3. Astra color tracking

3.1. Introduction

The Astra color tracking of the Yahboom mobile robot can recognize multiple colors at any time, and independently stores the currently recognized color, controls the car to follow the detected and recognized color, and maintains a certain distance from the object.

The color tracking of the Yahboom mobile robot can also realize the function of real-time control of HSV. By adjusting the high and low thresholds of HSV, the interfering colors are filtered out, so that the squares can be ideally identified in complex environments. If the color selection effect is not good, Ideally, at this time, the car needs to be moved to different environments and calibrated so that it can recognize the colors we need in complex environments.

3.1.1, HSV introduction

HSV (Hue, Saturation, Value) is a color space created by A. R. Smith in 1978 based on the intuitive characteristics of color, also known as the Hexcone Model.

The parameters of color in this model are: hue (H), saturation (S), and lightness (V).

H: 0 — 180

S: 0 — 255

V: 0 — 255

Here some reds are classified into the purple range:

	black	grey	white	red		orange	yellow	green	light blue	blue	Purple
H_min	0	0	0	0	156	11	26	35	78	100	125
H_max	180	180	180	10	180	25	34	77	99	124	155
S_min	0	0	0	43		43	43	43	43	43	43
S_max	255	43	30	255		255	255	255	255	255	255
V_min	0	46	221	46		46	46	46	46	46	46
V_max	46	220	255	255		255	255	255	255	255	255

3.1.2, HSV hexagonal pyramid

- **Hue H**

Represents color information, that is, the position of the spectral color. This parameter is represented by an angle, with a value ranging from 0° to 360°, starting from red and counting in counterclockwise direction. Red is 0°, green is 120°, and blue is 240°. Their complementary colors are: yellow is 60°, cyan is 180°, and purple is 300°.

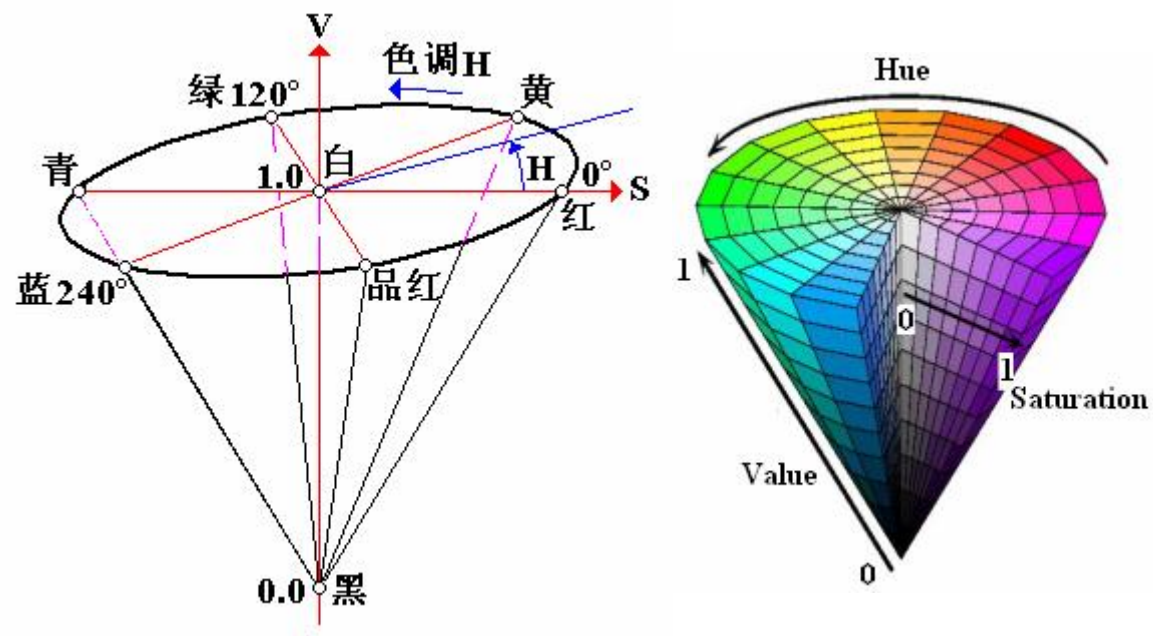
- **Saturation S**

Saturation S is expressed as the ratio between the purity of the selected color and the maximum purity of that color. When S=0, there is only grayscale. 120 degrees apart. Complimentary colors are 180 degrees apart. A color can be thought of as the result of mixing a certain spectral color with white. The greater the proportion of spectral colors, the closer the color is to spectral colors, and the higher the saturation of the color. The saturation is high and the color is deep and vivid. The white light component of the spectral color is 0, and the saturation reaches the highest level. Usually the value range is 0% ~ 100%. The larger the value, the more saturated the color.

