PythonSDK-V2 user's manual

Version: V20190227

Table of Contents

1. Introduction

To use the Python SDK, you must have a Gomer robot and a computer with WIFI.You need to connect the computer to the Gomer robot via WIFI.

This document introduces the detailed use of the API for PythonSDK-V2.

2. API specification

All interfaces return tuples of type: (status, value). The 'status' means the calling state and 0 means success, -1 means failure.If it's 0, 'value' is the actual return value.

2.1. Robot initialization

2.1.1. Robot(device_name)

Initializes a robot object. All the subsequent interface operations are on this object.

device_name : Robot ID, which you can see in the face of the robot
return : Returns a robot object

Example: robot_obj = robot.Robot('Gomer_ABC123')
Note: Create a robot object named 'Gomer_ABC123'.

2.1.2. connect()

Connect to robot via WIFI.

return : (status, value),'statu' : 0 means success, -1 means failure. 'value' : None

Example: result = robot.connect()
Note: Connect the robot, and you will know if it succeeded from the return value.

2.2. misc module

Misc contains the control of the light.

2.2.1. set_color(red, blue, green)

With this interface, you can configure the color of the light, by adjusting the value of the three primary colors, to achieve any color of the light.
Note: You need to turn on the light by function "light_on".

red/blue/green : the value of the color, limits 0~100
return : (status, value),'statu' : 0 means success, -1 means failure. 'value' : None

Example: result = misc.set_color(50, 50, 50)
Note: Set the color value to orange.

2.2.2. set_blink(mode, duration, light_time, dark_time)

Configure how lights blink.
Note: You need to turn on the light by function "light_on".

mode : 0=keep lighting; 1=blinking; 2=breathing.
duration : Total duration, range 0~30000 milliseconds, when set to 65535, is unlimited time
light_time : When the blink mode is valid, within a period, time of light on, the range 0~10000 milliseconds.
dark_time : When the blink mode is valid, within a period, time of light off, the range 0~10000 milliseconds.
return : (status, value),'statu' : 0 means success, -1 means failure. 'value' : None

Example: result = misc.set_blink(1, 20000, 500, 1500)
Note: Set the light to blink mode for 20 seconds, one period of 2000 milliseconds, light 500 milliseconds, and extinguish 1500 milliseconds.

2.2.3. light_on()

Turn on the light.
Tips: Please set the parameters of the light before turning on the light, otherwise it is invalid.

2.2.4. light_off()

Turn off the light.

2.3. motion module

The motion module is including motion control interface, both open-loop control and closed-loop control, to meet different needs. Closed-loop control, through motion algorithm, precise control of mechanical movement to the specified position.

2.3.1. wheels_move(speed, direction, hold_time)

Wheel movement, open-loop mode, with time as the stop signal.

speed : Motion speed, range 0~3, 0 for stop.
direction : Motion direction, range -180~180, 0 degrees straight ahead.
hold_time : Exercise time, range 0~10000 milliseconds, when time runs out, the motion stops.
return : (status, value),'statu' : 0 means success, -1 means failure. 'value' : None

Example: result = motion.wheels_move(3, 90, 5000)
Note: Move 5000 milliseconds to the right 90 degrees at 3 gear speed.

2.3.2. wheels_accurate_go_straight(speed, distance, wait_time)

Wheel straight line, closed-loop control, accurate to a certain position.

speed : Motion speed, range 0~3, 0 for stop.
distance : Motion distance, range -1000~1000 mm, positive number for forward, negative number for back.
wait_time: Wait for timeout, range 0~10000 milliseconds. When the movement reaches the position, it returns immediately, and when the time runs out, it returns automatically.
return : (status, value),'statu' : 0 means success, -1 means failure. 'value' : None

Example: result = motion.wheels_accurate_go_straight(3, 200, 3000)
Note: Straight forward 200 mm at 3 gears for up to 3000 milliseconds

