulation setup to follow the trajectory shown in the figure

Location of the traffic lights is obtained through GPS and TF Transforms

Traffic Light camera focussed on getting the images of traffic lights

Vehicle speed controlled through ULC CMD



Tools

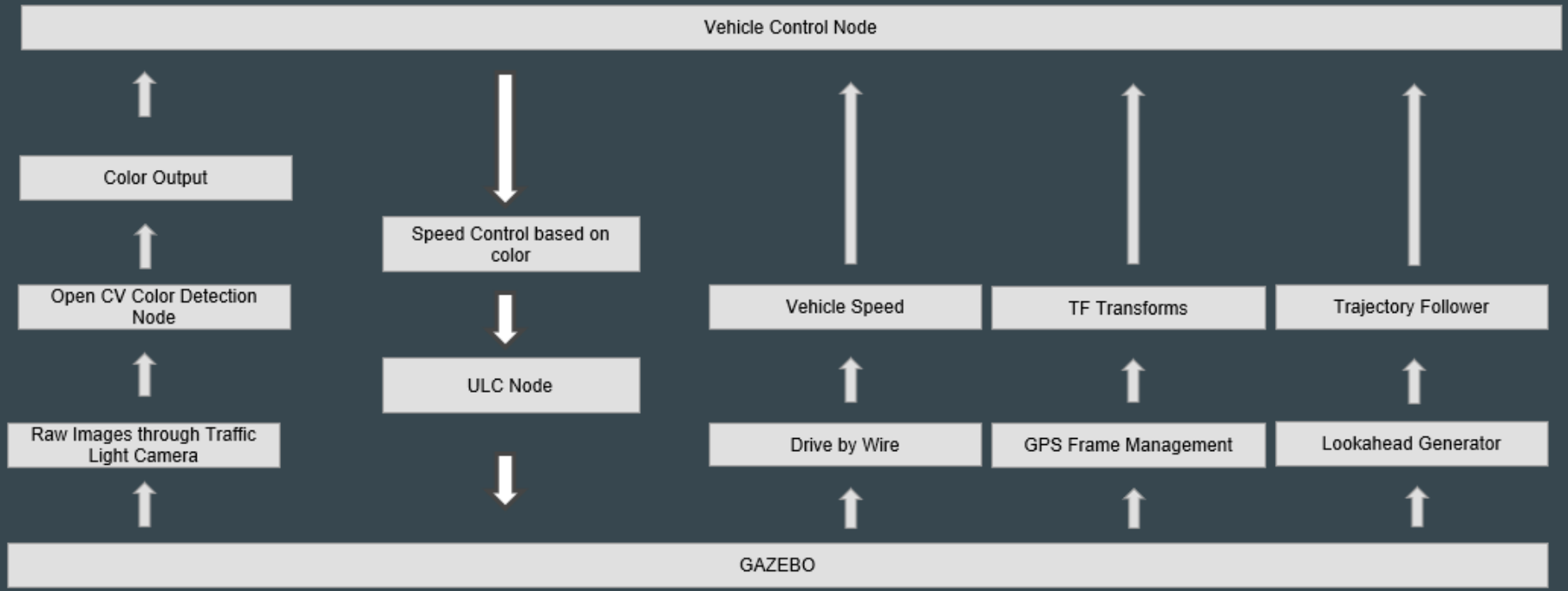
ROSCPP

TF Transforms

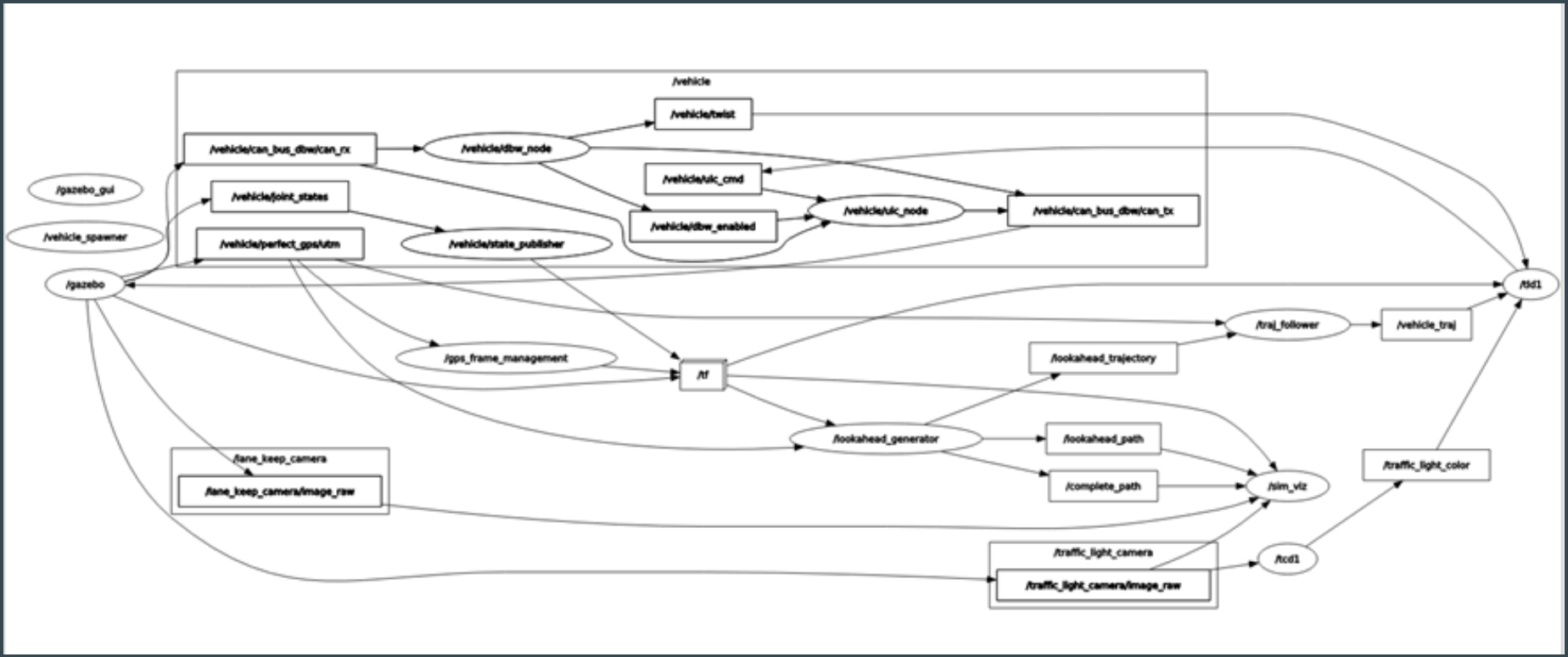
Dataspeed ULC CMD

OPEN CV

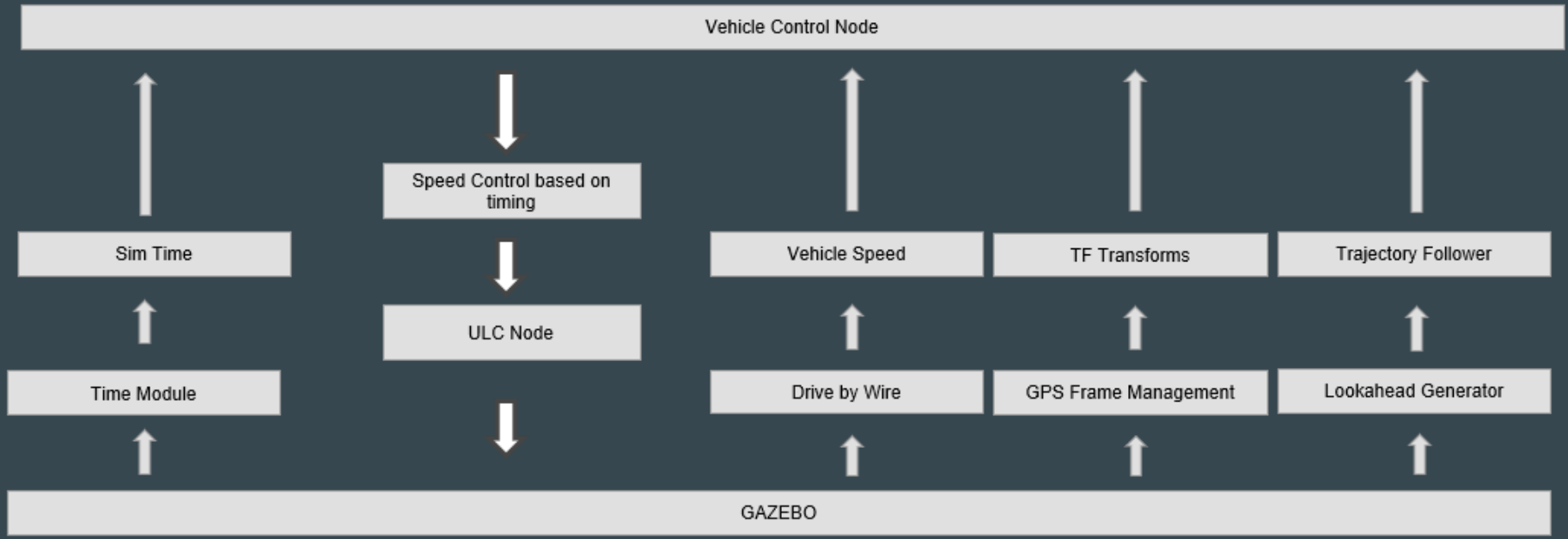
System Overview - A - Color Based Control