- **LightnessV**

Brightness represents the brightness of a color. For light source color, the brightness value is related to the brightness of the luminous body; for object color, this value is related to the transmittance or reflectance of the object. Usually the value range is 0% (black) to 100% (white). One thing to note: there is no direct relationship between it and light intensity.

The three-dimensional representation of the HSV model evolves from the RGB cube. If you imagine looking from the white vertices of the RGB along the diagonal of the cube to the black vertices, you can see the hexagonal shape of the cube. The hexagonal borders represent color, the horizontal axis represents purity, and lightness is measured along the vertical axis.



3.2. Operation steps

Note: [R2] on the remote control handle has the [pause/start] function for all gameplays.

3.2.1. Start

```
#Raspberry Pi 5 master needs to enter docker first, please perform this step
#If running the script into docker fails, please refer to ROS/07, Docker tutorial
~/run_docker.sh
```

Note: When the image is displayed, press the [q] key to exit.

Two startup methods, choose one, the demonstration case is method 2

method one

robot side

```
roslaunch yahboomcar_astra colorTracker.launch VideoSwitch:=true
```

Method 2

Can be controlled remotely for easy operation.

robot side

```
roslaunch yahboomcar_astra colorTracker.launch VideoSwitch:=false
```

virtual machine

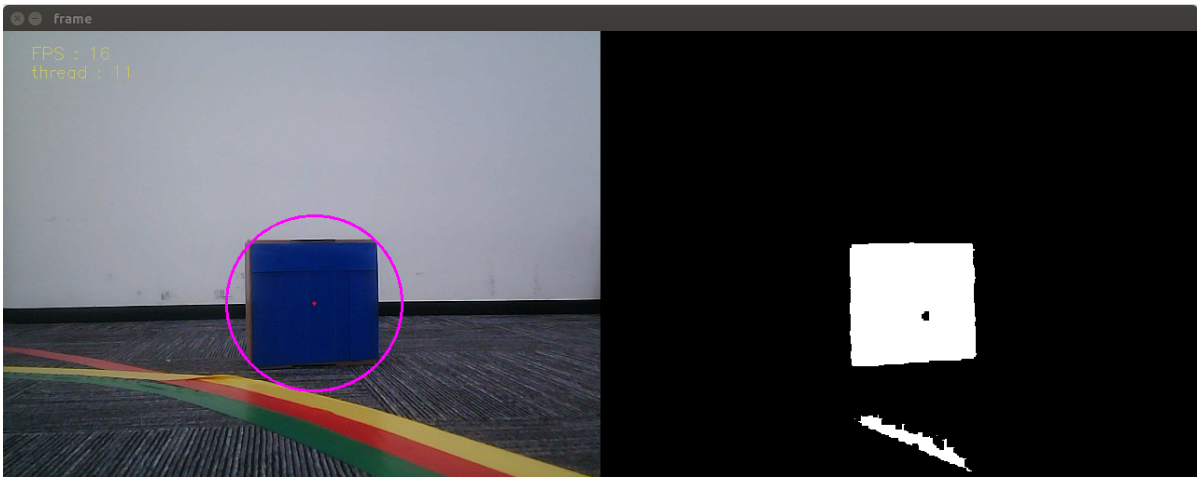
```
roslaunch yahboomcar_astra colorHSV.launch
```

- VideoSwitch parameter: whether to use the camera function package to start.

Set parameters according to needs, or modify the launch file directly, so there is no need to attach parameters when starting.