2.3.3. wheels_accurate_turn(speed, angle, wait_time)

Wheel turn, closed-loop control, precise rotation X angle.

speed : Motion speed, range 0~3, 0 for stop.
angle : Turn angle, range -360~360 mm, positive number for right turn, negative number for left.
wait_time : Wait for timeout, range 0~10000 milliseconds. When the movement reaches the position, it returns immediately, and when the time runs out, it returns automatically.
return : (status, value),'statu' : 0 means success, -1 means failure. 'value' : None

Example: result = motion.wheels_accurate_turn(3, 200, 3000)
Note: Turn right 200 degrees at 3 gear and wait up to 3000 milliseconds.

2.3.4. forearm_rotate(speed, hold_time)

Small arm movement, open-loop mode, with time as the stop signal.

speed: Motion speed, range -3~3, 0 for stop, positive number for outward stretching, negative number for inward shrinkage.
hold_time : Exercise time, range 0~10000 milliseconds, when time runs out, the motion stops.
return : (status, value),'statu' : 0 means success, -1 means failure. 'value' : None

Example: result = motion.forearm_rotate(3, 3000)
Note: The forearm stretches outward at a speed of 3, lasting 3000 milliseconds.

2.3.5. forearm_accurate_rotate(speed, angle, wait_time)

Small arm movement, closed-loop control, accurate rotation of X angle.

speed : Motion speed, range 0~3, 0 for stop.
angle : Target absolute angle, range 0~210, 0 degrees for full shrinkage position, 210 degrees for full stretch position
wait_time : Wait for timeout, range 0~10000 milliseconds. When the movement reaches the position, it returns immediately, and when the time runs out, it returns automatically.
return : (status, value),'statu' : 0 means success, -1 means failure. 'value' : None

Example: result = motion.forearm_accurate_rotate(3, 90, 3000)
Note: The forearm rotates to a 90-degree position at 3 gear, with a maximum wait of 3000 milliseconds

2.3.6. arm_rotate(speed, hold_time)

Arm movement, open-loop mode, with time as the stop signal.

speed : Motion speed, range -3~3, 0 for stop, positive number for backward shrinkage, negative number for forward stretching.
hold_time : Exercise time, range 0~10000 milliseconds, when time runs out, the motion stops.
return : (status, value),'statu' : 0 means success, -1 means failure. 'value' : None

Example: result = motion.arm_rotate(3, 3000)
Note: The arm stretches forward at a speed of 3, lasting 3000 milliseconds.

2.3.7. arm_accurate_rotate(speed, angle, wait_time)

Arm movement, closed-loop control, precise rotation of X angle.

speed : Motion speed, range -3~3, 0 for stop.
angle : Target absolute angle, range 0~160, 0 degrees for full shrinkage position, 160 degrees for full stretch position

wait_time : Wait for timeout, range 0~10000 milliseconds. When the movement reaches the position, it returns immediately, and when the time runs out, it returns automatically.
return : (status, value),'statu' : 0 means success, -1 means failure. 'value' : None

Example: result = motion.arm_accurate_rotate(3, 90, 3000)
Note: The arm rotates to a 90-degree position at 3 gear, with a maximum wait of 3000 milliseconds

2.3.8. head_rotate(speed, hold_time)

Head movement, open-loop mode, with time as the stop signal.

speed : Motion speed, range -3~3, 0 for stop, positive number for upward rise, negative number for down bow.
hold_time : Exercise time, range 0~10000 milliseconds, when time runs out, the motion stops.
return : (status, value),'statu' : 0 means success, -1 means failure. 'value' : None

Example: result = motion.head_rotate(3, 3000)
Note: The head stretches up at a speed of 3, lasting 3000 milliseconds.