RQT Graph


/vehicle
/vehicle/twist
/vehicle/can_bus_dbw/can_rx
/vehicle/dbw_node
/vehicle/ulc_cmd
/gazebo_gui
/vehicle/joint_states
/vehicle/dbw_enabled
/vehicle/ulc_node
/vehicle/can_bus_dbw/can_tx
/vehicle_spawner
/vehicle/perfect_gps/utm
/vehicle/state_publisher
/gazebo
/tfd1
/traj_follower
/vehicle_traj
/gps_frame_management
/tf
/lookahead_trajectory
/lookahead_generator
/lookahead_path
/lane_keep_camera
/complete_path
/traffic_light_color
/lane_keep_camera/image_raw
/sim_viz
/traffic_light_camera
/tcd1
/traffic_light_camera/image_raw

System Overview - B - Time Based Control

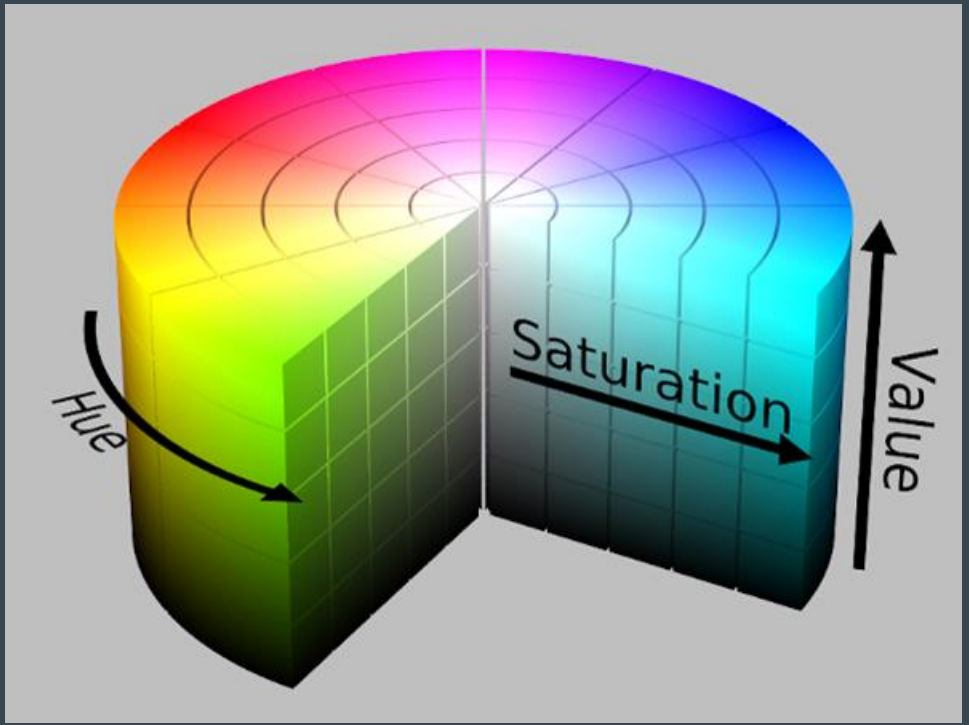


Color Detection

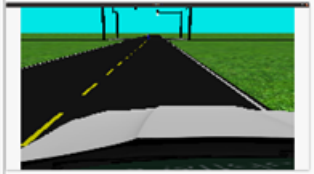
Two Main Code

1. Hue Saturation Value Hexcone Model to detect color

1. HoughCircles to find circles



```
int detected_r = 0, detected_y = 0, detected_g = 0, total = 0;
cv::inRange(raw_hsv_img, cv::Scalar(config.r1_h_l, config.r1_s_l, config.r1_v_l), cv::Scalar(config.r1_h_h, config.r1_s_h, config.r1_v_h), r_threshold);
cv::inRange(raw_hsv_img, cv::Scalar(config.r2_h_l, config.r2_s_l, config.r2_v_l), cv::Scalar(config.r2_h_h, config.r2_s_h, config.r2_v_h), r_threshold1)
// color: green
//cv::inRange(raw_hsv_img, cv::Scalar(40, 50, 200), cv::Scalar(90, 255, 255), g_threshold);
cv::inRange(raw_hsv_img, cv::Scalar(config.g1_h_l, config.g1_s_l, config.g1_v_l), cv::Scalar(config.g1_h_h, config.g1_s_h, config.g1_v_h), g_threshold);
// color: yellow
cv::inRange(raw_hsv_img, cv::Scalar(config.y1_h_l, config.y1_s_l, config.y1_v_l), cv::Scalar(config.y1_h_h, config.y1_s_h, config.y1_v_h), y_threshold);
```