3.2.2. Identification

After startup, the system defaults to [Target Detection Mode], as shown below:



Keyboard key control:

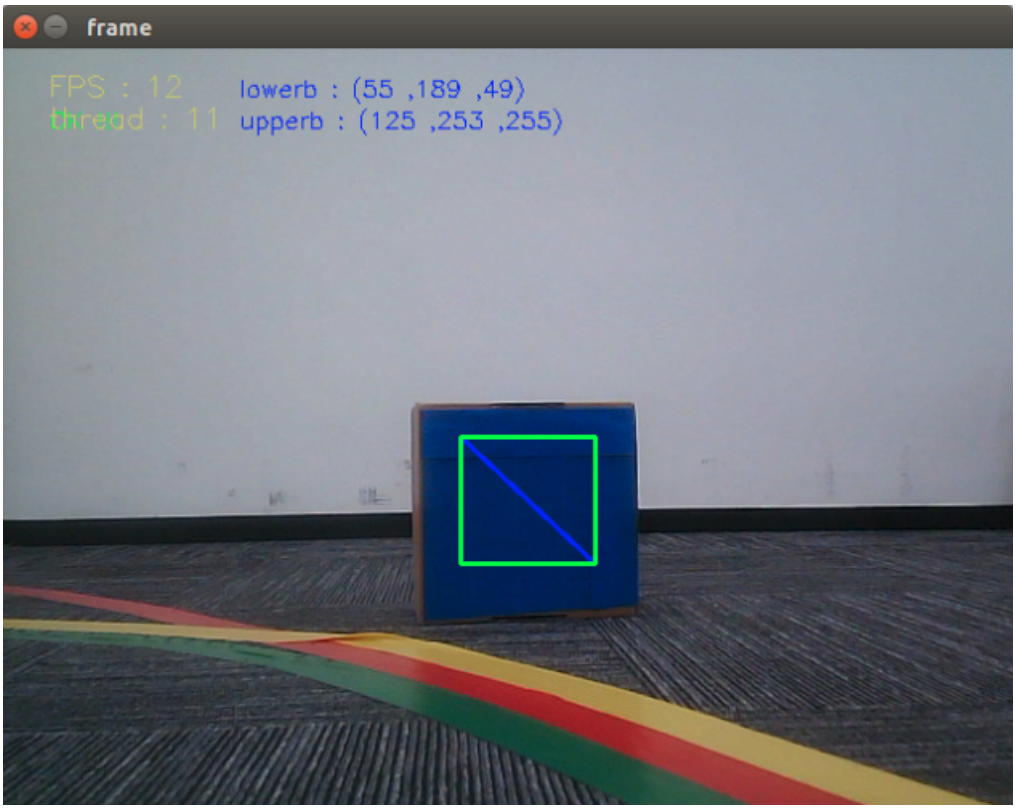
[r]: Color selection mode, you can use the mouse to select the area of the color to be recognized (cannot exceed the area range).

【i】: Target detection mode. The color image on the left (Color) and the binary image on the right (Binary).

[q]: Exit the program.

[Spacebar]: After identifying that there is no problem, click [Spacebar] on the keyboard to execute the color following program.

In the color selection mode, use the mouse to select the location of the colored object, as shown in the figure below, and release it to start recognition.

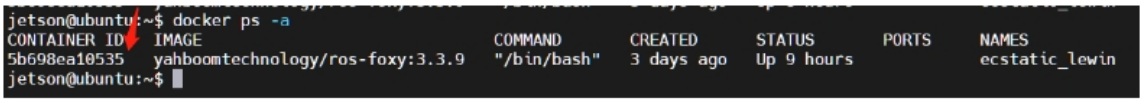


3.2.3. Color calibration

<PI5 needs to open another terminal to enter the same docker container

1. In the above steps, a docker container has been opened. You can open another terminal on the host (car) to view:

```
docker ps -a
```