2.3.9. head_accurate_rotate(speed, angle, wait_time)

Head movement, closed-loop control, precise rotation X angle.

speed : Motion speed, range -3~3, 0 for stop.
angle : The target absolute angle, the range-15~65,-15 degrees for the bow limit position, 0 degrees for facing the front, 65 degrees for the rise limit position.
wait_time : Wait for timeout, range 0~10000 milliseconds. When the movement reaches the position, it returns immediately, and when the time runs out, it returns automatically.
return : (status, value),'statu' : 0 means success, -1 means failure. 'value' : None

Example: result = motion.head_accurate_rotate(3, 40, 3000)
Note: The head rotates to a 40-degree position at 3 gear, with a maximum wait of 3000 milliseconds

2.3.10. paw_rotate(speed, hold_time)

Paw movement, open-loop mode, with time as the stop signal.

speed : Motion speed, range -3~3, 0 for stop, positive number for inward closure, negative number for outward opening.
hold_time : Exercise time, range 0~10000 milliseconds, when time runs out, the motion stops.
return : (status, value),'statu' : 0 means success, -1 means failure. 'value' : None

Example: result = motion.paw_rotate(3, 3000)
Note: The paw are closed inward at a speed of 3, lasting 3000 milliseconds

2.3.11. paw_accurate_rotate(speed, angle, wait_time)

Paw movement, closed-loop control, precise rotation X angle.

speed : Motion speed, range -3~3, 0 for stop.

angle : The target absolute angle, the range 0~100 度, 0 degrees for the open limit position, 100 degrees for the closed limit position.
wait_time : Wait for timeout, range 0~10000 milliseconds. When the movement reaches the position, it returns immediately, and when the time runs out, it returns automatically.
return : (status, value),'statu' : 0 means success, -1 means failure. 'value' : None

Example: result = motion.paw_accurate_rotate(3, 100, 3000)
Note: The paw are closed to 100 degrees at 3 gear and wait up to 3000 milliseconds

2.4. audio module

The audio module is including voice-related interfaces, TTS, music production, voice crack, playback, stop and other functions.

2.4.1. make_tts(text, file_name)

TTS synthesizer and save as a file.
text : Text information, less than 128 English characters
file_name : Save the file name, the file will be saved in the Gomer's brain, you can crack it, or play.
return : (status, value),'statu' : 0 means success, -1 means failure. 'value' : None

Example: result = audio.make_tts("Hello world!", "first_file")
Note: Convert text "Hello World" to audio and save an audio file named "first_file"

2.4.2. make_music(speed, music, file_name)

Synthesizing music and saved as a file.

speed: Beat speed, range 1~8
music: Music Simple spectrum information, limit 128 digits.
file_name : Save the file name, the file will be saved in the Gomer's brain, you can crack it, or play.

return : (status, value),'statu' : 0 means success, -1 means failure. 'value' : None

Example: result = audio.make_music(2, "517334245151", "first_file")
Note: Convert Simple Spectrum "517334245151" to audio with a beat speed of 2 and save an audio file named "first_file"

2.4.3. convert_audio(src_file, convert_type, dst_file)

Audio crack, the source audio according to a certain type of voice to the target audio.

src_file : source audio file.
convert_type : type, range 1~8
dst_file : Save the file name, the file will be saved in the Gomer's brain, Can be used for other purposes.
return : (status, value),'statu' : 0 means success, -1 means failure. 'value' : None

Example: result = audio.convert_audio("src_audio", 2, "convert_file")
Note: Cracking the audio file to voice with type 2, save an audio file named "convert_file"

2.4.4. get_volume()

Get current volume.

return : (status, value),'statu' : 0 means success, -1 means failure. 'value' : When there is a return value, the type is list, otherwise it is none.

Example:

```
ret = audio.get_volume()
if ret[0] < 0:
    logger.info("get volume failed!")
else:
    logger.info("volume is:", ret[1][0][0])
```

Note: Gets the volume, if it fails, the prompt fails, and if successful, displays the volume.

2.4.5. set_volume(vol)

Set the volume.

vol : number of the volume, range 0~5
return : (status, value),'statu' : 0 means success, -1 means failure. 'value' : When there is a return value, the type is list, otherwise it is none.

