1. Experiment preparation

Each block needs to be placed in the corresponding color area on map.

2. About code

Path: /home/dofbot/Dofbot/6.AI_Visual/3.color_grab.ipynb

```
#bgr8 to jpeg format
import enum
import cv2

def bgr8_to_jpeg(value, quality=75):
    return bytes(cv2.imencode('.jpg', value)[1])
```

```
#Import library
from Arm_Lib import Arm_Device

Arm = Arm_Device()
```

```
import traitlets
import ipywidgets.widgets as widgets
import time

import threading
import inspect
import ctypes

origin_widget = widgets.Image(format='jpeg', width=320, height=240)
mask_widget = widgets.Image(format='jpeg',width=320, height=240)
result_widget = widgets.Image(format='jpeg',width=320, height=240)
```

```
image_container = widgets.HBox([origin_widget, mask_widget, result_widget])
# image_container = widgets.Image(format='jpeg', width=600, height=500)
display(image_container)
```

```
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:

        ctypes.pythonapi.PyThreadState_SetAsyncExc(tid, None)

def stop_thread(thread):
    _async_raise(thread.ident, SystemExit)
```

```
def get_color(img):
    H = []
    color_name={}
    img = cv2.resize(img, (640, 480), )

    HSV = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)

    cv2.rectangle(img, (280, 180), (360, 260), (0, 255, 0), 2)
    #Take out the H, S, V values of each row and each column in turn and put them into the container
    for i in range(280, 360):
        for j in range(180, 260): H.append(HSV[j, i][0])
    #Calculate the maximum and minimum of H, S, and V respectively
    H_min = min(H);H_max = max(H)
#       print(H_min,H_max)
    #Judging the color
    if H_min >= 0 and H_max <= 10 or H_min >= 156 and H_max <= 180: color_name['name'] = 'red'
    elif H_min >= 26 and H_max <= 34: color_name['name'] = 'yellow'
    elif H_min >= 35 and H_max <= 78: color_name['name'] = 'green'
    elif H_min >= 100 and H_max <= 124: color_name['name'] = 'blue'
    return img, color_name
```

```
# Define variable parameters at different locations
look_at = [90, 164, 18, 0, 90, 90]
p_top = [90, 80, 50, 50, 270]
```

```
p_Yellow = [65, 22, 64, 56, 270]
p_Red = [118, 19, 66, 56, 270]

p_Green = [136, 66, 20, 29, 270]
p_Blue = [44, 66, 20, 28, 270]

p_gray = [90, 48, 35, 30, 270]
```

```
# Define control DOFBOT function, control No.1-No.6 servo，p=[S1,S2,S3,S4,S5,S6]
def arm_move_6(p, s_time = 500):
    for i in range(6):
        id = i + 1
        Arm.Arm_serial_servo_write(id, p[i], s_time)
        time.sleep(.01)
    time.sleep(s_time/1000)

#Define control DOFBOT function, control No.1-No.5 servo，p=[S1,S2,S3,S4,S5]
def arm_move(p, s_time = 500):
    for i in range(5):
        id = i + 1
        if id == 5:
            time.sleep(.1)
            Arm.Arm_serial_servo_write(id, p[i], int(s_time*1.2))
        elif id == 1 :
            Arm.Arm_serial_servo_write(id, p[i], int(3*s_time/4))
        else:
            Arm.Arm_serial_servo_write(id, p[i], int(s_time))
        time.sleep(.01)
    time.sleep(s_time/1000)

# enable=1：clip，=0：release
def arm_clamp_block(enable):
    if enable == 0:
        Arm.Arm_serial_servo_write(6, 60, 400)
    else:
        Arm.Arm_serial_servo_write(6, 130, 400)
    time.sleep(.5)
```

```
arm_move_6(look_at, 1000)
time.sleep(1)
```

```
global g_state_arm
g_state_arm = 0

def ctrl_arm_move(index):
```

```
    arm_clamp_block(0)
    if index == 1:
        print("Yellow")
        number_action(index)
        put_down_block()
    elif index == 2:
        print("Red")
        number_action(index)
        put_down_block()
    elif index == 3:
        print("Green")
        number_action(index)
        put_down_block()
    elif index == 4:
        print("Blue")
        number_action(index)
        put_down_block()

    global g_state_arm
    g_state_arm = 0
```

```
def start_move_arm(index):
    global g_state_arm
    if g_state_arm == 0:
        closeTid = threading.Thread(target = ctrl_arm_move, args = [index])
        closeTid.setDaemon(True)
        closeTid.start()

        g_state_arm = 1
```

```
# Main process
```

```
import cv2
import numpy as np
import ipywidgets.widgets as widgets


cap = cv2.VideoCapture(0)
cap.set(3, 640)
cap.set(4, 480)
cap.set(5, 30)
cap.set(cv2.CAP_PROP_FOURCC, cv2.VideoWriter.fourcc('M', 'J', 'P', 'G'))

# Red is selected by default, and the program will automatically switch the color according to the color detected in the box
# Red value
```

```
color_lower = np.array([0, 43, 46])
color_upper = np.array([10, 255, 255])

# Green value
# color_lower = np.array([35, 43, 46])
# color_upper = np.array([77, 255, 255])

# Blue value
# color_lower=np.array([100, 43, 46])
# color_upper = np.array([124, 255, 255])

# Yellow value
# color_lower = np.array([26, 43, 46])
# color_upper = np.array([34, 255, 255])

# Orange value
# color_lower = np.array([11, 43, 46])
# color_upper = np.array([25, 255, 255])


def Color_Recongnize():
    Arm.Arm_Buzzer_On(1)
    s_time = 300
    Arm.Arm_serial_servo_write(4, 10, s_time)
    time.sleep(s_time/1000)
    Arm.Arm_serial_servo_write(4, 0, s_time)
    time.sleep(s_time/1000)
    Arm.Arm_serial_servo_write(4, 10, s_time)
    time.sleep(s_time/1000)
    Arm.Arm_serial_servo_write(4, 0, s_time)
    time.sleep(s_time/1000)

    while(1):
        ret, frame = cap.read()
        frame, color_name = get_color(frame)
        if len(color_name)==1:
            global color_lower
            global color_upper
#               print ("color_name :", color_name)
#               print ("name :", color_name['name'])
            if color_name['name'] == 'yellow':
                color_lower = np.array([26, 43, 46])
                color_upper = np.array([34, 255, 255])
```

```
                start_move_arm(1)
            elif color_name['name'] == 'red':
                color_lower = np.array([0, 43, 46])
                color_upper = np.array([10, 255, 255])
                start_move_arm(2)
            elif    color_name['name'] == 'green':
                color_lower = np.array([35, 43, 46])
                color_upper = np.array([77, 255, 255])
                start_move_arm(3)
            elif color_name['name'] == 'blue':
                color_lower=np.array([100, 43, 46])
                color_upper = np.array([124, 255, 255])
                start_move_arm(4)


        origin_widget.value = bgr8_to_jpeg(frame)
        #cv2.imshow('Capture', frame)

        # change to hsv model
        hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)
        mask = cv2.inRange(hsv, color_lower, color_upper)
        #cv2.imshow('Mask', mask)
        mask_widget.value = bgr8_to_jpeg(mask)
        res = cv2.bitwise_and(frame, frame, mask=mask)
        #cv2.imshow('Result', res)
        result_widget.value = bgr8_to_jpeg(res)

        #         if cv2.waitKey(1) & 0xFF == ord('q'):
        #             break
        time.sleep(0.01)


    cap.release()
    #cv2.destroyAllWindows()
```

```
thread1 = threading.Thread(target=Color_Recongnize)
thread1.setDaemon(True)
thread1.start()
```

```
#End process, only need to execute this code at the end
stop_thread(thread1)
```

After running the above program,

Function 1: Put the block in front of the camera, the camera will detect the color of the block, and then DOFBOT will place the block from the middle area on the map to the area of the corresponding color.

Function 2: Put the blocks in the corresponding color area on the map, and then find the yellow, red, green, and blue cards and put them in front of the camera. The DOFBOT will grab the corresponding blocks according to the detected colors. Place it in the middle area on the map.
Note: Every time you grab a block, you need to remove the block in the middle area on map, otherwise it will block the next block.