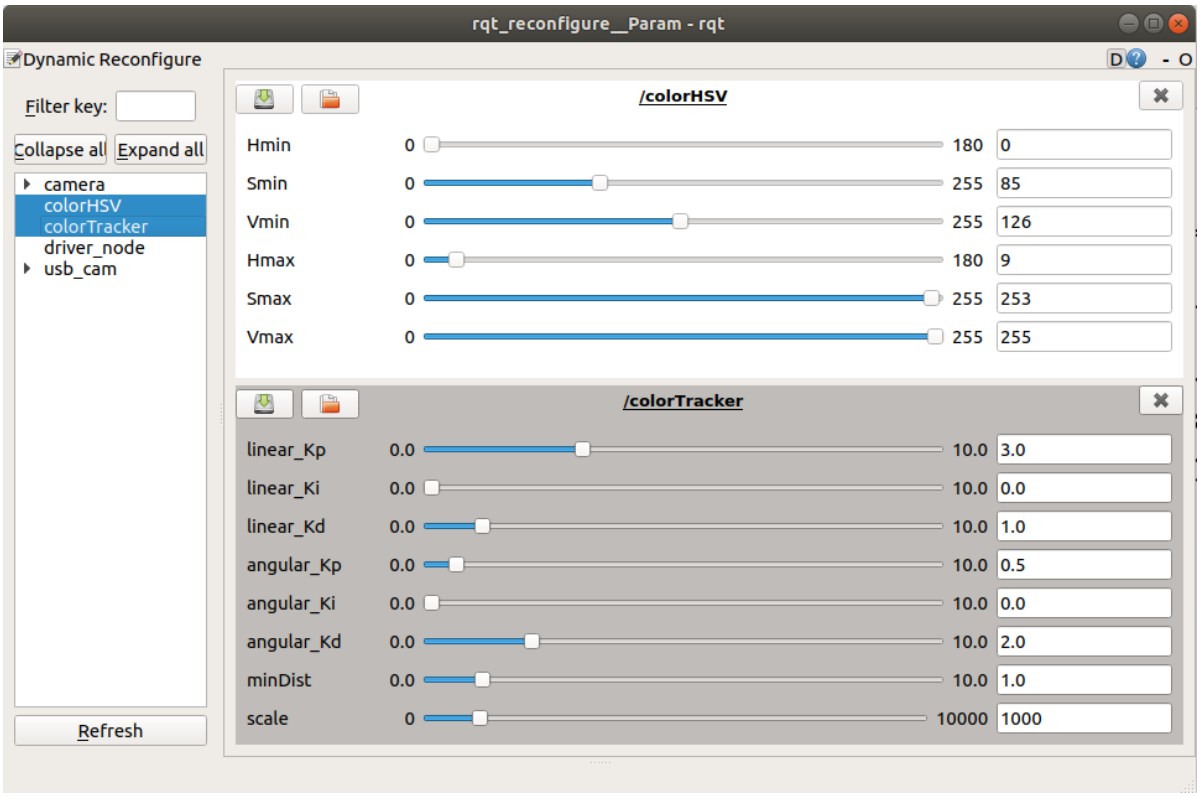
2. Now enter the docker container in the newly opened terminal:

```
docker exec -it 5b698ea10535 /bin/bash
```

After successfully entering the container, you can open countless terminals to enter the container.

Dynamic parameter tuning

```
rosrun rqt_reconfigure rqt_reconfigure
```



Select the [color_HSV] node and [color_Tracker] node. Generally, you only need to adjust [Hmin], [Smin], [Vmin], and [Hmax]. These four parameters can be easily identified. The slide bar is always in a dragging state and data will not be transferred to the system until it is released; you can also select a row and then slide the mouse wheel.

Parameter analysis:

[linear_Kp], [linear_Ki], [linear_Kd]: PID control of linear speed during car following.

[angular_Kp], [angular_Ki], [angular_Kd]: PID control of angular velocity during car following.

[minDist]: Follow the distance and keep this distance.

[scale]: PID scaling.

- Parameter modification

When the parameters are adjusted to the optimal state, the corresponding parameters are modified into the file, and no adjustment is required when using again.

According to the optimal parameters of the [rqt_reconfigure] debugging tool, enter the [scripts] folder of the [yahboomcar_astra] function package and modify the parameters corresponding to the [colorTracker.py] file, as shown below

```
class color_Tracker:
    def __init__(self):
        rospy.on_shutdown(self.cleanup)
        self.bridge = CvBridge()
        self.minDist = 1.0
        ... ...
        self.linear_PID = (3.0, 0.0, 1.0)
        self.angular_PID = (0.5, 0.0, 2.0)
        self.scale = 1000
        self.PID_init()
```

[rqt_reconfigure] Modification of the initial value of the debugging tool

```
gen = ParameterGenerator()
gen.add("linear_Kp", double_t, 0, "Kp in PID", 3.0, 0, 10.0)
gen.add("linear_Ki", double_t, 0, "Ki in PID", 0.0, 0, 10.0)
gen.add("linear_Kd", double_t, 0, "Kd in PID", 1.0, 0, 10.0)
gen.add("angular_Kp", double_t, 0, "Kp in PID", 0.5, 0, 10.0)
gen.add("angular_Ki", double_t, 0, "Ki in PID", 0.0, 0, 10.0)
gen.add("angular_Kd", double_t, 0, "Kd in PID", 2.0, 0, 10.0)
gen.add("minDist", double_t, 0, "minDist", 1.0, 0, 10)
gen.add("scale", int_t, 0, "scale", 1000, 0, 10000)
exit(gen.generate(PACKAGE, "colorTracker", "ColorTrackerPID"))
```