Example:

```
ret = audio.set_volume(3)
if ret[0] < 0:
    logger.info("set volume failed!")
else:
    logger.info("volume is:", ret[1][0][0])
```

Note: Set the volume to 3, if it fails, the prompt fails, and if successful, displays the current volume

2.4.6. get_player_status()

Gets the current playback state.

return : (status, value),'statu' : 0 means success, -1 means failure. 'value' : When there is a return value, the type is list, otherwise it is none.
state is idle : value is "[(0, None)]"
state is busy : value is "[(1, file_name)]"

Example:

```
ret = audio.get_player_status()
if ret[0] < 0:
    logger.info("get status failed!")
else:
    file = ret[1][0][1]
    logger.info("now is playing:", file)
```

Note: Gets the playback status, if it fails, the prompt fails, and if successful, displays the file being played

2.4.7. play_audio(audio_file)

Playback of audio file.

audio_file : the audio file you want to play.
return : (status, value),'statu' : 0 means success, -1 means failure. 'value' : None

2.4.8. stop_player()

Stop the audio player.

return : (status, value),'statu' : 0 means success, -1 means failure. 'value' : None

2.5. screen module

This chapter describes the screen display interface, which provides a variety of display methods, including user-defined pattern display.

2.5.1. show_picture(img_file, hold_time)

Displays the user's picture on the screen.

img_file : Picture path on PC, picture must be 320x240 resolution JPG format.
hold_time: Display time, range 1000~10000 milliseconds.
return : (status, value),'statu' : 0 means success, -1 means failure. 'value' :None.

Example:

```
ret = screen.show_picture("C:\sdk\gomer\my_screen.jpg", 3000)
```

Note: Display the user's picture file ("my_screen.jpg") on the screen for 3000 milliseconds

2.5.2. show_status()

Get the status of the display

return : (status, value),'statu' : 0 means success, -1 means failure. 'value' : When there is a return value, the type is list, otherwise it is none.
state is idle : value is "[(0, None)]"
state is busy : value is "[(1, file_name)]"

Example:

```
ret = screen.show_status()
is_busy = ret[1][0][0]
if is_busy > 0:
    logger.info("screen is busy")
    current_img = ret[1][0][0]
    logger.info(current_img)
else:
    logger.info("screen is idle")
```

Note: Gets the display status, if busy, print the picture name that is being displayed, and if idle, prompts for idle status

2.5.3. mirror_camera()

Mirror the camera's image onto the screen, which will remain open until the user shuts down.

return : (status, value),'statu' : 0 means success, -1 means failure. 'value' : None

2.5.4. stop_mirror_camera()

Stop mirror the image of camera.

return : (status, value),'statu' : 0 means success, -1 means failure. 'value' : None

2.5.5. stop_showing()

Stop all display of the screen.

return : (status, value),'statu' : 0 means success, -1 means failure. 'value' : None

2.6. emotion module

This chapter introduces emotional related interfaces and provides a variety of display emotional interfaces, including interfaces of editing user-defined emotions.

2.6.1. emotion_show(name, expression_switch, voice_switch, action_switch, paw_safe)

Emotional display, and the freedom to control the switch of each element.

name : The names of emotional and emotional classes, such as emotions "wonderful", or "happy", will random a emotion in the emotional class of happy.
expression_switch : expressions control.
voice_switch : voice control
action_switch : motion control
paw_safe : paw protection. If "action_switch" is true, the parameter is valid.When this parameter is enabled, the paw action is prohibited.

return : (status, value),'statu' : 0 means success, -1 means failure. 'value' : None

Example:

```
ret = emotion.emotion_show("wonderful", True, False, True, False)
```

Note: Display wonderful emotions and turn off sound, allowing paw movements.

