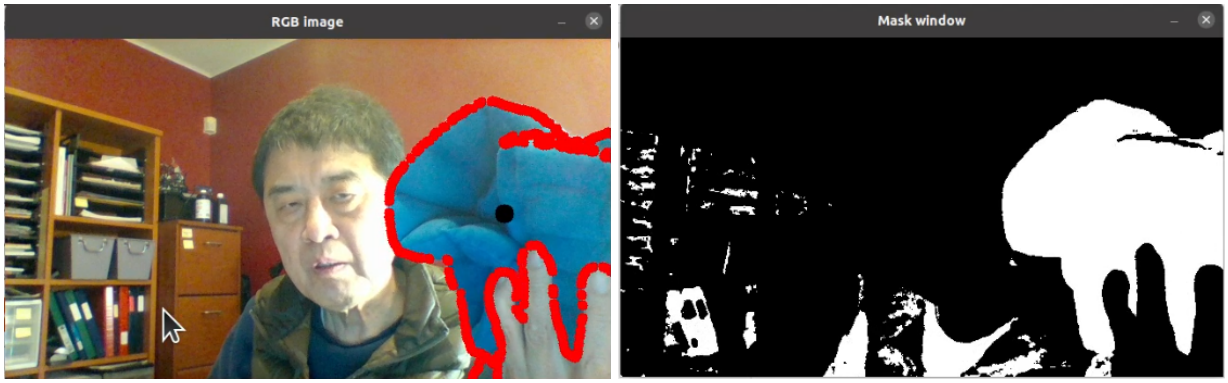
A Simple Lane Following Algorithm Using A Centroid of The Largest Blob

Chan-Jin Chung
cchung@ltu.edu
Department of Mathematics and Computer Science
Lawrence Technological University

1. Introduction

The idea of this simple lane following algorithm is first to find the centroid of the largest lane mark blob. It is assumed that only the right lane mark is processed using ROI (Region Of Interest) operation. Then the vehicle is steered so that the vehicle view center point gets closer to the left side of the centroid point to minimize the steering error. Next secretions show step by step instructions to write the code in Python for the ROS [1] environment using OpenCV[2] functions.

2. How to find the largest blue blob and draw a contour and a dot at the center of the contour (This was an assignment #11 during the workshop in 2022)

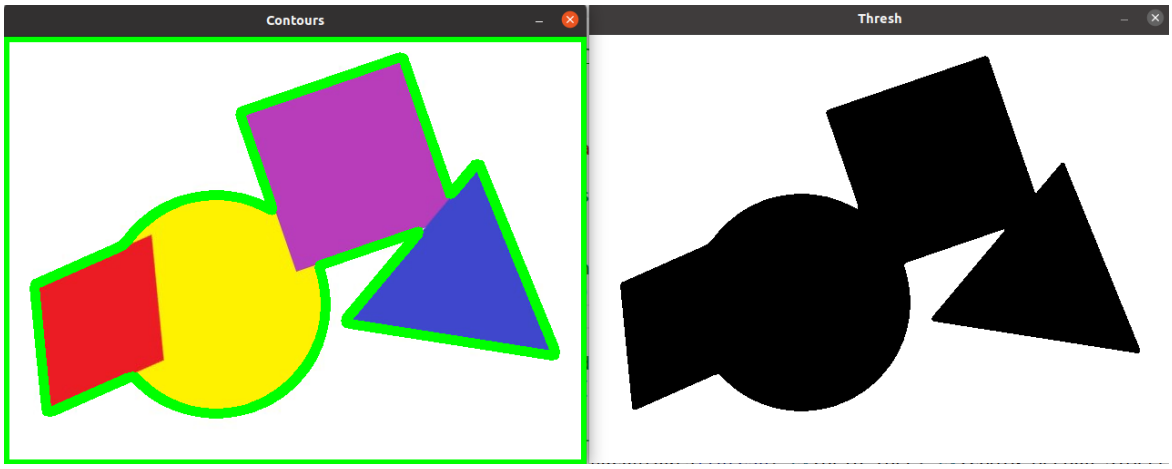


A contour can be explained simply as a curve joining all the continuous points (along the boundary), having the same color or intensity. The contours are a useful tool for shape analysis and object detection & recognition.