detected circles

r_threshold

y_threshold

g_threshold

Color Detection: HSV Part

Edit Selection View Go Run Terminal Help

EXPLORER

OPEN EDITO... 1 UNSAVED

SRC

.vscode

fall-2020-final-projec...

group5_sim_launch

cfg

traffic_light.cfg

launch

traffic_light.launch

rviz

src

traffic_light.cpp

traffic_light.hpp

traffic_node.cpp

test_results

urdf

yaml

CMakeLists.txt

ForTest_Green_1.j...

ForTest_Green_2.j...

ForTest_Green_3.j...

ForTest_Green_4.j...

ForTest_Red_1.jpg

ForTest_Red_2.jpg

ForTest_Red_3.jpg

ForTest_Red_4.jpg

ForTest_Yellow_1....

ForTest_Yellow_2....

ForTest_Yellow_3....

ForTest_Yellow_4....

Makefile

package.xml

homework3.launch | Homework3.cpp | traffic_light.cpp | traffic_node.cpp | CMakeLists.txt | HsvEx

fall-2020-final-project-group_5-main > group5_sim_launch > cfg > traffic_light.cfg

```
#! /usr/bin/env python
PACKAGE='opencv_traffic_lights'

from dynamic_reconfigure.msg import SensorLevels
from dynamic_reconfigure.parameter_generator_catkin import *

gen = ParameterGenerator()

#       Name            Type    Level   Description             Default  Min   Max
gen.add("r1_h_h",    int_t, 0,       "Hue threshold center",   1,       0,    255)
gen.add("r1_h_l",    int_t, 0,       "Hue threshold width",    0,       0,    255)
gen.add("r1_s_h",    int_t, 0,       "Saturation threshold",   255,     0,    255)
gen.add("r1_s_l",    int_t, 0,       "Value threshold",        254,     0,    255)
gen.add("r1_v_h",    int_t, 0,       "Value threshold",        255,     0,    255)
gen.add("r1_v_l",    int_t, 0,       "Hue threshold center",   105,     0,    255)

gen.add("r2_h_h",    int_t, 0,       "Hue threshold center",   255,     0,    255)
gen.add("r2_h_l",    int_t, 0,       "Hue threshold width",    255,     0,    255)
gen.add("r2_s_h",    int_t, 0,       "Saturation threshold",   255,     0,    255)
gen.add("r2_s_l",    int_t, 0,       "Value threshold",        255,     0,    255)
gen.add("r2_v_h",    int_t, 0,       "Value threshold",        255,     0,    255)
gen.add("r2_v_l",    int_t, 0,       "Hue threshold center",   255,     0,    255)

gen.add("y1_h_h",    int_t, 0,       "Hue threshold center",   35,      0,    255)
gen.add("y1_h_l",    int_t, 0,       "Hue threshold width",    10,      0,    255)
gen.add("y1_s_h",    int_t, 0,       "Saturation threshold",   255,     0,    255)
gen.add("y1_s_l",    int_t, 0,       "Value threshold",        0,       0,    255)
gen.add("y1_v_h",    int_t, 0,       "Value threshold",        255,     0,    255)
gen.add("y1_v_l",    int_t, 0,       "Hue threshold center",   255,     0,    255)

gen.add("g1_h_h",    int_t, 0,       "Hue threshold center",   60,      0,    255)
gen.add("g1_h_l",    int_t, 0,       "Hue threshold width",    60,      0,    255)
gen.add("g1_s_h",    int_t, 0,       "Saturation threshold",   255,     0,    255)
gen.add("g1_s_l",    int_t, 0,       "Value threshold",        255,     0,    255)
gen.add("g1_v_h",    int_t, 0,       "Value threshold",        255,     0,    255)
gen.add("g1_v_l",    int_t, 0,       "Hue threshold center",   103,     0,    255)
```

rqt_reconfigure__Param - rqt

Dynamic Reconfigure

Filter key:

