

### 2.1.3 Color tracking

Code path

[/home/pi/Yahboom\\_Project/2.AI\\_Visual\\_course/03.Mobile\\_tracking.ipynb](/home/pi/Yahboom_Project/2.AI_Visual_course/03.Mobile_tracking.ipynb)

import RPi.GPIO as GPIO import time
#bgr8 to jpeg format import enum import cv2  def bgr8_to_jpeg(value, quality=75): return bytes(cv2.imencode('.jpg', value)[1])
#display camera import cv2 import traitlets import ipywidgets.widgets as widgets from IPython.display import display import time # Thread function operation library import threading import inspect import ctypes  image_widget = widgets.Image(format='jpeg', width=300, height=300) display(image_widget)
#Servo pin definition ServoPin = 11  #S2 ServoPinB = 9  #S3 #Set GPIO port to BCM coding mode GPIO.setmode(GPIO.BCM)
#set servo pin to Output mode def init(): GPIO.setup(ServoPin, GPIO.OUT) GPIO.setup(ServoPinB, GPIO.OUT)
#Define a pulse function, used to simulate the pwm value #When base pulse is 20ms, the high level part of the pulse is controlled from 0 to 180 degrees in 0.5-2.5ms def servo_pulse(myangleA, myangleB): pulsewidth = myangleA GPIO.output(ServoPin, GPIO.HIGH) time.sleep(pulsewidth/1000000.0) GPIO.output(ServoPin, GPIO.LOW) time.sleep(20.0/1000-pulsewidth/1000000.0)  pulsewidthB = myangleB

```
GPIO.output(ServoPinB, GPIO.HIGH)
time.sleep(pulsewidthB/1000000.0)
GPIO.output(ServoPinB, GPIO.LOW)
time.sleep(20.0/1000-pulsewidthB/1000000.0)
```

#According to the steering gear pulse control range is 500-2500usec

```
def Servo_control(angle_1, angle_2):
```

```
    init()
    if angle_1 < 500:
        angle_1 = 500
    elif angle_1 > 2500:
        angle_1 = 2500
```

```
    if angle_2 < 500:
        angle_2 = 500
    elif angle_2 > 2500:
        angle_2 = 2500
```

```
    servo_pulse(angle_1, angle_2)
```

```
def _async_raise(tid, exctype):
```

```
    """raises the exception, performs cleanup if needed"""
```

```
    tid = ctypes.c_long(tid)
```

```
    if not inspect.isclass(exctype):
```

```
        exctype = type(exctype)
```

```
    res = ctypes.pythonapi.PyThreadState_SetAsyncExc(tid,
```

```
ctypes.py_object(exctype))
```

```
    if res == 0:
```

```
        raise ValueError("invalid thread id")
```

```
    elif res != 1:
```

```
        # """if it returns a number greater than one, you're in trouble,
```

```
        # and you should call it again with exc=NULL to revert the effect"""
```

```
        ctypes.pythonapi.PyThreadState_SetAsyncExc(tid, None)
```

```
def stop_thread(thread):
```

```
    _async_raise(thread.ident, SystemExit)
```

```
image = cv2.VideoCapture(0)
```

```
image.set(3, 320)
```

```
image.set(4, 240)
```

```
image.set(5, 90) #set frame
```

```
# fourcc = cv2.VideoWriter_fourcc(*"MPEG")
```

```
image.set(cv2.CAP_PROP_FOURCC, cv2.VideoWriter_fourcc('M', 'J', 'P', 'G'))
```

```
# image.set(cv2.CAP_PROP_BRIGHTNESS, 40) # -64 - 64 0.0
```

```
# image.set(cv2.CAP_PROP_CONTRAST, 50) # -64 - 64 2.0
```

```
# image.set(cv2.CAP_PROP_EXPOSURE, 156) # 1.0 - 5000 156.0
```

```
ret, frame = image.read()
```

```

image_widget.value = bgr8_to_jpeg(frame)

global color_x, color_y, color_radius
color_x = color_y = color_radius = 0
global target_valuex
target_valuex = 1500
global target_valuey
target_valuey = 1500
global g_mode
g_mode = 0

## Create an array to store HSV color gamut color classification data
import numpy as np
global color_lower
color_lower = np.array([156, 43, 46])
global color_upper
color_upper = np.array([180, 255, 255])

## set pid parameter
import PID
xservo_pid = PID.PositionalPID(1.1, 0.2, 0.8)
yservo_pid = PID.PositionalPID(0.8, 0.2, 0.8)

## Color selection button configuration
Redbutton = widgets.Button(
    value=False,
    description='Red',
    disabled=False,
    button_style="", # 'success', 'info', 'warning', 'danger' or ''
    tooltip='Description',
    icon='uncheck' )
Greenbutton = widgets.Button(
    value=False,
    description='Green',
    disabled=False,
    button_style="", # 'success', 'info', 'warning', 'danger' or ''
    tooltip='Description',
    icon='uncheck' )
Bluebutton = widgets.Button(
    value=False,
    description='Blue',
    disabled=False,
    button_style="", # 'success', 'info', 'warning', 'danger' or ''
    tooltip='Description',
    icon='uncheck' )
Yellowbutton = widgets.Button(
    value=False,
    description='Yellow',