First watch this demo video and test the following 2 sample Python programs (not ROS) related to contours using OpenCV. It is recommended to use the directory, opencv_intro, used on Day 5 for OpenCV tests. Also read this medium.com article [3]:

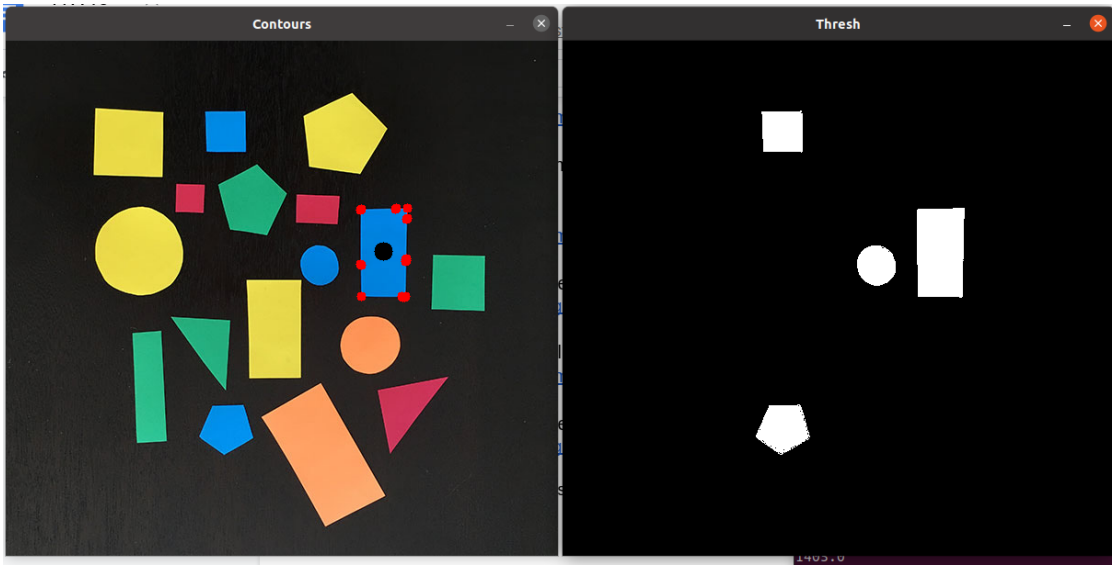
contour0.py

- This program needs shapes3.jpg file under “images” directory
- Blur will increase the number of dots for contours
- Note that the contour of the whole image rectangle was shown since `cv.RETR_TREE` was used: `cv.findContours(thresh, cv.RETR_TREE, cv.CHAIN_APPROX_SIMPLE)`



contour_color0.py

- This program needs shapes2.jpg file under “images” directory.
- Without blur(), fewer dots are used to draw contours
- This sample program tries to recognize the largest blue contour and draws a black dot at the centroid of it.
- `cv.RETR_LIST` is used instead of `cv.RETR_TREE`
- Expected result of contour_color.py is:

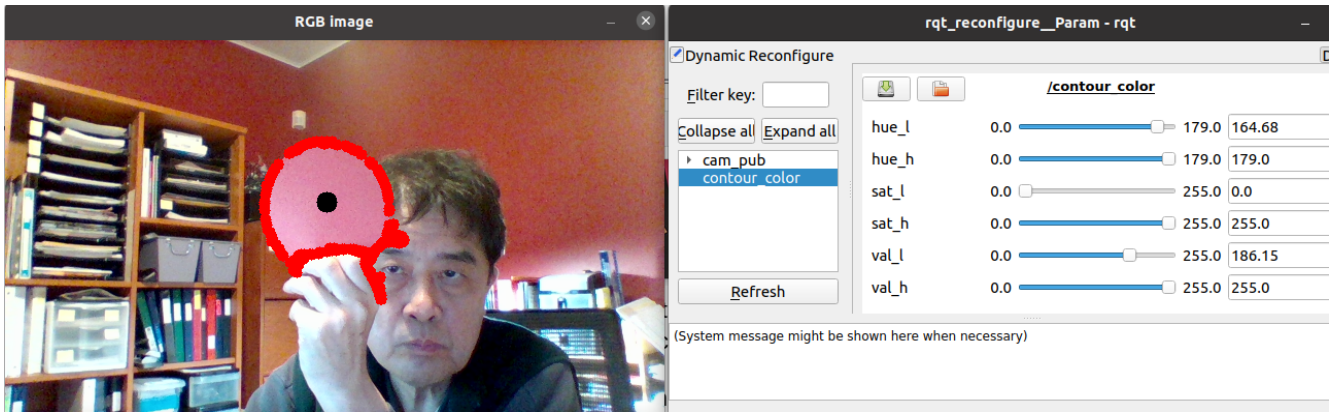


- As you see above, the largest blue shape has red points to represent the contour and a black dot at the center point of the contour.