2.6.2. emotion_status()

Get the emotions you are currently showing.

return : (status, value),'statu' : 0 means success, -1 means failure. 'value' : When there is a return

value, the type is list, otherwise it is none.
state is idle : value is "[(0, None)]"
state is busy : value is "[(1, emotion_name)]"

Example:

```
ret = emotion.emotion_status()
is_busy = ret[1][0][0]
if is_busy > 0:
    logger.info("emotion is busy")
    emotion_name = ret[1][0][0]
    logger.info(emotion_name)
else:
    logger.info("emotion is idle")
```

Note: Get emotional state. If you are showing emotions, then print the emotions name, and if it's idle, prompt for free

2.6.3. stop_emotion()

Stop displaying the emotional

return : (status, value),'statu' : 0 means success, -1 means failure. 'value' : None

2.7. event module

This chapter describes the event messages, the event message is automatically feedback from the robot, and users can handle the event themselves.

2.7.1. open_manager()

Starting event Manager, the listening thread will be started, and users can use the registration interface to add listening events.

return : (status, value),'statu' : 0 means success, -1 means failure. 'value' : None

2.7.2. close_manager()

Closing event manager, the listening thread will be closed, and no longer responds to events.

return : (status, value),'statu' : 0 means success, -1 means failure. 'value' : None

2.7.3. bind_event_callback(event_type, callback_func)

Registers an event to the event manager and binds the event callback function.

event_type:
- button_click
- button_click_twice
- very_low_power
- low_power
- normal_power
- paw_issue
- forearm_issue
- arm_issue
- head_issue
- foot_issue
- posture_change

callback_func : the event callback function
return : (status, value),'statu' : 0 means success, -1 means failure. 'value' : None.

Example:

ret = event.bind_event_callback("button_click", button_click_func)

Note: Register button Click event, bind event handler function named "button_click_func".

2.8. face module

This chapter introduces the face recognition interface, the many face algorithm data is opened to the user, can use the data to carry on the operation again and creates the new idea.

2.8.1. detect_face(image_type, min_face_size, max_face_size, min_face_score, display)

Perform a face recognition, you can choose a data source, such as from a local picture, camera, you can set the face filter standards. Information such as the recognized face coordinates will be returned.

image_type : Select the data source, default camera "camera", image array "Image_array", image file "file".
min_face_size : Minimum face size, range 10~300, default 10
max_face_size : Maximum face size, range 10~300, default 300
min_face_score : Minimum face score, range 0~100, default 10, which is a score that scores the integrity and clarity of the face.

display : Display the picture of this test on the screen.

return : (status, value),'statu' : 0 means success, -1 means failure. 'value' : When there is a return value, the type is list, otherwise it is none.

Return value format : (0, [{"face0":[x,y,w,h,score]}, {"face1":[x,y,w,h,score]}])
Return value note : status = 0
value = [{"face0":[x,y,w,h,score]}, {"face1":[x,y,w,h,score]}]

There are one or more dictionaries in List, a face has 5 parameters: starting point coordinates x, y, wide w, h, face score value score.

Example:

```
ret = face.detect_face("camera", 30, 300, 10, True)
x = ret[1][0]["face0"][0]
y = ret[1][0]["face0"][1]
width = ret[1][0]["face0"][2]
height = ret[1][0]["face0"][3]
score = ret[1][0]["face0"][4]
logger.info("face0 :", x, y, width, height, score)
```

Note: Recognition of the face from the camera, set the face size range 30-300, face score minimum of 10, and display the image on the screen, get the face of the parameter value and print.

2.8.2. get_face_feature(face_index, display)

Gets the characteristic points of the face, such as the coordinates of the eyes, nose, eyebrows, and mouth, this interface depend on the result of function "face.detect_face".

face_index: Select the face index name, such as "Face0".
display : Display the picture of this test on the screen.
return : (status, value),'statu' : 0 means success, -1 means failure. 'value' : When there is a return value, the type is list, otherwise it is none.