Collapse all | Expand all

traffic_light

Refresh

/traffic_light

r1_h_h	0	255	1
r1_h_l	0	255	0
r1_s_h	0	255	255
r1_s_l	0	255	254
r1_v_h	0	255	255
r1_v_l	0	255	105
r2_h_h	0	255	255
r2_h_l	0	255	0
r2_s_h	0	255	255
r2_s_l	0	255	0
r2_v_h	0	255	255
r2_v_l	0	255	0
y1_h_h	0	255	35
y1_h_l	0	255	10
y1_s_h	0	255	255
y1_s_l	0	255	0

Color Detection: HoughCircles Part

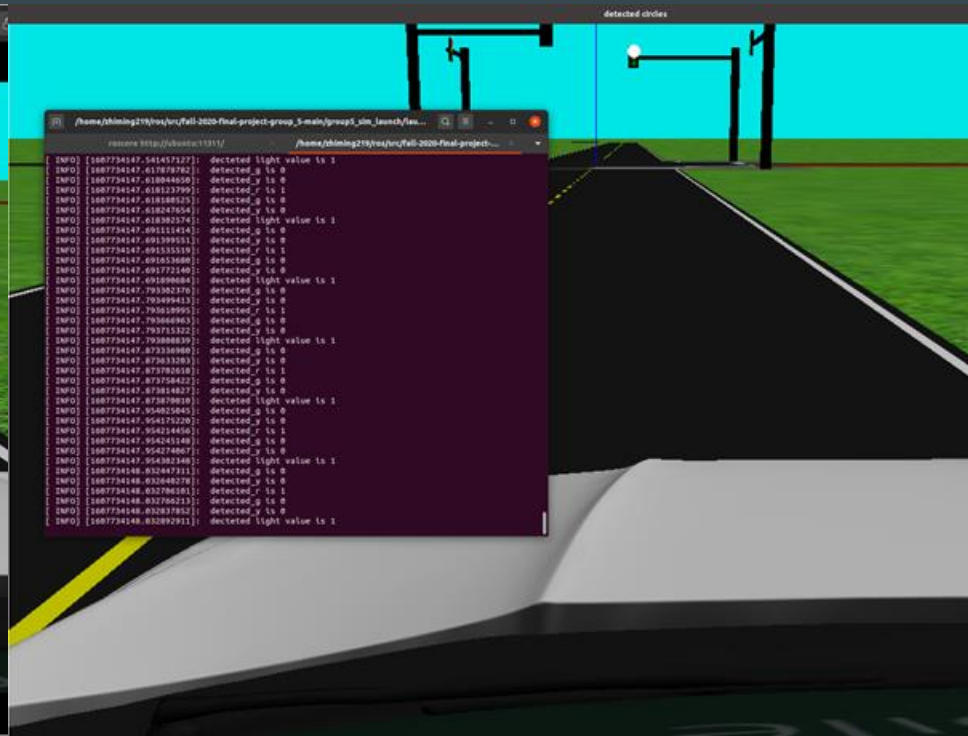
```
int detected_r = 0, detected_y = 0, detected_g = 0;
cv::HoughCircles(r_threshold, r_circle, cv::HOUGH_GRADIENT,1, 20, 50, 3, 3, 10);
cv::HoughCircles(r_threshold1, r_circle1, cv::HOUGH_GRADIENT,1, 10, 50, 5, 2, 30);

// find green circles
cv::HoughCircles(g_threshold, g_circle, cv::HOUGH_GRADIENT,1, 10, 50, 5, 2, 30);
// find yellow circles
cv::HoughCircles(y_threshold, y_circle, cv::HOUGH_GRADIENT,1, 10, 50, 5, 2, 30);
```

image	8-bit, single-channel, grayscale input image.
circles	output vector of found circles(cv.CV_32FC3 type). Each vector is encoded as a 3-element floating-point vector (x,y,radius) .
method	detection method(see **cv.HoughModes**). Currently, the only implemented method is HOUGH_GRADIENT
dp	inverse ratio of the accumulator resolution to the image resolution. For example, if dp = 1 , the accumulator has the same resolution as the input image. If dp = 2 , the accumulator has half as big width and height.
minDist	minimum distance between the centers of the detected circles. If the parameter is too small, multiple neighbor circles may be falsely detected in addition to a true one. If it is too large, some circles may be missed.
param1	first method-specific parameter. In case of HOUGH_GRADIENT , it is the higher threshold of the two passed to the Canny edge detector (the lower one is twice smaller).
param2	second method-specific parameter. In case of HOUGH_GRADIENT , it is the accumulator threshold for the circle centers at the detection stage. The smaller it is, the more false circles may be detected. Circles, corresponding to the larger accumulator values, will be returned first.
minRadius	minimum circle radius.
maxRadius	maximum circle radius.

