3D Camera Face Detection and Pan-Tilt Tracking

I. Experimental Objectives

1. Understanding and mastering image processing: Through the experiment, learn how to use the OpenCV library for image capture, processing, and face detection.

2. Mastering serial communication: Through the experiment, understand how to use serial communication to control the movement of hardware devices (such as pan-tilt mechanisms).

3. Practicing the basic principles of computer vision: Through face detection practice, understand the implementation of computer vision in practical applications.

4.Integrating software and hardware control: Learn how to combine software image processing with hardware control to complete the experiment of automatic tracking and pan-tilt control systems.

II. Experimental Content

3D camera face detection and pan-tilt tracking is achieved by using the ORBBECDaBai depth camera to capture images and detect faces, controlling the pan-tilt camera to move with the movement of the face.

Experimental business process:

1.Camera image capture: Use the Orrbec camera to capture real-time color image data.

2.Image processing and display: Use the OpenCV library to process the captured images and display the processed results on the screen.

3.Face detection: Use the Haar cascade classifier to detect faces in the processed images and draw rectangular frames in the detected face areas.

4.Pan-tilt control: Adjust the angle of the pan-tilt according to the position of the face in the image to achieve automatic tracking.

III. Experimental Environment

Experimental Equipment	Artificial Intelligence Experiment Box
Operating System	Linux
Experimental Accessories	Depth Camera

IV. Experimental Principles

1.Hardware principles

The hardware of ORBBECDaBai includes two depth cameras, an RGB camera, and a structured light infrared projector. Depth shutter camera (scanning line by line), infrared structured light distance measurement.

2.Algorithm principles

This experiment will detect faces in the face images captured by the RGB camera to obtain the face detection box, and at the same time, obtain the center coordinates of the face detection box. By mapping the center coordinates of the face detection box with the angle of the pan-tilt movement, the angle at which the pan-tilt servo should rotate when the face moves left and right can be obtained, and the pan-tilt can track the face and keep it consistent with the face.

3.Related functions

blinx_face_detect_demo()

The usage of the face detection function blinx_face_detect_demo() is as follows:

blinx_face_detect_demo() is an encapsulated function. The implementation process is to read the picture, convert the picture to RGB format and grayscale image, and call the OpenCV's built-in face detection model file "haarcascade_frontalface_default.xml" to get the face detection box, and draw the face detection box.

blinx_bus_stepping_motor()

The control function for the pan-tilt blinx_bus_stepping_motor() is also an encapsulated function. It controls the rotation of the pan-tilt servo by sending a hexadecimal array through the serial port. The array [0xFF,0xFE,0x02,0x01,0x00,0x01,0x3C,0x00,0x0D,0x0A] starts from the 0th bit, and the fifth and sixth bits correspond to the angle value of the servo rotation.

For specific control of the pan-tilt servo, please refer to the related documents in (6. Embedded Systems and Applications/11 Attitude Gyroscope Cloud Platform Control).

The entire experimental principle process is as follows:

1.Image capture and processing:

Use the Orrbec camera to capture real-time video streams and convert the captured frames into BGR formatted images for processing.

The OpenCV library is used for image processing operations, including color space conversion, image display, etc.

2.Face detection:

Use the Haar cascade classifier to detect faces in the images. This classifier is a model trained based on machine learning and can quickly detect face areas in the images.

Draw rectangular frames in the detected face areas to mark the position of the face.

3.Pan-tilt control:

Calculate the offset of the face relative to the center of the image based on the detected face position. Send control commands through the serial port to adjust the pan-tilt angle so that the face is always in the center of the image, achieving automatic tracking.

V. Experimental Steps

1.Run Jupyter Lab

(1) Open the "Experiment" folder on the desktop, right-click on an empty space, and click "Open in Terminal". Enter "jupyter lab" in the terminal interface;

(2) In the Jupyter Lab programming interface, select Python3 under Notebook to enter the program editor;

(3) Right-click on the "Untitled.ipynb" new program block, and select "Rename" to name the program with the experiment name;

(4) Enter "%load 3D Camera Face Detection and Pan-Tilt Tracking.py" in the new cell and click the run button to load the program.

.

2.Importing Library Files and Functions

Import Python's standard libraries and third-party libraries. sys is used for system-related operations, cv2 for image processing, time for controlling time delays, serial for serial communication, pyorbbecsdk for controlling the Orrbec camera, and the frame_to_bgr_image function in utils for converting camera frames to BGR image format.

