3 Astra Color Tracking

3.1 Introduction

The Astra color tracking of the Yahboom mobile robot has the ability to recognize multiple colors at any time, and autonomously store the currently recognized color, control the car to follow the detected color, and maintain a certain distance from the object.

The color tracking of the Yahboom mobile robot can also realize the function of real-time HSV control. By adjusting the high and low thresholds of HSV, the interfering colors are filtered out, so that the squares can be recognized ideally in complex environments. Ideally, at this time, the car needs to be moved to a different environment for calibration, so that the color we need can be recognized in a complex environment.

3.1.1 Introduction to HSV

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

The parameters of the color in this model are: Hue(H), Saturation(S), Lightness(V).

H: 0 --180

S: 0 -- 255

V: 0 -- 255

Part of the red is classified as purple here:

	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 at which it is located. This parameter is represented by an angle, and the value range is 0° ~ 360°. It is calculated counterclockwise from red, red is 0°, green is 120°, and blue is 240°. Their complementary colors are: yellow at 60°, cyan at 180°, and purple at 300°.

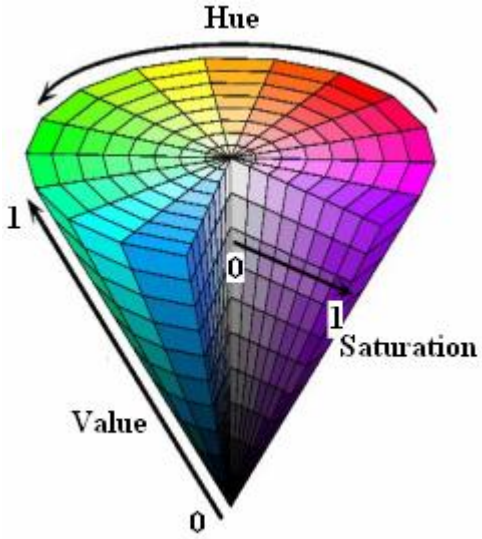
- **Saturation S**

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. Complementary colors are 180 degrees apart. A color that can be seen as the result of a spectral color mixed with white. The greater the proportion of spectral colors, the higher the degree of color close to spectral colors, the higher the color saturation. The saturation is high, and the color is deep and vivid. The white light component of the spectral color is 0, and the saturation is the highest. Usually the value ranges from 0% to 100%, the larger the value, the more saturated the color.

- **Brightness V**

Lightness indicates the brightness of the color. For light source color, the lightness value is related to the brightness of the illuminant; for object color, this value is related to the transmittance or reflectance of the object. Usually the value ranges from 0%(black) to 100%(white). One thing to note: there is no direct connection between it and light intensity.

The 3D representation of the HSV model evolves from an RGB cube. Imagine looking from the white vertices to the black vertices along the diagonal of the cube, and 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] of the remote controller has the function of [pause/on] for all gameplays.

3.2.1 Start

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

Two startup methods, choose one, the demo case is the second method

method one

robot side

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

Method 2

It can be controlled remotely for easy operation.