Return value format: (0, [{"point0":[lx,ly,rx,ry,nx,ny,lmx,lmy,rmx,rmy]}])
Return value note: status = 0
value = [{"point0":[lx,ly,rx,ry,nx,ny,lmx,lmy,rmx,rmy]}]

There are one or more dictionaries in List, a face feature has 10 parameters: Left eye coordinates lx, ly, right eye coordinate rx, ry, Nose coordinate nc, ny, left corner coordinates lmx, lmy, right corner coordinates rmx, rmy.

Example:

```
ret = face.get_face_feature("face0", True)
left_eye_x = ret[1][0]["point0"][0]
left_eye_y = ret[1][0]["point0"][1]
```

```
right_eye_x = ret[1][0]["point0"][2]
right_eye_y = ret[1][0]["point0"][3]
nose_x = ret[1][0]["point0"][4]
nose_y = ret[1][0]["point0"][5]
left_mouth_x = ret[1][0]["point0"][6]
left_mouth_y = ret[1][0]["point0"][7]
right_mouth_x = ret[1][0]["point0"][8]
right_mouth_y = ret[1][0]["point0"][9]
```

Note: Gets the feature points of the face "face0" and displays the image on the screen.

2.8.3. face_register(face_index, face_name)

Naming the face and adding it into the database, this interface depend on the result of function "face.detect_face".

face_index: The face index name, such as "face0".
face_name: The name of the face.
return : (status, value),'statu' : 0 means success, -1 means failure. 'value' : None

Example:

```
ret = face.face_register("face0", "Tom")
```

Note: Add the face "face0" into the robot database and name it Tom.

2.8.4. identity_recognize(face_index, display)

Compare the face with the robot's database, identify the identity, this interface depend on the result of function "face.detect_face".

face_index: The face index name, such as "face0".
display : Display the picture of this test on the screen.
return : (status, value),'statu' : 0 means success, -1 means failure. 'value' : When there is a return value, the type is list, otherwise it is none. The return value maybe "unkown" or name.

Example:

```
ret = face.identity_recognize("face0", True)
if ret[0] < 0:
    logger.info("recognize face failed")
else:
    name = ret[1][0]
    logger.info("recognize face :", name)
```

Note: Compare the face "face0" with the robot database and return the result to Tom.

2.8.5. expression_recognize(face_index, display)

To recognize the expression of a human face, this interface depend on the result of function "face.detect_face".

face_index: The face index name, such as "face0".
display : Display the picture of this test on the screen.
return : (status, value),'statu' : 0 means success, -1 means failure. 'value' : When there is a return value, the type is list, otherwise it is none.

Example:

```
ret = face.expression_recognize("face0", True)
if ret[0] < 0:
    logger.info("recognize face failed")
else:
    expression = ret[1][0]
    logger.info("recognize face expression:", expression)
```

Note: Identify the facial "face0" expression and return the result happy.

2.8.6. get_face_identity(face_index, identity_name, display)

Get identity feature data for faces, this interface depend on the result of function "face.detect_face".

face_index: The face index name, such as "face0".
identity_name: Name the identity data and save it as a temporary file (in the robot's brain).
display : Display the picture of this test on the screen.
return : (status, value),'statu' : 0 means success, -1 means failure. 'value' : None

Example:

```
ret = face.get_face_identity ("face0", "id_tom", True)
```

Note: Gets the identity feature data for the face "face0", saves it as a file named "id_tom", and displays the image on the screen.

2.8.7. get_face_similarity(identity_name1, identity_name2)

Compared to the identity characteristic data of two faces, this interface depend on the result of function "face.detect_face".

identity_name1: The identity characteristic data file name, it's the result of the function "face.get_face_identity".

identity_name2: The identity characteristic data file name, it's the result of the function "face.get_face_identity".
return: (status, value),'statu' : 0 means success, -1 means failure. 'value' : When there is a return value, the type is list, otherwise it is none. Returns the similarity, the range 0~100.