Use the existing “color_pct_pkg” and reuse the same .cfg file for HVS sliders. contour_color.launch file must include (1) rqt_reconfigure (2) camera_publisher.launch (3) contour_color.py

Do a Google search to find sample code for finding a centroid of the largest contour. Here is an incomplete contour_color.py file. Handle divide by zero exceptions caused by unclosed contours.

Demonstrate your program at least 2 different colors of your choice. Here is another example of detecting red.



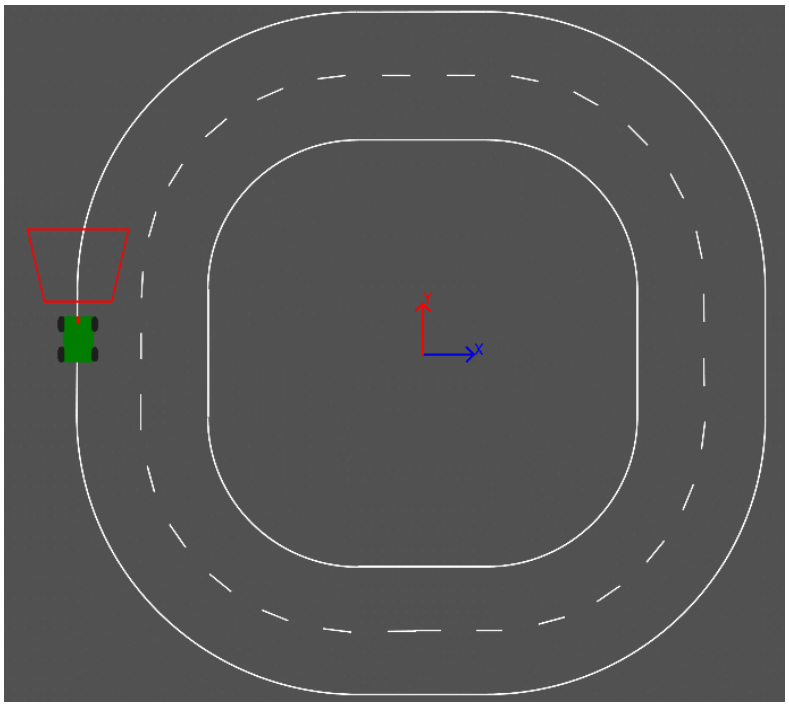
3. Line following using "simple_sim" (This was a project during the workshop in 2022)

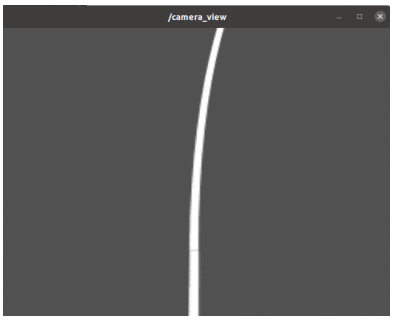
Note: Before starting this project, please *review* the "**stop_at_crosswalk**" project.

Download `simple_sim_circleroad.zip` and unzip. Place the "simple_sim_circleroad" package in your "<workspace>/src" directory, catkin_make, and test the following roslaunch command:

```
$ roslaunch simple_sim_circleroad circleroad.launch
```