robot side

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

virtual machine

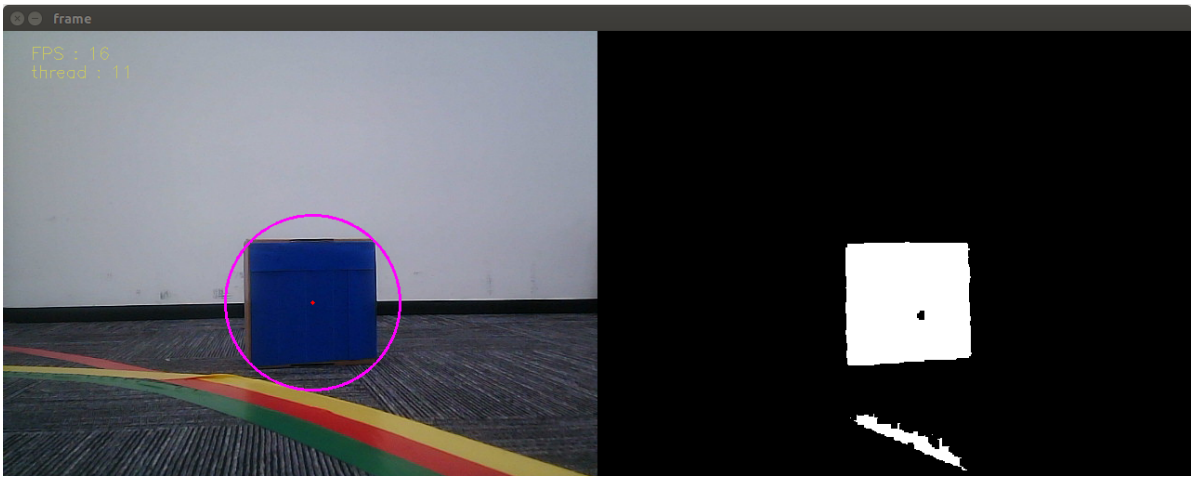
```
roslaunch yahboomcar_astra colorHSV.launch
```

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

Set parameters according to your needs, you can also directly modify the launch file, and you don't 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:

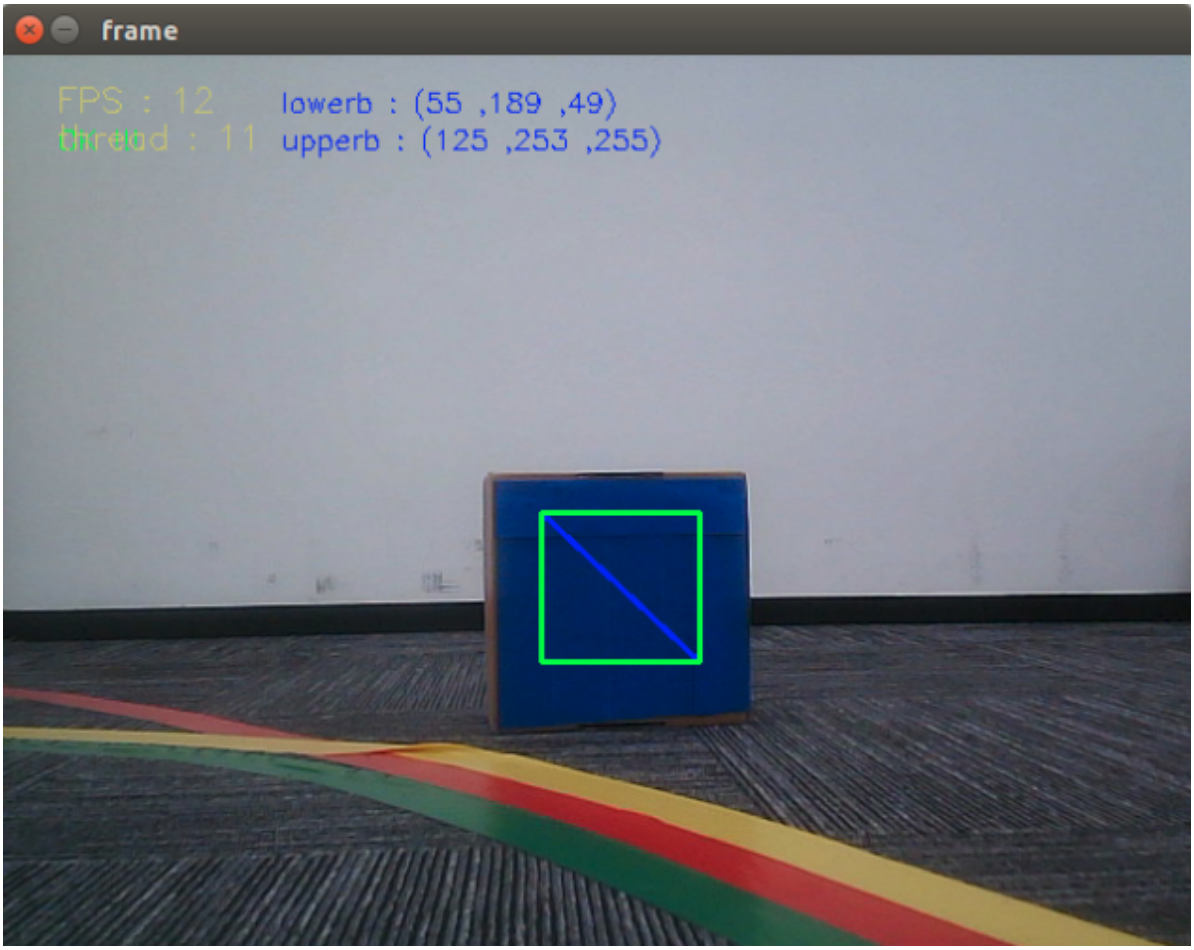
【r】: Color selection mode, you can use the mouse to select the area of the color to be recognized(cannot exceed the area range).