Example:

```
ret = face.get_face_similarity("id_sam", "id_tom")
if ret[0] < 0:
    logger.info("compare faces failed")
else:
    similarity = ret[1][0]
    logger.info("face similarity:", similarity)
```

Note: Camparing the identity data of the face Id_sam and Id_tom. The similarity is 80.

2.8.8. face_rename(old_name, new_name)

Rename the face and not be the same as the name of the person in the database.
old_name: The old face name.
new_name: The new face name.
return: (status, value),'statu' : 0 means success, -1 means failure. 'value' : None

Example:

```
ret = face.face_rename("tom", "sam")
```

Note: Rename the face Tom in the database to Sam.

2.8.9. get_face_list()

Gets all the faces in the robot database.

return: (status, value),'statu' : 0 means success, -1 means failure. 'value' : When there is a return value, the type is list, otherwise it is none.

Return value format: (0, [{"list":["tom","tom"]}])
Return value note: status = 0
value = [{"list":["tom","tom"]}]
There is a dictionary in List, include all face names.

Example:

```
ret = face.get_face_list()
name1 = ret[1][0]["list"][0]
name2 = ret[1][0]["list"][1]
```

Note: Gets the list of faces in the database and gets back "Tom", "Sam".

2.9. pattern module

This chapter introduces the pattern recognition interface and opens many pattern algorithm data to the user.

2.9.1. detect_pattern(image_type, min_size, max_size, min_score, display)

Perform a pattern recognition, you can select a data source, such as from a local picture, camera, you can set the pattern filter standard. The recognized pattern information will be returned.

image_type : Select the data source, default camera "camera", image array "Image_array", image file "file".
min_size : Minimum pattern size, range 30~300, default 30
max _size : Maximum pattern size, range 30~300, default 300
min_score : Minimum pattern score, range 0~100, default 10, which is a score that scores the integrity and clarity of the pattern.
display : Display the picture of this test on the screen.
return : (status, value),'statu' : 0 means success, -1 means failure. 'value' : When there is a return value, the type is list, otherwise it is none.

Return value format : (0, [{"pattern0":[lux,luy,ldx,ldy,rdx,rdy,rux,ruy,score]}])
Return value note : status = 0
value = [{"pattern0":[lux,luy,ldx,ldy,rdx,rdy,rux,ruy,score]}]
There are one or more dictionaries in List, a pattern has 9 parameters: top left x, y, bottom left x, y, bottom right x, y, top right x, y, pattern score.

Example:

```
ret = pattern.detect_face("camera", 30, 300, 10, True)
left_up_x = ret[1][0]["pattern0"][0]
left_up_y = ret[1][0]["pattern0"][1]
left_down_x = ret[1][0]["pattern0"][2]
left_down_y = ret[1][0]["pattern0"][3]
right_down_x = ret[1][0]["pattern0"][4]
right_down_y = ret[1][0]["pattern0"][5]
right_up_x = ret[1][0]["pattern0"][6]
right_up_y = ret[1][0]["pattern0"][7]
pattern_score = ret[1][0]["pattern0"][8]
```

Note: Identify a pattern from the camera, set the pattern size range 30-300, the pattern score minimum of 10, and display the image.

2.9.2. add_pattern(pattern_path, width, hight)

Add a pattern from the PC.

pattern_path: Absolute path of image files, pattern name should be less than 16 characters.
width: The true width of pattern, in centimeters
hight: The true height of pattern, in centimeters
return : (status, value),'statu' : 0 means success, -1 means failure. 'value' : None

Example:

```
ret = pattern.add_pattern("C:\sdk\demo\pattern.jpg", 30, 30)
```

Note: Add the pattern "pattern.jpg" into the robot database and set the pattern size 30x30.

