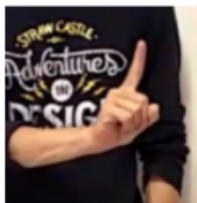


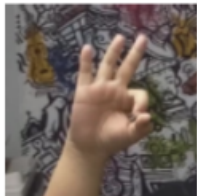

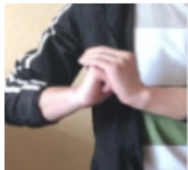
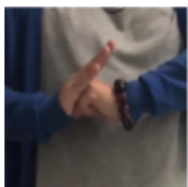
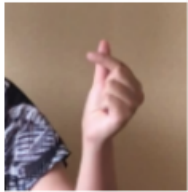


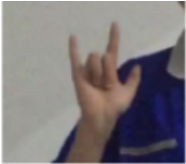

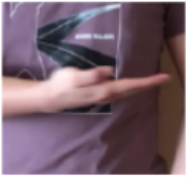
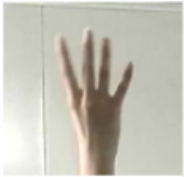

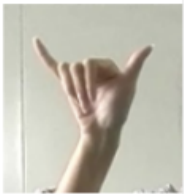

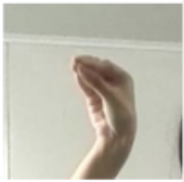
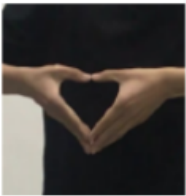
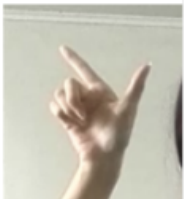
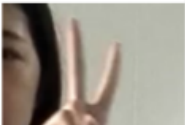
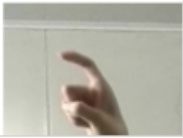



1. Gesture recognition stack blocks

1.1 Gesture recognition instructions

1. The gesture recognition used in this routine is based on the service of Baidu Intelligent Cloud Platform. This service has 50,000 free use opportunities every day. It is for learning only and is not for commercial use. If you need long-term use, please purchase related services. Our company will not be responsible.
2. For KEY application instructions and API documentation, please see the course content in the previous section.
3. Gestures supported by gesture recognition and example images

Serial number	Gestures name	Sample
1	number_1	
2	number_5	
3	fist	
4	ok	
5	pray	
6	congratulation	
7	honour	
8	heart_single	
9	thumb_up	
10	thumb_down	

11	i_love_you		17	number_3	
12	palm_up		18	number_4	
13	heart_1		19	number_6	
14	heart_2		20	number_7	
15	heart_3		21	number_8	
16	number_2		22	number_9	

23	rock	
----	------	--

1.2 Experimental placement

The experimental placement is as follows: yellow building blocks are placed in the yellow area, red building blocks are placed in the red area, green building blocks are placed in the green area, and blue building blocks are placed in the blue area.



1.3 Code content

Code path: /home/dofbot/Dofbot/6.AI_Visual/2.gesture_stack.ipynb

```
#bgr8 to jpeg format
import enum
import cv2
def bgr8_to_jpeg(value, quality=75):
    return bytes(cv2.imencode('.jpg', value)[1])
```

```
#Import related modules

import threading
import time
from Arm_Lib import Arm_Device

# Create robot arm object
Arm = Arm_Device()
time.sleep(.1)
```

```
# Define gesture recognition function part
import cv2

import time

import demjson
```

```

import pygame

from aip import AipBodyAnalysis
from aip import AipSpeech
from PIL import Image, ImageDraw, ImageFont
import numpy
import ipywidgets.widgets as widgets

# For specific gestures, please see the official website
https://ai.baidu.com/ai-doc/BODY/4k3cpywrv

hand={ 'One': 'number_1', 'Two': 'number_2', 'Three': 'number_3', 'Four': 'number_4',

        'Five': 'number_5', 'Six': 'number_6', 'Seven': 'number_7',

        'Eight': 'number_8', 'Nine': 'number_9', 'Fist': 'fist', 'Ok': 'OK',

        'Prayer': 'pray', 'Congratulation': 'Congratulation', 'Honour': 'honou',

        'Heart_single': 'heart_single', 'Thumb_up': 'thumb_up', 'Thumb_down': 'Diss',

        'ILY': 'i love you', 'Palm_up': 'Palm up', 'Heart_1': 'Heart1',

        'Heart_2': 'Heart2', 'Heart_3': 'Heart3', 'Rock': 'Rock', 'Face': 'face'}

# The following keys need to be replaced with your own

""" human body analysis APPID AK SK """

APP_ID = '18550528'

API_KEY = 'K6PWqtiUTKYK1fYaz1308E3i'

SECRET_KEY = 'IDBUII1j6srF1XVNDX32I2WpuWBwczzK'

client = AipBodyAnalysis(APP_ID, API_KEY, SECRET_KEY)

g_camera = cv2.VideoCapture(0)
g_camera.set(3, 640)
g_camera.set(4, 480)
g_camera.set(5, 30) #Set frame rate
g_camera.set(cv2.CAP_PROP_FOURCC, cv2.VideoWriter_fourcc('M', 'J', 'P', 'G'))
g_camera.set(cv2.CAP_PROP_BRIGHTNESS, 40) #Set brightness -64 - 64 0.0
g_camera.set(cv2.CAP_PROP_CONTRAST, 50) #Set contrast -64 - 64 2.0
g_camera.set(cv2.CAP_PROP_EXPOSURE, 156) #Set exposure value 1.0 - 5000 156.0

ret, frame = g_camera.read()