Enter the [cfg] folder of the [yahboomcar_astra] function package and modify the initial values of the parameters corresponding to the [ColorTrackerPID.cfg] file.

```
gen.add("linear_Kp", double_t, 0, "Kp in PID", 3.0, 0, 10.0)
```

Take the above article as an example to analyze

Parameters	Analysis	Corresponding parameters
name	name of the parameter	"linear_Kp"
type	parameter data type	double_t
level	a bitmask passed to the callback	0
description	A description parameter	"Kp in PID"
default	Initial value for node startup	3.0
min	parameter minimum value	0
max	parameter maximum value	10.0

Note: After modification, you must recompile and update the environment to be effective.

```
cd ~/yahboomcar_ws
catkin_make
source devel/setup.bash
```

3.3. Program analysis

colorTracker.launch file

```
<launch>
    <arg name="VideoSwitch" default="false"/>
    <!-- Handle control node Handle control node-->
    <include file="$(find yahboomcar_ctrl)/launch/yahboom_joy.launch"/>
    <!-- depth camera node astra depth node-->
    <include file="$(find astra_camera)/launch/astra.launch"/>
    <!-- robot drive node-->
    <include file="$(find yahboomcar_bringup)/launch/yahboomcar.launch"/>
```

```
    <!-- usb_cam color node-->
    <include file="$(find usb_cam)/launch/usb_cam-test.launch" unless="$(arg
VideoSwitch)"/>
    <node pkg="yahboomcar_astra" type="colorTracker.py" name="colorTracker"
required="true" output="screen"/>
    <node pkg="yahboomcar_astra" type="colorHSV.py" name="colorHSV"
required="true" output="screen" if="$(arg VideoSwitch)">
        <param name="VideoSwitch" type="bool" value="$(arg VideoSwitch)"/>
    </node>
</launch>
```