Reference Code :

Create a new program block "3D Camera Face Detection and Pan-Tilt Tracking.ipynb"

```
import sys    # Import system module for system operations (e.g., exiting the program)
import cv2    # Import OpenCV library for image processing and computer vision tasks
import time    # Import time module for delay or timing operations
import serial    # Import serial communication module for communication with external devices
from pyorbbecsdk import *    # Import the Orrbec SDK for controlling the Orrbec camera
from utils import frame_to_bgr_image    #Import custom utility function to convert camera frames to BGR image format
```

3.Initialize serial port and variables

Set up serial communication to control the rotation of the pan-tilt mechanism. Initialize the value variable to 90 degrees to control the angle of the pan-tilt.

Reference Code :

Continue writing the program block "3D Camera Face Detection and Pan-Tilt Tracking.ipynb".

```
data_ser = serial.Serial("/dev/user_robot", 115200, timeout=5)    # Initialize serial port for controlling the pan-tilt, baud rate set to 115200, timeout set to 5 seconds
value = 90    # Initialize global variable value for controlling the pan-tilt rotation angle, initial value set to 90 degrees
```

4. Camera initialization function

Initialize the camera and configure its parameters, start the camera stream. Load the face detection model and return the camera pipeline and face detector. If an error occurs during initialization, print the error message and exit the program.

Reference Code :

Continue writing the program block "3D Camera Face Detection and Pan-Tilt

Tracking.ipynb".

```
# Camera initialization function
def blinx_CameraInit():
    try:
        pipeline = Pipeline()   # Initialize camera stream pipeline for obtaining image data streams
        config = Config()   # Initialize camera configuration parameters
        profile_list = pipeline.get_stream_profile_list(OBSensorType.COLOR_SENSOR) # Get the stream configuration list for the color sensor
        color_profile = profile_list.get_default_video_stream_profile()   # Get the default video stream configuration
        config.enable_stream(color_profile)   # Enable the camera video stream
        pipeline.start(config)   # Start the camera stream
        face_detector = cv2.CascadeClassifier("haarcascade_frontalface_default.xml") # Load face detection model data
        return pipeline, face_detector   # Return the camera stream pipeline and face detector
    except Exception as e:   #   Catch exceptions
        print("Camera initialization error:", e)   # Print error message
        sys.exit(1)   # Exit the program
```

5.Get image frame function

Obtain color image frames from the camera data stream and convert them to BGR formatted images. If acquisition or conversion fails, return None.

Reference Code :

Continue writing the program block "3D Camera Face Detection and Pan-Tilt Tracking.ipynb".

```
# Get image frame function
def blinx_get_frames(pipeline):
    try:
        frames = pipeline.wait_for_frames(100)   # Wait for camera frames, timeout is 100 milliseconds
        if frames is None:   # If no frames are obtained, return None
            return None
        color_frame = frames.get_color_frame()   # Get color image frame
        if color_frame is None:   # If no color image frame is obtained, return None
            return None
        color_image = frame_to_bgr_image(color_frame)   # Convert color frame to BGR formatted image
        return color_image   # Return the converted image
    except Exception as e:   # Catch exceptions
```

```
        print("Color image capture error:", e)    # Print error message
        return None    # Return None
```

6.Face detection and pan-tilt control function

Identify faces through the face detector, calculate the position of the face in the image, and adjust the pan-tilt angle according to the position to make it follow the face movement.

Reference Code :

Continue writing the program block "3D Camera Face Detection and Pan-Tilt Tracking.ipynb".

```
# Face detection function, receive image and return image with detection box
def blinx_face_detect_demo(image_src, face_detector):
    global value    #Declare global variable
    image = cv2.flip(image_src, 1)    # Flip the image horizontally
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)    #    Convert the image to
grayscale
    faces = face_detector.detectMultiScale(gray, 1.3, 5)    # Detect faces in the image
    for x, y, w, h in faces:    # Iterate through all detected faces
        cv2.rectangle(image, (x, y), (x + w, y + h), (0, 0, 255), 2)    # Draw a red rectangle
on the image
        print("x:", x + w / 2)    # Print the x-coordinate of the face center
        if 0 <= x + w / 2 <= 270:    #    Check if the face is on the left side
            value -= 2    #    Decrease the pan-tilt angle if on the left
            value = max(value, 20)    # Ensure the pan-tilt angle is not lower than 20
degrees
        if 380 <= x + w / 2 <= 640:    # Check if the face is on the right side
            value += 2    # Increase the pan-tilt angle if on the right
            value = min(value, 160)    # Ensure the pan-tilt angle does not exceed 160
degrees
        print("value:", value)    # Print the current pan-tilt angle
    blinx_bus_stepping_motor(value)    # Call the function to control the pan-tilt rotation
    return image    #    Return the image with the detection box
```

7.Control pan-tilt rotation function

Control the rotation of the pan-tilt by sending commands through the serial port.