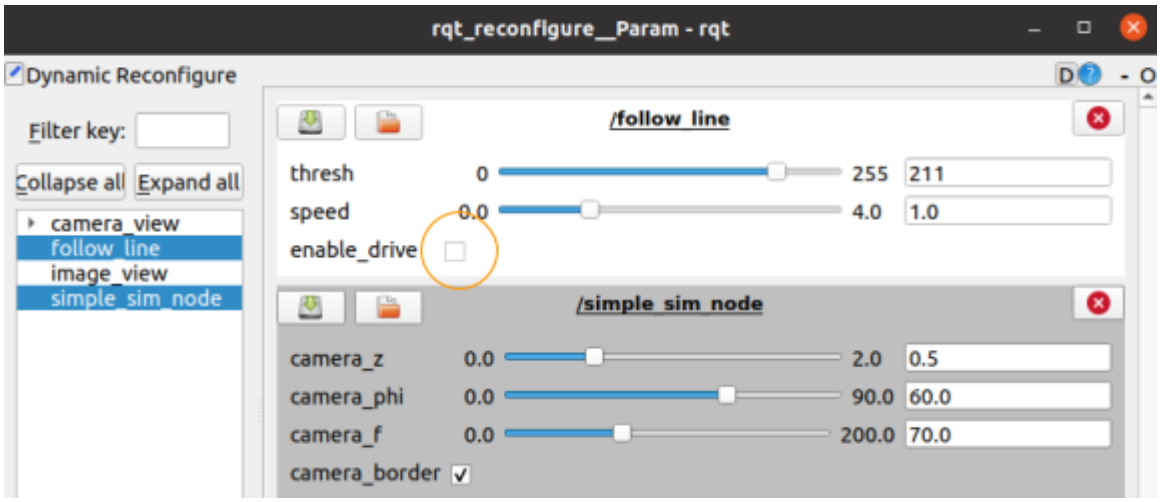
Then you will see the following circle road and the camera view from the green Ackerman vehicle's camera. The camera view border is shown as an isosceles trapezoid in red below left.





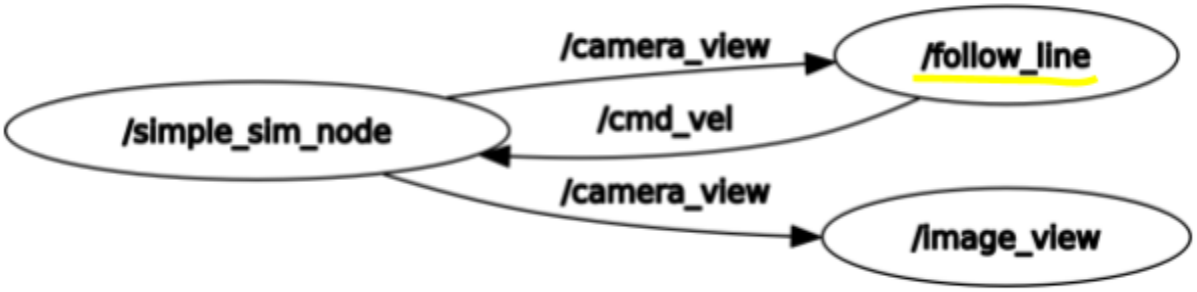
Project requirements: create “follow_lane_pkg” package with a node “follow_line.py” to follow the outer line indefinitely when the “enable_drive” checkbox shown below in orange circle is checked.

Here is a sample video. There is no starter code for this project. However, it is similar to the “stop_at_crosswalk” project.



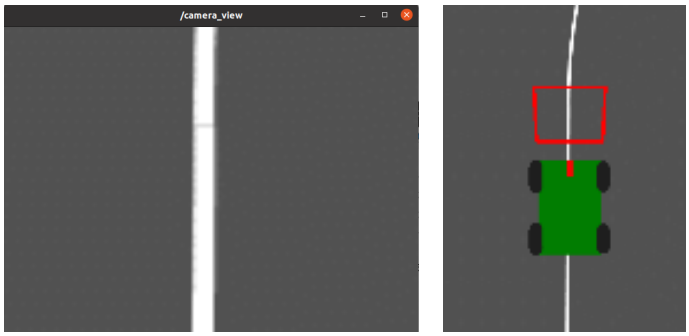
The launch file needs to include circleroad.launch of “simple_sim_circleroad”.

```
<launch>
   <include file="$(find simple_sim_circleroad)/launch/circleroad.launch">
   </include>

   <node name="follow_line" pkg="follow_lane_pkg" type="follow_line.py" required="true" output="screen">
         <param name="imgtopic_name" type="str" value="camera_view" />
   </node>

</launch>
```



