

Lesson 20 Color Detection

20.1 Overview

Color detection is a crucial aspect in robotics and computer vision applications. It enables robots to identify and interact with their environment based on the colors of objects. By leveraging color detection algorithms, the Adept RaspClaws Robot can perform tasks such as object recognition, sorting, and navigation. This lesson will explore the concepts of color spaces, how to run the color detection function on the robot, and the underlying code implementation.

20.2 Color Detection & Color Space

Color Space

Color space refers to a method of organizing colors. With the help of color space and tests of physical devices, we can get a certain analog and digital representation of colors. Color space can be defined by random colors. For example, Pantone series uses a group of specific colors as sample, and define each color with a name and code. Or, define it mathematically like Adobe RGB, sRGB, etc.

RGB Color Space

RGB uses additive colors as it describes a specific proportion by which various kinds of "light" produces colors. Starting from black, light keeps mixing to generate different colors. RGB means the value of red, green, and blue; RGBA produces a transparent color based on RGB with the alpha channel.

Common color spaces based on the RGB mode include sRGB, Adobe RGB and Adobe Wide Gamut RGB.

HSV Color Space

HSV, or hue, saturation, and value, also referred to as HSB (B for brightness), is commonly used by artists because it's more understandable to describe color by hue and saturation compared with terms of additive or subtractive color mixing. HSV is a variant of RGB color space and closely related in content and color standards as it derives from RGB.

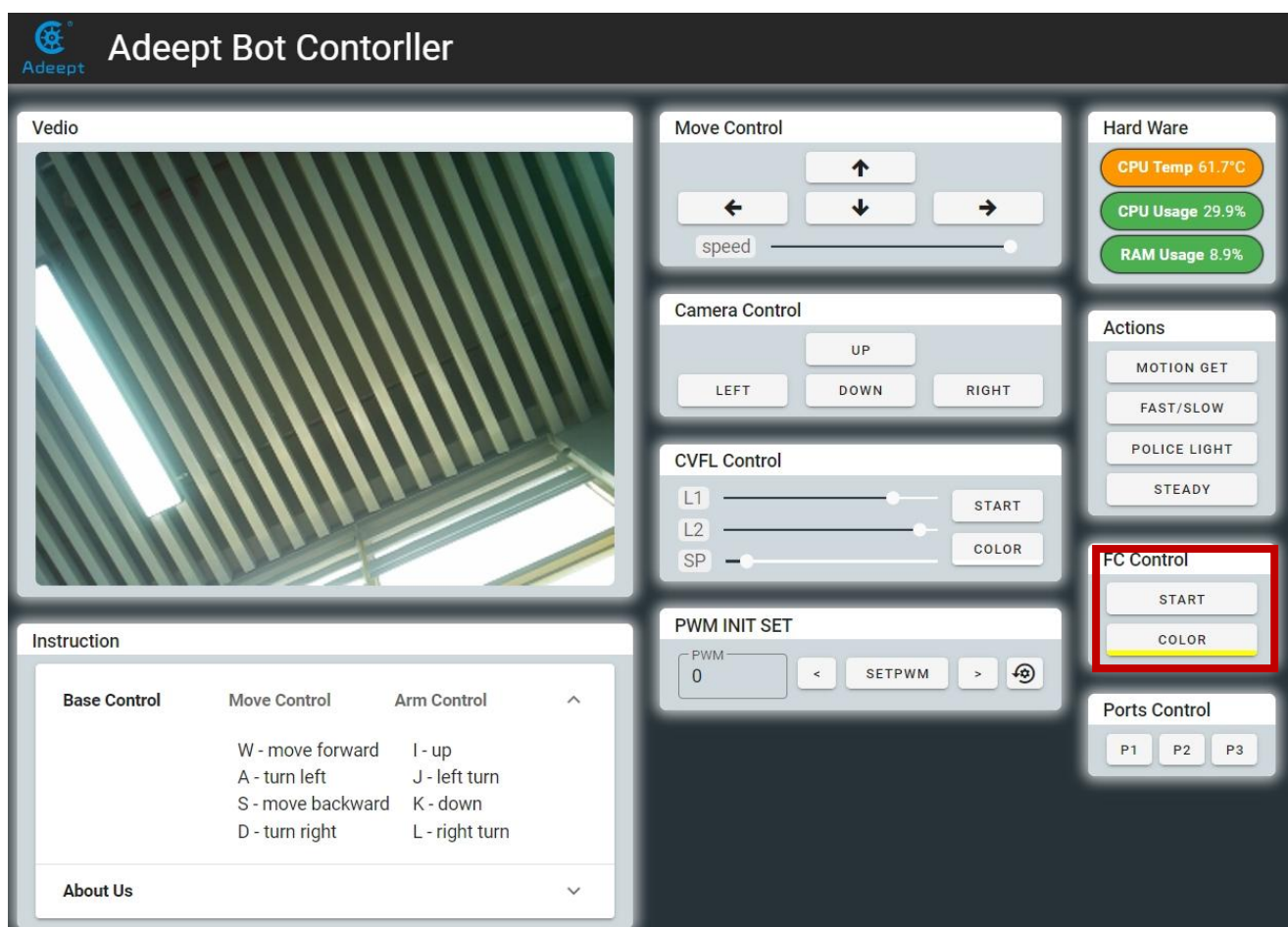
HSV color space is used for color detection with OpenCV since it's less effected by ambient light and brings more accurate detection results. Besides, it's easy to define color range with HSV.