Color Detection: Output 1 for RED

Original

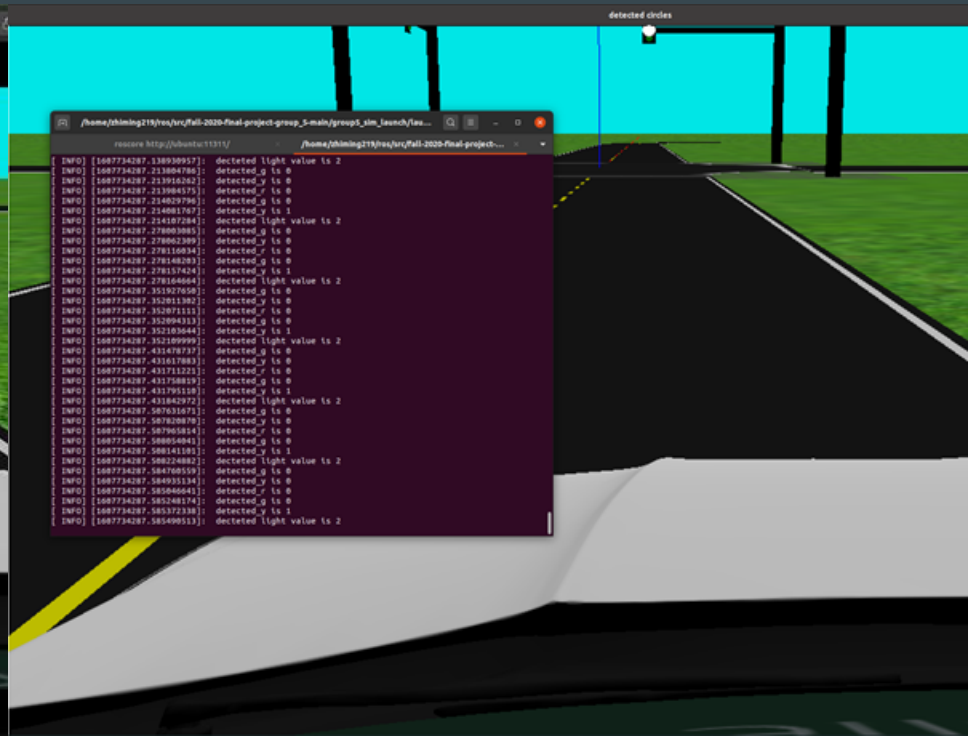
Identified



Color Detection: Output 2 for YELLOW

Original

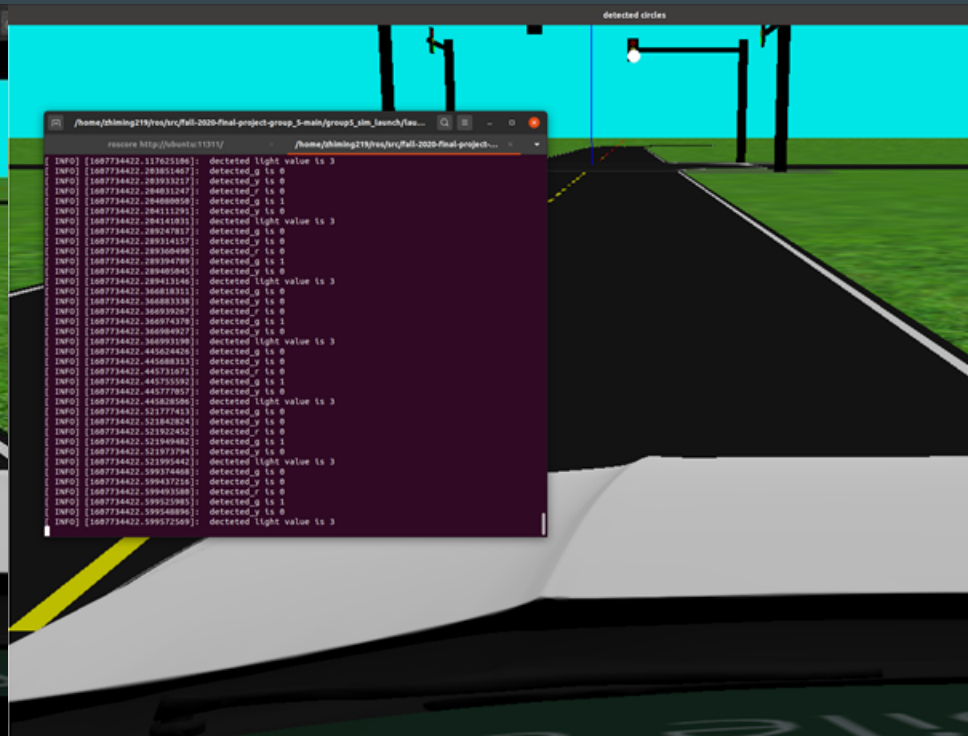
Identified



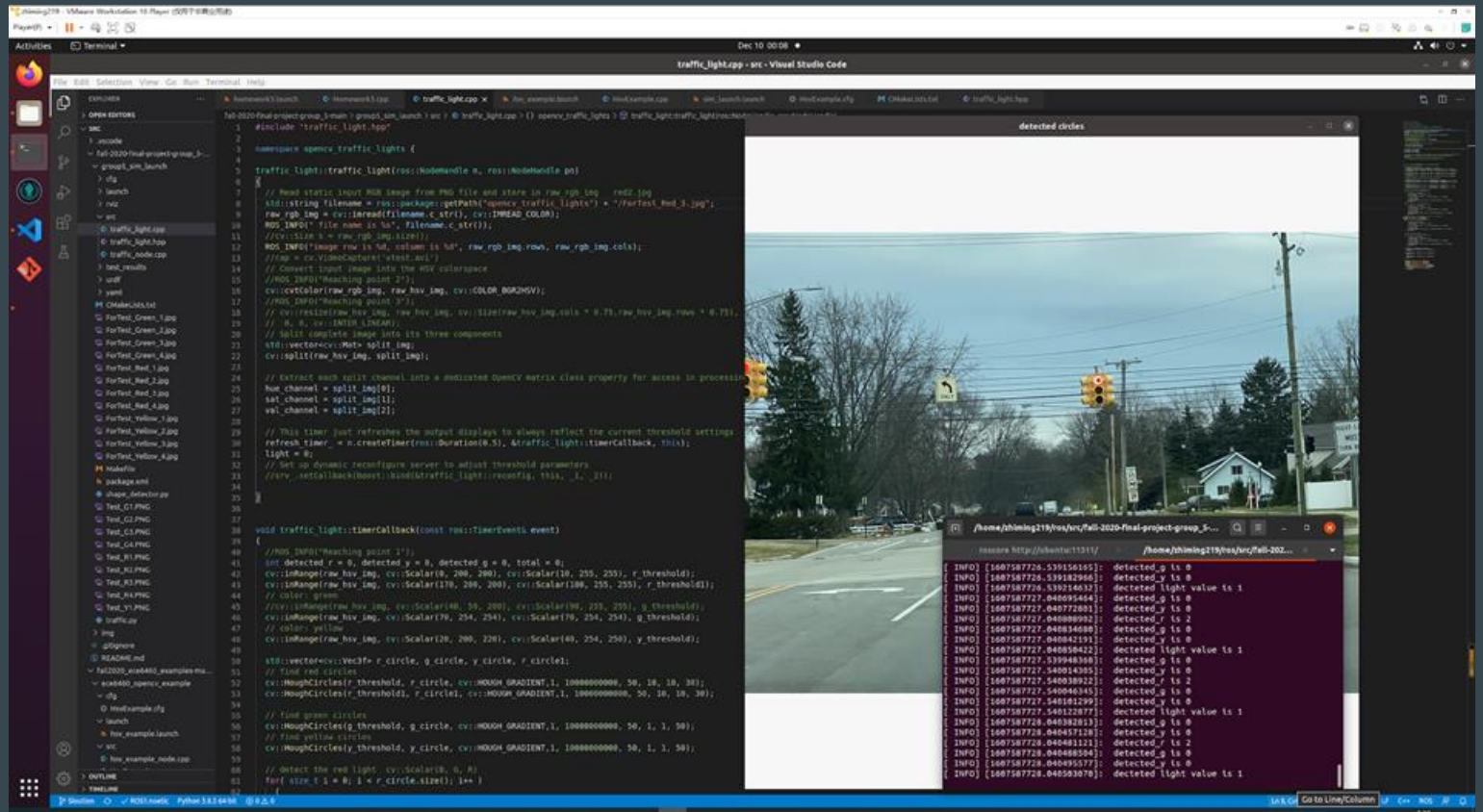
Color Detection: Output 3 for GREEN

Original

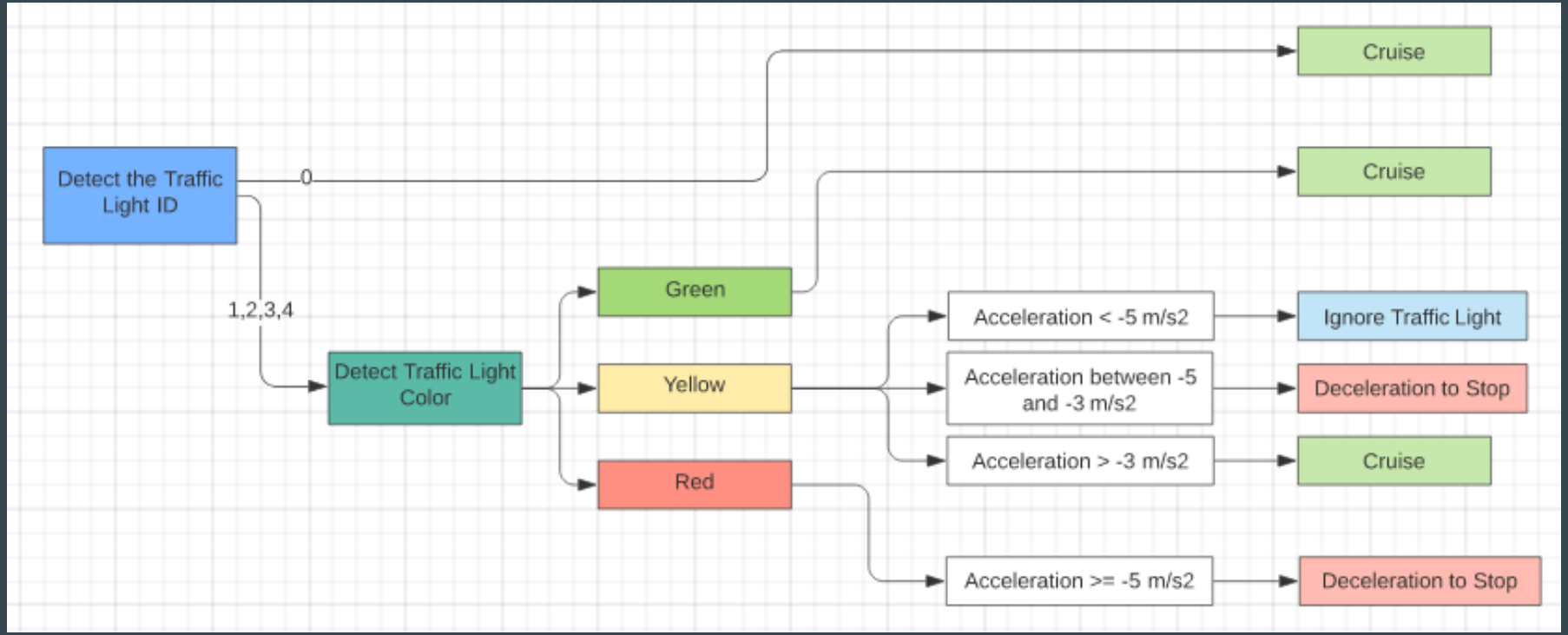
Identified



Color Detection: Real Application



Controls - Main Overview



Controls - Detect Traffic Light Structure

```
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
if (transform1.transform.translation.y < 1 && transform1.transform.translation.y >-1 && transform1.transform.translation.x>0 && std::abs(transform1.transform.rotation.z) >0.98)
    {
      Traffic_Light = 1;
      dist_traffic_light.data = transform1.transform.translation.x;
    }

    else
    {
      if (std::abs(transform3.transform.rotation.z) > 0.98 && transform3.transform.translation.y < 1 && transform3.transform.translation.y >-1 && transform3.transform.translation.x>0)
      {
        Traffic_Light = 3;
        dist_traffic_light.data = transform1.transform.translation.x;
      }
    else
      {
       Traffic_Light = 0;
      }
    }
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
```

Time based Control - Traffic Light Color Detection

```
////////////////////////////////////////////////////////////
    cycle_secs = secs - (i*35);

    if (cycle_secs<10)
    {
      Color = Green; //Green
    }
     if ((cycle_secs>=10)&&(cycle_secs<15))
    {
      Color = Yellow; //Yellow
    }
     if ((cycle_secs>=15)&&(cycle_secs<35))
    {
      Color = Red; //Red
    }
    if (cycle_secs>=35)
    {
      i = i + 1;
    }
////////////////////////////////////////////////////////////
```

Future Work

- Proper implementation of image processing node with control node
- Extend real world scenario capability: four way stop blinking red/yellow

Q&A