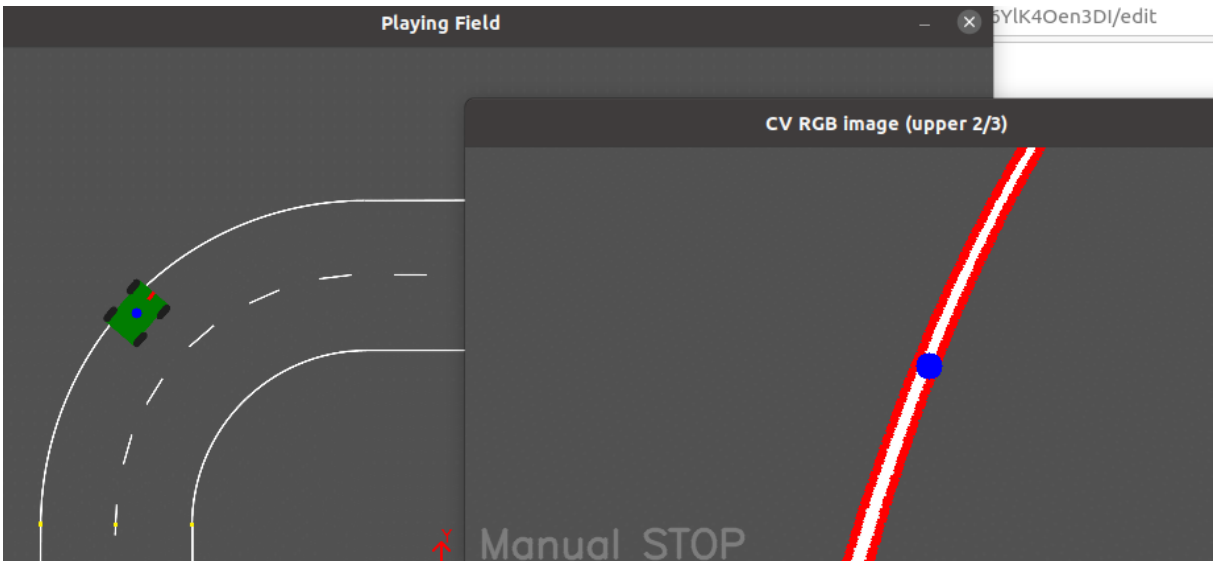
Hints:

- You may adjust camera parameters. For example, camera_f can make the line thicker as shown below.



- Find a centroid of the largest white contour.
- Rudimentary line following algorithm using the centroid as follows. The figure after the algorithm box demonstrates a window showing a contour with a centroid point in blue color.

```
linear.x = speed value from the GUI
mid view point = cols / 2
If centroid x point is greater than mid view point + torenance value:
    angular.z is negative Kz (or proportional to the diff between mid view p and centroid x)
    Publish Twist message
elif: centroid x point is less than mid view point - torenance value:
    angular.z is positive Kz (or proportional to the diff between mid view p and centroid x)
    Publish Twist message
else:
    angular.z is zero
    Publish Twist message
```



Problems of the above algorithm? What if the line is lost?

4. Lane following using the line following algorithm in Section 3.

A simple idea is to change the target point to follow from “centroid” to left side of the right lane. The actual number of pixels to subtract from the centroid x can be calculated proportionally from the width of the road lane. The modified algorithm can be:

```
linear.x = speed value from the GUI
mid viewpoint = cols / 2
lane_center = centroid - k*lane_width/2
If lane_center is greater than mid viewpoint + torenance value:
    angular.z is negative Kz (or proportional to the diff between mid view p and centroid x)
    Publish Twist message
elif: lane_center is less than mid viewpoint - torenance value:
    angular.z is positive Kz (or proportional to the diff between mid view p and centroid x)
    Publish Twist message
else:
    angular.z is zero
    Publish Twist message
```

Another simpler algorithm to minimize the steering error (the difference between the target point and the current mid viewpoint) can be:

```
linear.x = speed value from the GUI
mid_viewpoint = cols / 2
lane_center = centroid - k*lane_width/2
steer_error = mid_viewpoint - lane_center
angular.z = 0.01 * steer_error
publish Twist message
```

References

[1] Stanford Artificial Intelligence Laboratory et al. (2018). *Robotic Operating System*. Retrieved from https://www.ros.org

[2] https://opencv.org/

[3] Maximinusjoshus, Draw Contours on an Image using OpenCV, https://medium.com/featurepreneur/draw-contours-on-an-image-using-opencv-186b67f87c92