During color detection, the code is to recognize some range of colors instead of some color. Therefore, the HSV color space is used for color detection since it is similar to the human perception of color.

20.3 Running the Color Detection Function

Running the Automatic Obstacle Avoidance program

1. Start the Adeep RaspClaws Robot. It may take about 30-50s to boot.
2. After Adeep RaspClaws is turned on, open the Chrome browser on your mobile or computer, enter the IP address of your Raspberry Pi and access port ":5000" into the IP address bar, like this: **192.168.3.31:5000**. The web controller will then be displayed on the browser.



3. After finding "FC Control", click the "START" button. RaspClaws Robot will detect yellow objects and label them accordingly.
4. Click "START" again to stop detecting.
5. To change the color to be detected, you can modify the code in **camera_opencv.py**.

```
colorUpper = np.array([44, 255, 255])
colorLower = np.array([24, 100, 100])
```

(44, 255, 255) and (24, 100, 100) represent the maximum and minimum values of HSV for yellow in (H, S, V). You can change them for your desired color. `colorUpper` and `colorLower` are the upper and lower range values respectively.

For example, to detect green objects, you can change it to:

```
colorUpper = np.array([77, 255, 255])
colorLower = np.array([35, 43, 36])
```

20.4 Code

The main code is as follows. For the complete code, please check [camera_opencv.py](#).

```
01 def findColor(self, frame_image):
02     hsv = cv2.cvtColor(frame_image, cv2.COLOR_BGR2HSV)
03     mask = cv2.inRange(hsv, colorLower, colorUpper)
04     mask = cv2.erode(mask, None, iterations=2)
05     mask = cv2.dilate(mask, None, iterations=2)
06     cnts = cv2.findContours(mask.copy(), cv2.RETR_EXTERNAL,
07                             cv2.CHAIN_APPROX_SIMPLE)[-2]
08     center = None
09     if len(cnts) > 0:
10         self.findColorDetection = 1
11         c = max(cnts, key=cv2.contourArea)
12         ((self.box_x, self.box_y), self.radius) = cv2.minEnclosingCircle(c)
13         M = cv2.moments(c)
14         center = (int(M["m10"] / M["m00"]), int(M["m01"] / M["m00"]))
15         X = int(self.box_x)
16         Y = int(self.box_y)
17         error_Y = 240 - Y
18         error_X = 320 - X
19         CVThread.serveMove(CVThread.P_servo, CVThread.T_direction, error_X)
20     else:
21         self.findColorDetection = 0
22     self.pause()
```