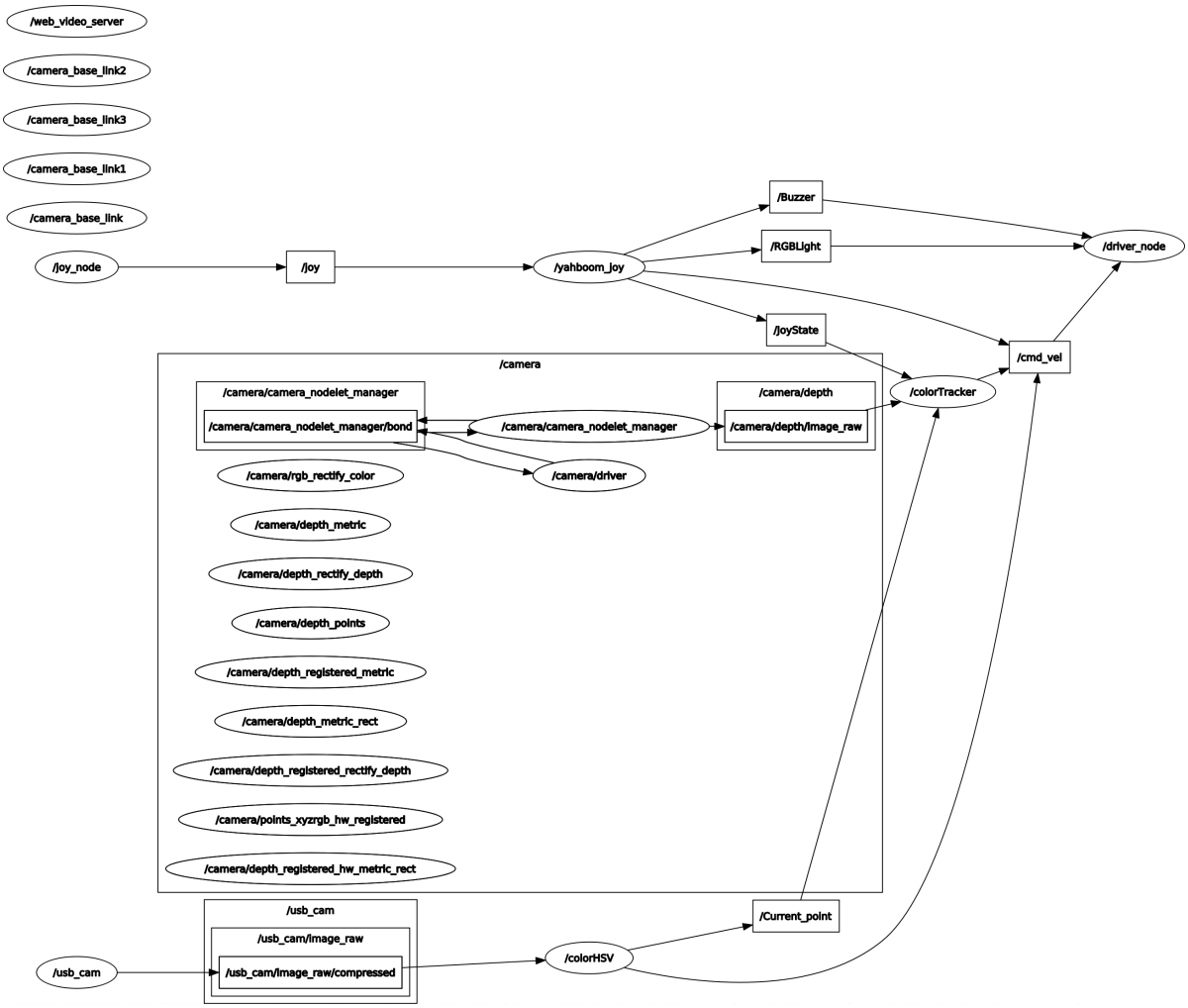
launch file analysis

If the [VideoSwitch] parameter is true, the monocular color camera will not be started and the [colorHSV] node will be started. The method of obtaining color images is directly implemented by the [colorHSV] node using [video*].

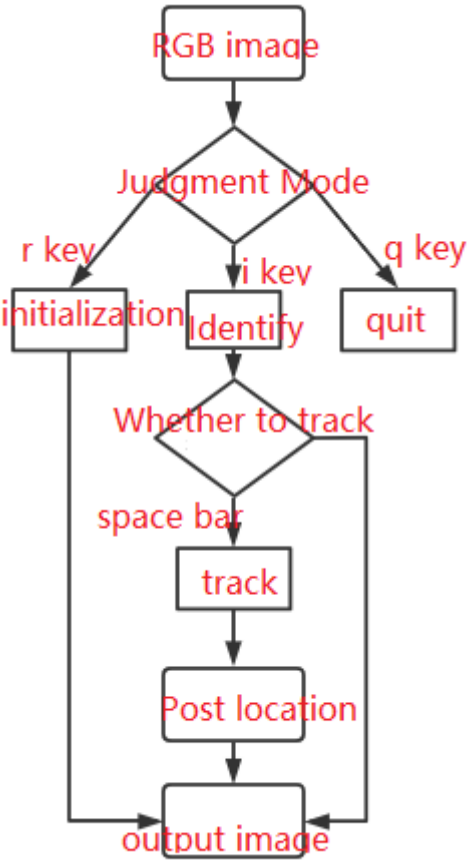
If the [VideoSwitch] parameter is false, the monocular color camera is started and the [colorHSV] node is not started. Support web page monitoring. At this time, the [colorHSV] node needs to be started separately.

- Node view

```
rqt_graph
```

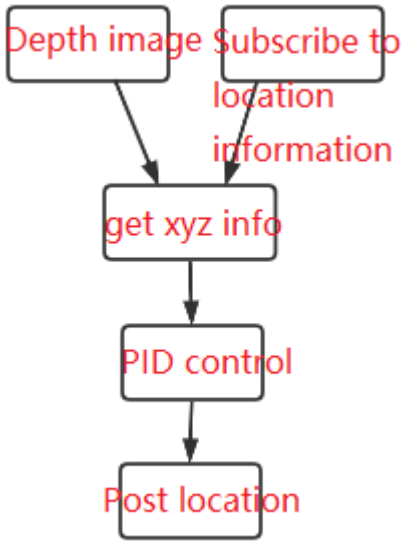


Node【color_HSV】



- Subscribe to color images
- Publish the location of the recognized object in the image
- Issue control instructions (only stop commands are issued when no color is recognized in the picture)

Node【color_Tracker】



- Subscribe to depth images
- Subscribe to handle control information
- Subscribe to the location information of the recognized object in the image
- Issue car follow control instructions