```

```
# Define the camera display component

image_widget = widgets.Image(format='jpeg', width=600, height=500) #Set up
the camera display component

display(image_widget)

image_widget.value = bgr8_to_jpeg(frame)
```

```
# Define conversion display Chinese function

def cv2ImgAddText(img, text, left, top, textColor=(0, 255, 0), textSize=20):

    if (isinstance(img, numpy.ndarray)): # Determine whether the OpenCV picture
type

        img = Image.fromarray(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))

    # Create an object that can draw on the given image

    draw = ImageDraw.Draw(img)

    # Font format

    fontStyle = ImageFont.truetype(

        "simhei.ttf", textSize, encoding="utf-8")

    # draw text

    draw.text((left, top), text, textColor, font=fontStyle)

    # Convert back to OpenCV format

    return cv2.cvtColor(numpy.asarray(img), cv2.COLOR_RGB2BGR)
```

```
# 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_layer_4 = [90, 76, 40, 17, 270]
p_layer_3 = [90, 65, 44, 17, 270]
p_layer_2 = [90, 65, 25, 36, 270]
p_layer_1 = [90, 48, 35, 30, 270]

p_push_over_1 = [90, 90, 5, 0, 90, 150]
p_push_over_2 = [90, 90, 0, 50, 90, 150]
```

```
#Define the state of grabbing blocks
```

```
yellow_grabbed = 0
```

```
red_grabbed = 0
```

```
green_grabbed = 0
```

```
blue_grabbed = 0
```

```
#Define the number of gesture recognition times
```

```
Count_One = 0
```

```
Count_Two = 0
```

```
Count_Three = 0
```

```
Count_Four = 0
```

```
Count_Fist = 0
```

```
# Define the mobile robot arm function to simultaneously control the movement  
of servos No. 1-6, 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 the mobile robot arm function to simultaneously control the movement of  
servos No. 1-5, 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)
```

```

# Define the function of clamping building blocks, enable=1: clamp, =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)

```

#Digital function definition

```

def number_action(index):

    if index == 1:

        # Grab the yellow building blocks

        arm_move(p_top, 1000)

        arm_move(p_Yellow, 1000)

        arm_clamp_block(1)

        # time.sleep(.5)

        arm_move(p_top, 1000)

    elif index == 2:

        # Grab the red building block

        arm_move(p_top, 1000)

        arm_move(p_Red, 1000)

        arm_clamp_block(1)

        arm_move(p_top, 1000)

    elif index == 3:

        # Grab the green building blocks

        arm_move(p_top, 1000)

        arm_move(p_Green, 1000)

        arm_clamp_block(1)

        arm_move(p_top, 1000)

    elif index == 4:

```

```
# Grab the blue building blocks

arm_move(p_top, 1000)

arm_move(p_Blue, 1000)

arm_clamp_block(1)

arm_move(p_top, 1000)


def put_down_block(layer):

    if layer == 1:

        arm_move(p_layer_1, 1000)

        arm_clamp_block(0)

        arm_move_6(look_at, 1000)

    elif layer == 2:

        arm_move(p_layer_2, 1000)

        arm_clamp_block(0)

        arm_move_6(look_at, 1000)

    elif layer == 3:

        arm_move(p_layer_3, 1000)

        arm_clamp_block(0)

        arm_move_6(look_at, 1000)

    elif layer == 4:

        arm_move(p_layer_4, 1000)

        time.sleep(.1)

        arm_clamp_block(0)

        arm_move_6(look_at, 1000)

# knock over building blocks
```



```

def push_over_block():

    arm_move_6(p_push_over_1, 1000)

    time.sleep(.2)

    arm_move_6(p_push_over_2, 1000)

    time.sleep(.1)

    arm_move_6(look_at, 1000)

    time.sleep(1)

    global g_layer

    g_layer = 0

```

```

#Let the robotic arm move to the position where the camera looks forward

arm_move_6(look_at, 1000)

time.sleep(1)

