Color recognition grabbing building blocks

1. Experimental placement

The experimental placement position is to place yellow building blocks in the yellow area, red building blocks in the red area, green building blocks in the green area, and blue building blocks in the blue area.

2. Code content

Code path:/root/Dofbot/6.AI_Visual/3.color_grab.ipynb

The following code content needs to be executed according to the actual step. It cannot be run all at once. Running the last unit will directly exit the thread.

```
#bgr8转jpeg格式
import   enum
import   cv2

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

```
#导入相关的模块
import   threading
import   time
from   Arm_Lib import Arm_Device

#   创建机械臂对象
Arm   = Arm_Device()
time.sleep(.1)
```

```
#摄像头组件显示
import   traitlets
import   ipywidgets.widgets as widgets
import   time

#   线程功能操作库
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)

#   创建一个水平框容器，将图像小部件相邻放置
image_container   = widgets.HBox([origin_widget, mask_widget, result_widget])
#   image_container = widgets.Image(format='jpeg', width=600, height=500)
display(image_container)
```

```
#线程相关函数
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:
        # 如果res的值大于1，则需要异常处理
          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), )
    # 将彩色图转成HSV
    HSV = cv2.cvtColor(img,   cv2.COLOR_BGR2HSV)
    # 画矩形框
```

```
    cv2.rectangle(img, (280, 180), (360,   260), (0, 255, 0), 2)
    # 依次取出每行每列的H,S,V值放入容器中
    for i in range(280, 360):
        for j in range(180, 260):   H.append(HSV[j, i][0])
    # 分别计算出H,S,V的最大最小
    H_min = min(H);H_max = max(H)
#     print(H_min,H_max)
   # 判断颜色

    if H_min >= 0 and H_max
    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
```

```
#   定义不同位置的变量参数
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]
```

```
#   定义移动机械臂函数,同时控制1-6号舵机运动，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)

#   定义移动机械臂函数,同时控制1-5号舵机运动，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)

#   定义夹积木块函数，enable=1：夹住，=0：松开
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("黄色")
        number_action(index)
        put_down_block()

    elif index == 2:
        print("红色")
        number_action(index)
        put_down_block()

    elif index == 3:
        print("绿色")
        number_action(index)
        put_down_block()

    elif index == 4:
        print("蓝色")
        number_action(index)
        put_down_block()

    global g_state_arm

    g_state_arm = 0
```

```
def   start_move_arm(index):

    # 开启机械臂控制线程
    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
```

```
#   主进程
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)  #设置帧率
cap.set(cv2.CAP_PROP_FOURCC,   cv2.VideoWriter.fourcc('M', 'J', 'P', 'G'))
#   image.set(cv2.CAP_PROP_BRIGHTNESS, 40) #设置亮度 -64 - 64  0.0
#   image.set(cv2.CAP_PROP_CONTRAST, 50)     #设置对比度 -64 - 64  2.0
#   image.set(cv2.CAP_PROP_EXPOSURE, 156)    #设置曝光值 1.0 - 5000  156.0
#   默认选择红色的,程序会自动根据方框中检测到的颜色切换颜色
#   红色区间
color_lower   = np.array([0, 43, 46])
color_upper   = np.array([10, 255, 255])
#   #绿色区间
#   color_lower = np.array([35, 43, 46])
#   color_upper = np.array([77, 255, 255])
#   #蓝色区间
#   color_lower=np.array([100, 43, 46])
#   color_upper = np.array([124, 255, 255])
#   #黄色区间
#   color_lower = np.array([26, 43, 46])
#   color_upper = np.array([34, 255, 255])
#   #橙色区间
#   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):
        # get a frame and show 获取视频帧并转成HSV格式，利用cvtColor()将BGR格式转成HSV格式，参数为cv2.COLOR_BGR2HSV。
        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)

        # get mask 利用inRange()函数和HSV模型中蓝色范围的上下界获取mask，mask中原视频中的蓝色部分会被弄成白色，其他部分黑色。
        mask = cv2.inRange(hsv, color_lower,   color_upper)
        #cv2.imshow('Mask', mask)
        mask_widget.value =   bgr8_to_jpeg(mask)

        # detect blue 将mask于原视频帧进行按位与操作，则会把mask中的白色用真实的图像替换：
        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()
```

```
#启动进程
thread1   = threading.Thread(target=Color_Recongnize)
thread1.setDaemon(True)
thread1.start()
```

```
#结束进程，只有在结束时才需要执行此段代码
stop_thread(thread1)
```

You can place the building blocks in front of the camera. The camera will detect the color of the building blocks, and then place the building blocks in the area of the corresponding color. The robotic arm will pick up the building blocks in the corresponding area to the middle based on the detected color. Area. Or first place the building blocks in the corresponding color area, then find cards in four colors of yellow, red, green, and blue and place them in front of the camera. The robotic arm will grab the corresponding building blocks based on the detected colors and place them in the middle. area.

Note: Each time you grab a building block, you need to remove the building blocks in the middle area, otherwise it will block the next time you place the building blocks.