Let me catch you

1. Introduction to gameplay

You let me catch gameplay path: /root/dofbot_ws/src/dofbot_color_sorting/you let me catch.ipynb

Start the code block, and the robot arm will automatically reach the recognition position and place the block in the gray square. After the robot arm has correctly recognized it for 10 times, it will automatically grab it and place it in the square of the corresponding color.

- Import header files

```
#!/usr/bin/env python
# coding: utf-8
import cv2 as cv
import threading
from time import sleep
from dofbot_config import *
import ipywidgets as widgets
from IPython.display import display
from color_sorting import color_sorting
```

- Initialize the robot arm position

```
import Arm_Lib
Arm = Arm_Lib.Arm_Device()
joints_0 = [90, 135, 0, 45, 90, 30]
Arm.Arm_serial_servo_write6_array(joints_0, 1000)
```

- Create an instance and initialize parameters

```
#Create instance
sorting = color_sorting()
#Initialization mode
model = 'General'
# Color HSV threshold
color_hsv = {"red" : ((0, 43, 46), (10, 255, 255)),
         "green" : ((35, 43, 46), (77, 255, 255)),
         "blue" : ((100, 43, 46), (124, 255, 255)),
         "yellow": ((26, 43, 46), (34, 255, 255))}
# HSV parameter path
HSV_path="/root/dofbot_ws/src/dofbot_color_sorting/HSV_config.txt"
# Read the HSV configuration file and update the HSV value
try: read_HSV(HSV_path,color_hsv)
except Exception: print("Read HSV_config Error!!!")
```

- Create controls

```
#Create control layout
button_layout = widgets.Layout(width='200px', height='70px',align_self='center')
# Output printing
output = widgets.Output()
# Exit button
exit_button=widgets.Button(description='Exit',button_style='danger',layout=button
_layout)
#Image control
imgbox = widgets.Image(format='jpg', height=480, width=640,
layout=widgets.Layout(align_self='center'))
# Place vertically
controls_box = widgets.VBox([imgbox,
exit_button],layout=widgets.Layout(align_self='center'))
# ['auto','flex-start', 'flex-end', 'center', 'baseline', 'stretch', 'inherit',
'initial', 'unset']
```

- control button

```
def exit_button_Callback(value):
    global model
    model = 'Exit'
# with output: print(model)
exit_button.on_click(exit_button_Callback)
```

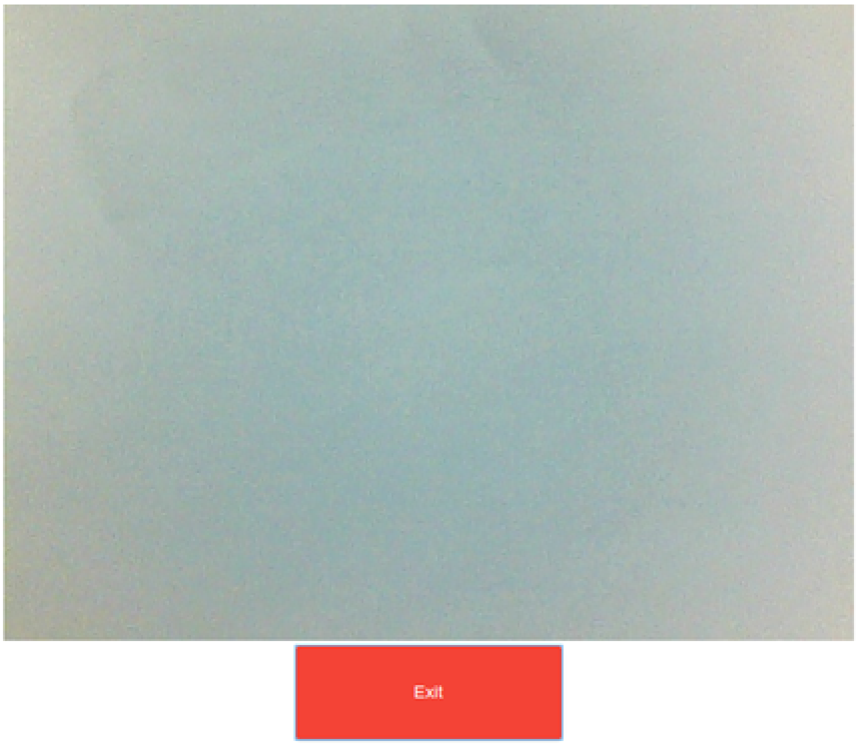
- Main program

```
def camera():
    #Open camera
    capture = cv.VideoCapture(0)
    # Loop execution when the camera is turned on normally
    while capture.isOpened():
        try:
            # Read each frame of the camera
            _, img = capture.read()
            # Unify image size
            img = cv.resize(img, (640, 480))
            # Get sports information
            img = sorting.Sorting_grap(img, color_hsv)
            if model == 'Exit':
                cv.destroyAllWindows()
                capture.release()
                break
            # Add text
            imgbox.value = cv.imencode('.jpg', img)[1].tobytes()
        except KeyboardInterrupt:capture.release()
```

- start up

```
display(controls_box,output)
threading.Thread(target=camera, ).start()
```

- Sample images



For the detailed gameplay code, please see /root/dofbot_ws/src/dofbot_color_sorting/you let me catch.ipynb

2. Library code design

- HSV filters out the target color

```
# Convert image to HSV.
HSV_img = cv.cvtColor(mask, cv.COLOR_BGR2HSV)
# Filter out elements between two arrays.
img = cv.inRange(HSV_img, lowerb, upperb)
# Set all non-mask detection parts to black
mask[img == 0] = [0, 0, 0]
```

- Morphological transformation

```
# Get structural elements of different shapes
kernel = cv.getStructuringElement(cv.MORPH_RECT, (5, 5))
# Morphological closing operation
dst_img = cv.morphologyEx(mask, cv.MORPH_CLOSE, kernel)
```

- Find contours in binary images

```
# Convert image to grayscale
dst_img = cv.cvtColor(dst_img, cv.COLOR_RGB2GRAY)
# Image binarization operation
ret, binary = cv.threshold(dst_img, 10, 255, cv.THRESH_BINARY)
# Get the contour point set (coordinates)
contours, heriachy = cv.findContours(binary, cv.RETR_EXTERNAL,
cv.CHAIN_APPROX_SIMPLE)
```

- Calculate outline borders

```
for i, cnt in enumerate(contours):
     # The boundingRect function calculates the border value, x, y are the coordinate values, w, h are the width and height of the rectangle
     x, y, w, h = cv.boundingRect(cnt)
     # Calculate the area of the contour
     area = cv.contourArea(cnt)
```

Obtain the target and drive the robotic arm to grab. The grabbing process is as follows:

Lift the robotic arm -> release the clamping jaw -> move to the block position -> tighten the clamping claw -> lift -> move to the target position -> release the clamping claw -> reset the robotic arm.