```

```

        disabled=False,
        button_style="", # 'success', 'info', 'warning', 'danger' or ''
        tooltip='Description',
        icon='unchecked' )
Orangebutton = widgets.Button(
    value=False,
    description='Orange',
    disabled=False,
    button_style="", # 'success', 'info', 'warning', 'danger' or ''
    tooltip='Description',
    icon='unchecked' )
Closebutton = widgets.Button(
    value=False,
    description='Closed',
    disabled=False,
    button_style="", # 'success', 'info', 'warning', 'danger' or ''
    tooltip='Description',
    icon='unchecked' )
output = widgets.Output()
display(Redbutton, Greenbutton, Bluebutton, Yellowbutton, Orangebutton,
Closebutton, output)

def ALL_Uncheck():
    Redbutton.icon = 'unchecked'
    Greenbutton.icon = 'unchecked'
    Bluebutton.icon = 'unchecked'
    Yellowbutton.icon = 'unchecked'
    Orangebutton.icon = 'unchecked'

def on_Redbutton_clicked(b):
    global color_lower, color_upper, g_mode
    global target_valuex, target_valuey
    ALL_Uncheck()
    b.icon = 'check'
    color_lower = np.array([0, 43, 46])
    color_upper = np.array([10, 255, 255])
    target_valuex = target_valuey = 2048
    Servo_control(1500, 1500)
    g_mode = 1
    with output:
        print("RedButton clicked.")

def on_Greenbutton_clicked(b):
    global color_lower, color_upper, g_mode

```

```

global target_valuex, target_valuey
ALL_Uncheck()
b.icon = 'check'
color_lower = np.array([35, 43, 46])
color_upper = np.array([77, 255, 255])
target_valuex = target_valuey = 2048
Servo_control(1500, 1500)
g_mode = 1
with output:
    print("GreenButton clicked.")

def on_Bluebutton_clicked(b):
    global color_lower, color_upper, g_mode
    global target_valuex, target_valuey
    ALL_Uncheck()
    b.icon = 'check'
    color_lower=np.array([100, 43, 46])
    color_upper = np.array([124, 255, 255])
    target_valuex = target_valuey = 2048
    Servo_control(1500, 1500)
    g_mode = 1
    with output:
        print("Bluebutton clicked.")

def on_Yellowbutton_clicked(b):
    global color_lower, color_upper, g_mode
    global target_valuex, target_valuey
    ALL_Uncheck()
    b.icon = 'check'
    color_lower = np.array([26, 43, 46])
    color_upper = np.array([34, 255, 255])
    target_valuex = target_valuey = 2048
    Servo_control(1500, 1500)
    g_mode = 1
    with output:
        print("Yellowbutton clicked.")

def on_Orangebutton_clicked(b):
    global color_lower, color_upper, g_mode
    global target_valuex, target_valuey
    ALL_Uncheck()
    b.icon = 'check'
    color_lower = np.array([11, 43, 46])
    color_upper = np.array([25, 255, 255])

```

```

target_valuex = target_valuey = 2048
Servo_control(1500, 1500)
g_mode = 1
with output:
    print("Orangebutton clicked.")

def on_Closebutton_clicked(b):
    global g_mode

    ALL_Uncheck()
    g_mode = 0
    with output:
        print("CloseButton clicked.")

Redbutton.on_click(on_Redbutton_clicked)
Greenbutton.on_click(on_Greenbutton_clicked)
Bluebutton.on_click(on_Bluebutton_clicked)
Yellowbutton.on_click(on_Yellowbutton_clicked)
Orangebutton.on_click(on_Orangebutton_clicked)
Closebutton.on_click(on_Closebutton_clicked)

##  main loop
def Color_track():
    global color_lower, color_upper, g_mode
    global target_valuex, target_valuey
    t_start = time.time()
    fps = 0
    while True:
        ret, frame = image.read()
        frame = cv2.resize(frame, (300, 300))
        frame_ = cv2.GaussianBlur(frame,(5,5),0)
        hsv = cv2.cvtColor(frame,cv2.COLOR_BGR2HSV)
        mask = cv2.inRange(hsv,color_lower,color_upper)
        mask = cv2.erode(mask,None,iterations=2)
        mask = cv2.dilate(mask,None,iterations=2)
        mask = cv2.GaussianBlur(mask,(3,3),0)
        cnts =
cv2.findContours(mask.copy(),cv2.RETR_EXTERNAL,cv2.CHAIN_APPROX_SIMPLE)[-2]
        if g_mode == 1: # Push button switch
            if len(cnts) > 0:
                cnt = max (cnts, key = cv2.contourArea)
                (color_x,color_y),color_radius = cv2.minEnclosingCircle(cnt)
                if color_radius > 10:
                    # Mark the detected colors