Reference Code :

Continue writing the program block "3D Camera Face Detection and Pan-Tilt Tracking.ipynb".

```
# Control pan-tilt rotation function
def blinx_bus_stepping_motor(value):
    ddata = [0xFF, 0xFE, 0x02, 0x01, 0x00, 0x01, 0x46, 0x00, 0x0D, 0x0A]    # Initialize
command data packet
    ddata[5] = value    #    Write the pan-tilt angle value into the data packet
    data_ser.write(ddata)    # Send the data packet through the serial port
    time.sleep(0.1)    # Delay 100 milliseconds to ensure the command is sent
```

8.Define the main program

After initializing the camera and face detector, enter a loop to continuously obtain images, detect faces, adjust pan-tilt angles, and display real-time images in the window. Press the 'q' key to exit the program.

Reference Code :

Continue writing the program block "3D Camera Face Detection and Pan-Tilt Tracking.ipynb".

```
# Main program entry
def main():
    pipeline, face_detector = blinx_CameraInit()    # Call the camera initialization function
to get the camera stream and face detector
    global value    # Declare global variable value
    while True:    # Infinite loop to continuously process images
        color_image = blinx_get_frames(pipeline)    # Get image frames
        if color_image is None:    # If no image is obtained, skip this loop iteration
           conti
            continue
        if color_image is None:    # If the image is empty, return None
            return None
        color_image = blinx_face_detect_demo(color_image, face_detector)    #    Call the
face detection function to process the image
        if color_image is None:    # If image processing fails, skip this loop iteration
            continue
        cv2.imshow("Camera View", color_image)    #    Display the image in a window
        key = cv2.waitKey(1) & 0xFF    # Wait for keyboard input and return the key value
        if key == ord('q'):    #    If the 'q' key is pressed, exit the loop
            break
    pipeline.stop()    # Stop the camera stream
    cv2.destroyAllWindows()    # Close all OpenCV windows
```

9.Run the main program

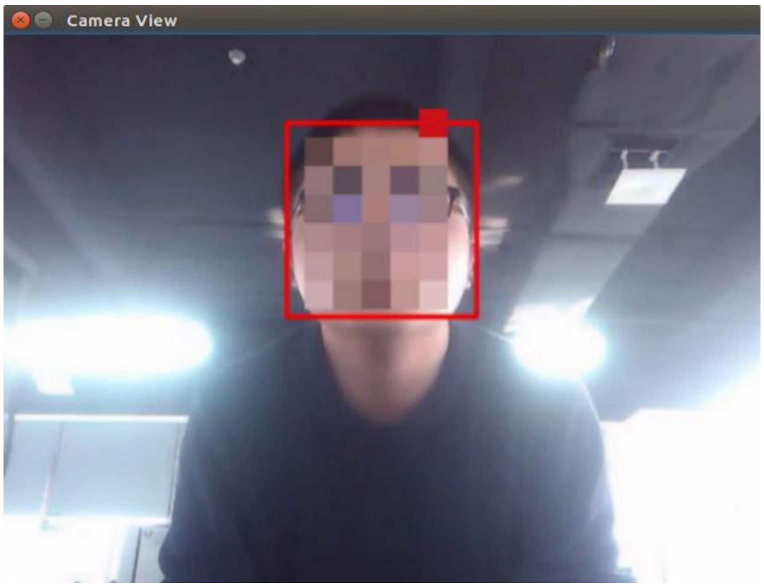
Check if the program is running as the main module, and if so, start the main function main to execute the main process.

Reference Code :

Continue writing the program block "3D Camera Face Detection and Pan-Tilt Tracking.ipynb".

```
if __name__ == "__main__":    # Main program entry
    main()    # Call the main program function
```

10.The effect of running the main program code is shown in the figure below.


Camera View