```

```

global g_state_arm

g_state_arm = 0

global g_layer

g_layer = 0

def ctrl_arm_move(index):

    global g_layer

    g_layer = g_layer + 1

    if g_layer >= 5:

        g_layer = 1

    arm_clamp_block(0)

    if index == 1:

        number_action(index)

        put_down_block(g_layer)

    elif index == 2:

        number_action(index)

```

```

        put_down_block(g_layer)

    elif index == 3:

        number_action(index)

        put_down_block(g_layer)

    elif index == 4:

        number_action(index)

        put_down_block(g_layer)

    elif index == 5:

        time.sleep(1)

        push_over_block()

global g_state_arm

g_state_arm = 0

```

```

def start_move_arm(index):
    # Open the robot arm control thread
    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

```

```

try:

    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 True:

    """1.Photograph    """

    ret, frame = g_camera.read()

    """ 2.Invoking gesture recognition    """

    raw = str(client.gesture(image_widget.value))

    text = demjson.decode(raw)

    try:

        res = text['result'][0]['classname']

    except:

        #          print('Recognition result: Nothing was recognized~' )

        #          img = cv2ImgAddText(frame, "Not recognized", 250, 30, (0, 0 ,
255), 30)

        img = frame

    else:

        #          print('Recognition result: ' + hand[res])

        #          img = cv2ImgAddText(frame, hand[res], 250, 30, (0, 255 , 0), 30)

        if res == 'One':

            Count_One = Count_One + 1

            Count_Two = 0

            Count_Three = 0

            Count_Four = 0

            Count_Fist = 0

            if Count_One >= 3:

```

```

        print('Recognition results: ' + hand[res])

        img = cv2ImgAddText(frame, hand[res], 250, 30, (0, 255 ,
0), 30)

        if yellow_grabbed == 0:

            start_move_arm(1)

            #                global yellow_grabbed

            yellow_grabbed = 1

elif res == 'Two':

    Count_Two = Count_Two + 1

    Count_Three = 0

    Count_Four = 0

    Count_Fist = 0

    if Count_Two >= 3:

        print('Recognition results: ' + hand[res])

        img = cv2ImgAddText(frame, hand[res], 250, 30, (0, 255 ,
0), 30)

        if red_grabbed == 0:

            start_move_arm(2)

            #                global red_grabbed

            red_grabbed = 1

elif res == 'Three':

    Count_Three = Count_Three + 1

    Count_Two = 0

    Count_Four = 0

    Count_Fist = 0

    if Count_Three >= 3:

        print('Recognition results: ' + hand[res])

        img = cv2ImgAddText(frame, hand[res], 250, 30, (0, 255 ,
0), 30)

```

```

        if green_grabbed == 0:

            start_move_arm(3)

#                global green_grabbed

            green_grabbed = 1

elif res == 'Four':

    Count_Four = Count_Four + 1

    Count_Two = 0

    Count_Three = 0

    Count_Fist = 0

    if Count_Four >= 3:

        print('Recognition results: ' + hand[res])

        img = cv2ImgAddText(frame, hand[res], 250, 30, (0, 255 ,
0), 30)

        if blue_grabbed == 0:

            start_move_arm(4)

#                global blue_grabbed

            blue_grabbed = 1

elif res == 'Fist': #fist

    Count_Fist = Count_Fist + 1

    Count_One = 0

    Count_Two = 0

    Count_Three = 0

    Count_Four = 0

    if Count_Fist >= 3:

        print('Recognition results: ' + hand[res])

        img = cv2ImgAddText(frame, hand[res], 250, 30, (0, 255 ,
0), 30)

        start_move_arm(5)

```

```

        yellow_grabbed = 0

        red_grabbed = 0

        green_grabbed = 0

        blue_grabbed = 0

        Count_Fist = 0
    else:
        img = frame

        image_widget.value = bgr8_to_jpeg(img)

except KeyboardInterrupt:

    print(" Program closed! ")

    pass

```

After running the above program, if the set gesture action is recognized, the robotic arm will perform the corresponding action.

When robotic arm recognizes a number for the first time, it picks up the building blocks and places them on the first layer.

When robotic arm recognizes a number for the second time, it picks up the building blocks and places them on the second layer.

When robotic arm recognizes a number for the third time, it picks up the building blocks and places them on the third layer.

When robotic arm recognizes a number for the fourth time, pick up the building blocks and place them on the fourth layer;

The color of the blocks picked up by the robotic arm each time is determined by the number recognized.

The number 1 represents yellow, number 2 represents red, number 3 represents green, number 4 represents yellow.

If the same number is recognized again, the robotic arm will not move.

If a fist is detected, the robotic arm knocks down all the blocks, clears the record, and starts over.