```

```

cv2.circle(frame,(int(color_x),int(color_y)),int(color_radius),(255,0,255),2)
        # Proportion-Integration-Differentiation
        xservo_pid.SystemOutput = color_x
        print(color_x)
        xservo_pid.SetStepSignal(150)
        xservo_pid.SetInertiaTime(0.01, 0.1)
        target_valuex = int(1500+xservo_pid.SystemOutput)
        yservo_pid.SystemOutput = color_y
        yservo_pid.SetStepSignal(150)
        yservo_pid.SetInertiaTime(0.01, 0.1)
        target_valuey = int(1500+yservo_pid.SystemOutput)
        Servo_control(target_valuex,target_valuey)

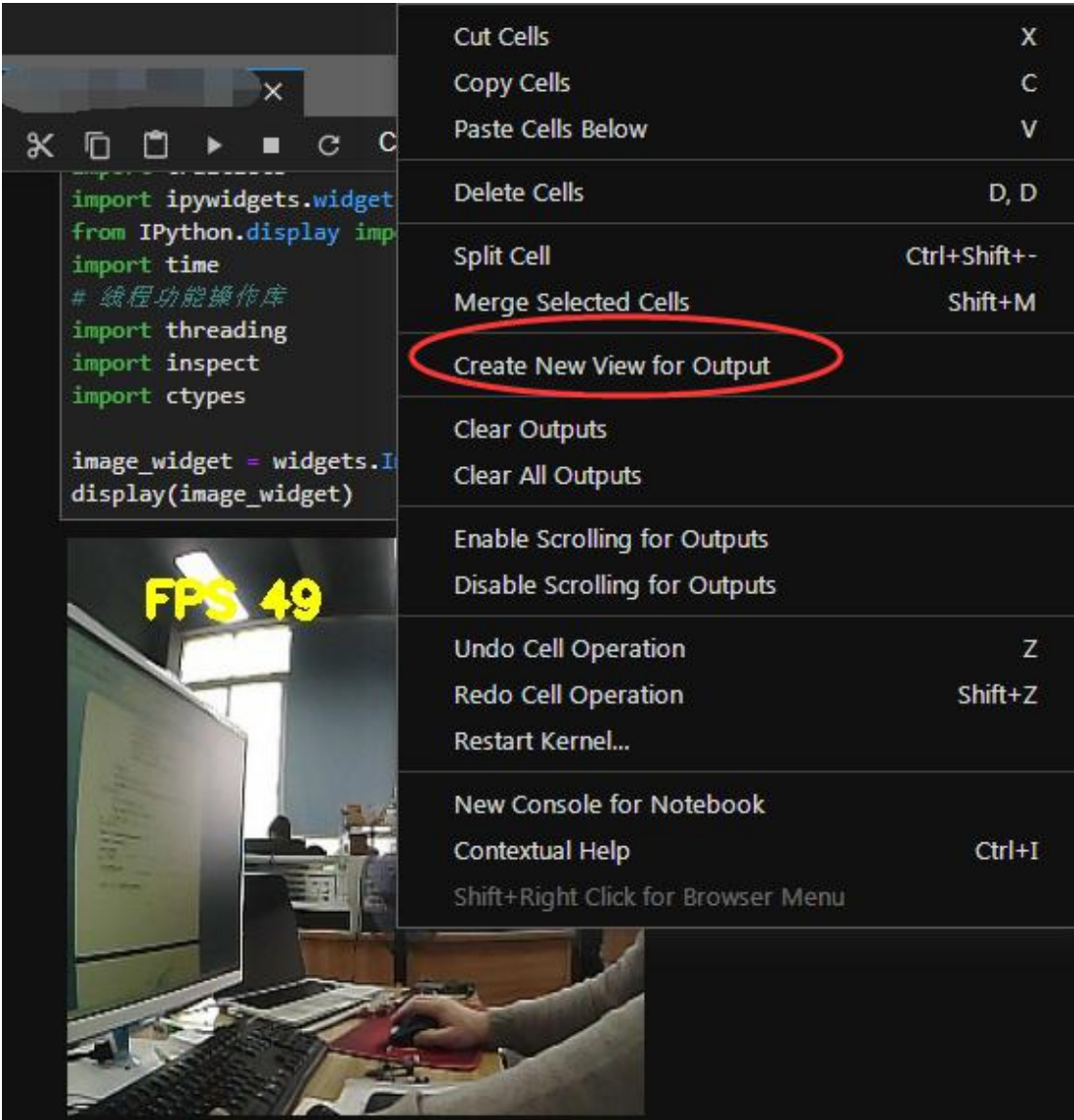
        fps = fps + 1
        mfps = fps / (time.time() - t_start)
        cv2.putText(frame, "FPS " + str(int(mfps)), (40,40),
cv2.FONT_HERSHEY_SIMPLEX, 0.8, (0,255,255), 3)
        # display image in real time
        image_widget.value = bgr8_to_jpeg(frame)

##  start thread
thread1 = threading.Thread(target=Color_track)
thread1.setDaemon(True)
thread1.start()

##  close thread
stop_thread(thread1)