[i]: Target detection mode. Color image on the left(Color), and binary image on the right(Binary).

【q】: Exit the program.

[Spacebar]: After identifying no problem, click [Spacebar] on the keyboard to execute the color follow program.

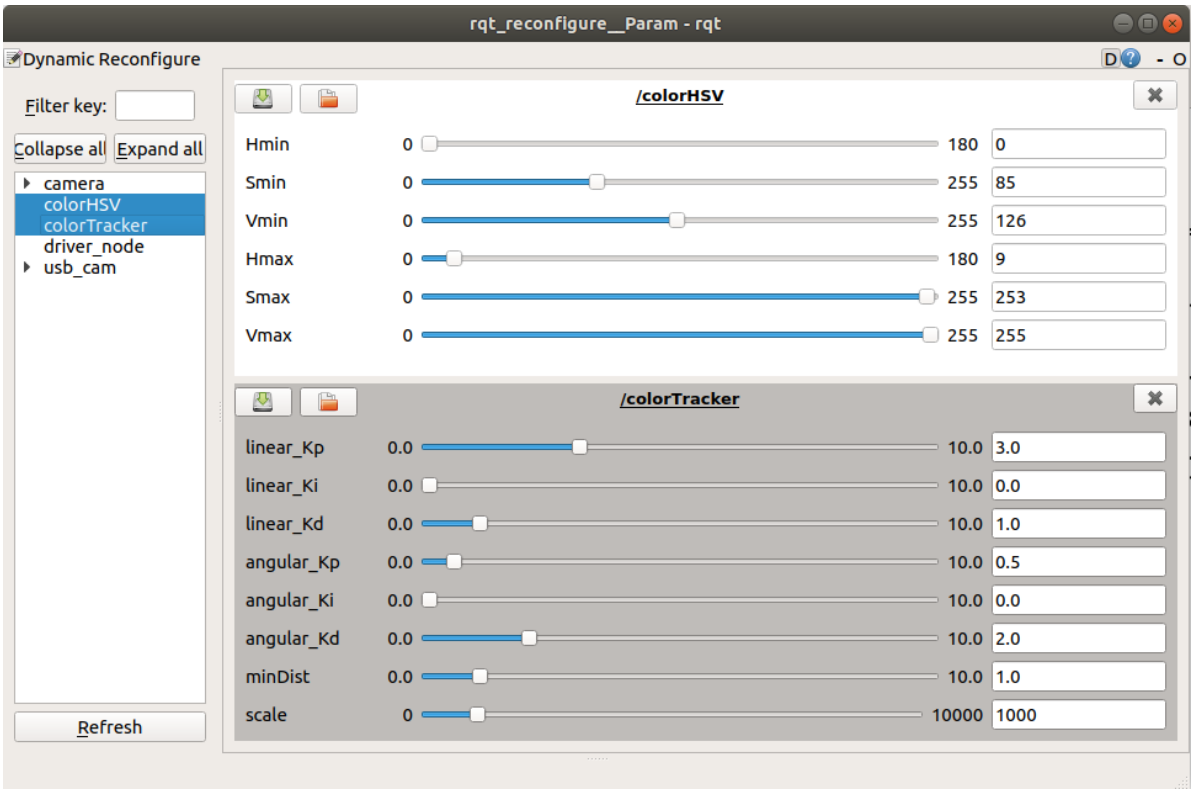
In the color selection mode, use the mouse to select the location of the object with color, as shown in the figure below, release it to start recognition.