2.9.3. get_3d_position(pattern_index, display)

Gets the 3D gesture of the pattern in the current screen, this interface depend on the result of function "pattern.detect_pattern".

pattern_index: The index name of the pattern.
display: Display the picture of this test on the screen.
return : (status, value),'statu' : 0 means success, -1 means failure. 'value' : When there is a return value, the type is list, otherwise it is none.

Return value format : (0, [{"pattern0":[x,y,z,theta_x,theta_y,theta_z]}])
Return value note : status = 0
value = [{"pattern0":[x,y,z,theta_x,theta_y,theta_z]}]
There is a dictionary in List, a pattern has 6 parameters: Spatial coordinates x, y, z, deflection values theta_x, theta_y, theta_z.

Example:

```
ret = pattern.get_3d_position("pattern0", True)
x = ret[1][0]["pattern0"][0]
y = ret[1][0]["pattern0"][1]
z = ret[1][0]["pattern0"][2]
theta_x = ret[1][0]["pattern0"][3]
theta_y = ret[1][0]["pattern0"][4]
theta_z = ret[1][0]["pattern0"][5]
```

Note: Gets the 3D coordinates of the pattern "pattern0" and displays the screen.

2.9.4. move_to_pattern(pattern_name, distance)

Moving to the front of the pattern and keeping X distance, the interface will return motion parameters, which are controlled by the user to reach the specified position.

pattern_name: The name of the pattern.It can be set to G, L, I.

distance: The target distance from the pattern, in centimeters。

Return value format : (0, [prm1, prm2, prm3, prm4, prm5])
Return value note : status = 0
value = [prm1, prm2, prm3, prm4, prm5]
There are 5 parameters in the list:

"prm1" represents whether the position has been reached. If the value is 101, it means the position has been reached. Otherwise, it is not in place

"prm2" represents the position where the head still needs to move, and a value of 255 indicates that the head is in place

"prm3" represents the position where the wheel needs to turn for the first time. If the value is 0, it indicates that the wheel has turned in place

"prm4" represents the position where the wheel still needs to go straight. If the value is 0, it means that the line is in place

"prm5" represents the position where the wheel still needs to turn the second time. If the value is 0, it indicates that the steering is in place

Example:

```
while True:
    # Detect or not of Pattern
    value = pattern.move_to_pattern("G", 9)
    if value[0] == 0:
        if value[1][0] == 101:
            break
        if value[1][1] != 255:
            motion.head_accurate_rotate(2, value[1][1], 3000)
        if value[1][2] != 0:
            motion.wheels_accurate_turn(1, value[1][2], 3000)
        if value[1][3]  != 0:
            motion.wheels_accurate_go_straight(2, value[1][3], 3000)
        if value[1][4] != 0:
            motion.wheels_accurate_turn(2, value[1][4], 3000)
```

Note: Moving to the front of pattern G and keeping a distance of 9 cm will return to the movement path until the target position is reached.

3. Example Programs

3.1. hello_world.py

```
from sdk.gomer import robot


def say_hello_world():
    robot_obj = robot.Robot('GomerAY1773')
```

```
    robot_obj.connect()
    valume = robot_obj.audio.get_volume()

    print('当前音量大小为：', valume)

    if valume[1][0] < 3:
        robot_obj.audio.set_volume(1)
        robot_obj.audio.make_tts('hello world','hello')
    for i in range(10):
        robot_obj.audio.play_audio('hello')


if __name__ == "__main__":
    say_hello_world()
```

1. Firstly, import robot module
2. Create a robot object and specify the robot ID. The SDK will initialize a robot control system.
3. Start to connect the robot. The SDK will search and connect the specified robot.
4. If the connection is successful, you can create any Python program through the API in the document.
5, in the example, get the robot volume, if the volume is less than 3, will use "make_tts" to generate a "helloworld" voice, through the play voice interface "play_audio", you can hear "helloworld" voice.

4. Copyright notice

5. Other

version	reviser	revision record
V20190131	Kevin	Create an initial version
V20190225	Kevin	Add detailed sample code