```

Tip: We can put the components in other Windows as shown below.  
Click "right button" of mouse --> **【Create New View for Output】** .



The image shows a Jupyter Notebook interface. On the left, a code cell contains the following Python code:

```
import ipywidgets.widget
from IPython.display import
import time
# 线程功能操作库
import threading
import inspect
import ctypes

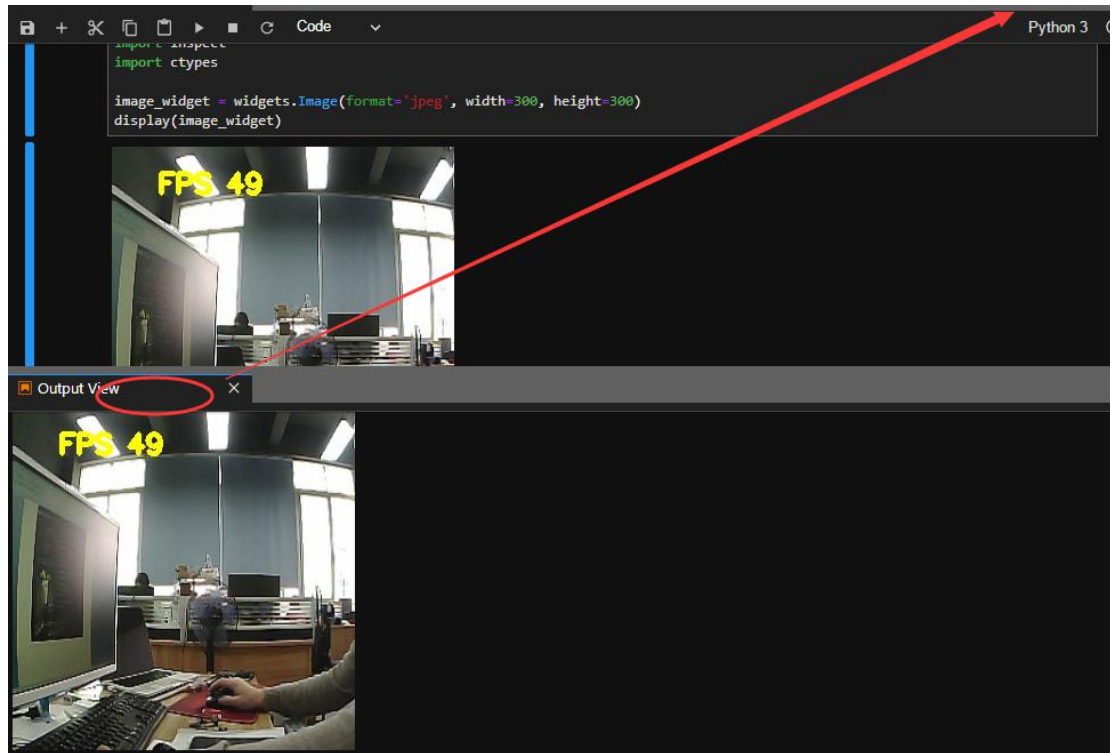
image_widget = widgets.I
display(image_widget)
```

Below the code cell, a video frame shows a person's hands using a mouse on a desk. The text "FPS 49" is overlaid in yellow on the video frame.

On the right, a context menu is open, listing various actions and their keyboard shortcuts. The option "Create New View for Output" is circled in red.

Cut Cells	X
Copy Cells	C
Paste Cells Below	V
Delete Cells	D, D
Split Cell	Ctrl+Shift+-
Merge Selected Cells	Shift+M
Create New View for Output	
Clear Outputs	
Clear All Outputs	
Enable Scrolling for Outputs	
Disable Scrolling for Outputs	
Undo Cell Operation	Z
Redo Cell Operation	Shift+Z
Restart Kernel...	
New Console for Notebook	
Contextual Help	Ctrl+I
Shift+Right Click for Browser Menu	





We also pulled button interface to the right together, as shown below.  
We can choose different color, camera will tracking it.