3.2.3 Color calibration

Dynamic parameter tuning

```
rosrun rqt_reconfigure rqt_reconfigure
```



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

Parameter parsing:

[linear_Kp], [linear_Ki], [linear_Kd]: Linear speed PID control during the following process of the car.

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

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

【scale】: PID scaling.

- parameter modification

When the parameters are adjusted to the optimal state, modify the corresponding parameters to the file, and there is no need to adjust them when using them 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] Modify 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)
```

Analysis of the above one as an example

parameter	Parse	Corresponding parameters
name	the 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	0
max	parameter maximum	10.0

Note: After modification, the update environment must be recompiled 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" />
    <!--  control  Handle  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 >
```

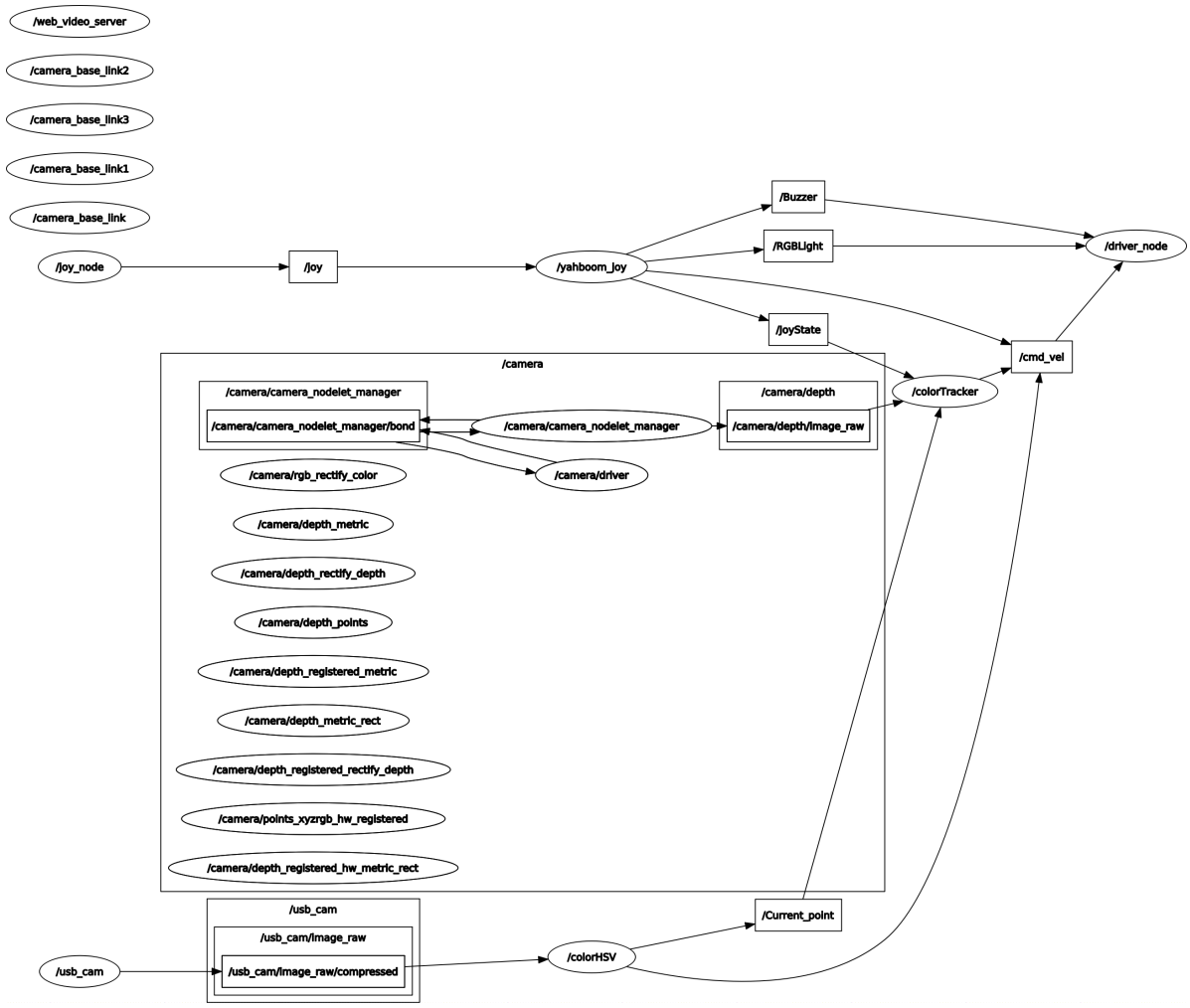
launch file analysis

If the [VideoSwitch] parameter is true, the monocular color camera will not be activated, and the [colorHSV] node will be activated. The way to obtain color images is directly implemented by the [colorHSV] node using [video*].

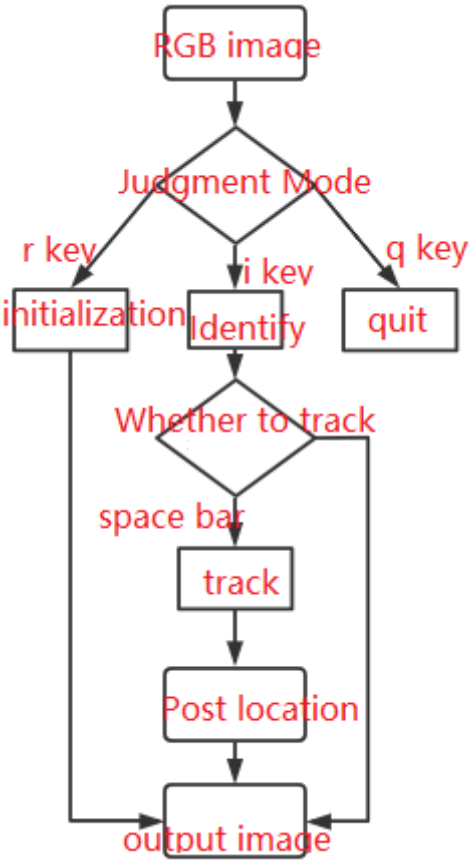
If the [VideoSwitch] parameter is false, the monocular color camera is activated, and the [colorHSV] node is not activated. Support web page monitoring. At this point, you need to start the [colorHSV] node separately.

- Node view

```
rqt_graph
```

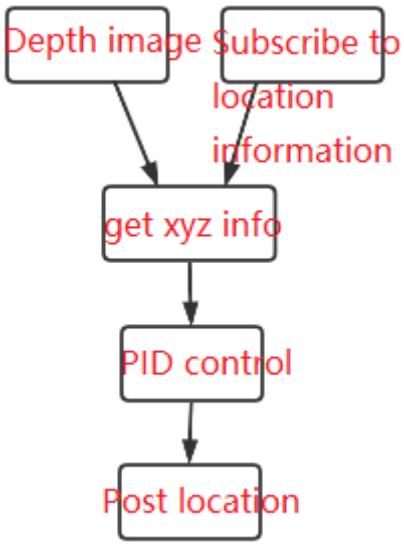


Node [color_HSV]



- Subscribe to Color Images
- Publish the position of the recognized object in the image
- Issue control commands(only issue stop commands when no color is recognized in the picture)

Node [color_Tracker]



- Subscribe to depth images
- Subscribe to handle control information
- Subscribe to the position information of the identified object in the image
- Issue the car following control command