RaspberryPi 64 bit OS

October 27, 2023

1 Installation of required packages

In 64 bit OS, we can install opencv-python, Flask, ultralytics with normal pip commands.

2 Camera issue

In 64 bit OS, we don't have legacy camera support. You can checkout this link for more information.

3 Alternative camera option

We can use libcamera for camera support. You can checkout this link for more information.

Note:

1. In order to use libcamera first disable legacy camera.
2. 'sudo raspi-config' and
3. Select 'Interfacing Options' and
4. Then 'Camera' and select 'No'.'

4 Libcamera installation

1. sudo apt install build-essential meson ninja libyaml-dev python3-yaml python3-ply python3-jinja2 libssl-dev openssl git
2. git clone https://git.libcamera.org/libcamera/libcamera.git
3. cd libcamera
4. meson build
5. sudo ninja -C build install

5 Check Libcamera

1. libcamera-hello
2. libcamera-still -o test.jpg
3. libcamera-vid -o test.mp4

6 Modified client code

```python
import cv2
import requests
import time
import datetime
import csv
import subprocess
import io
import numpy as np

# Define the TiE-API server URL
server_url = 'http://192.168.20.17:30030/video_feed'

# Create a CSV file to store the data
csv_file = open('data.csv', mode='w', newline='')
csv_writer = csv.writer(csv_file)
csv_writer.writerow(['RTT', 'Processing-Delay', 'Network_Delay',
                     'Server-to-Browser-Delay', 'Total-Frames-Processed',
                      'Total-Frames-Received'])

try:
    print('Connected to Cache Server')
    while True:  # Capture images continuously
        start_time = time.time()

        # Capture an image using libcamera-still with output
        libcamera_command = "libcamera-still -o - -n"
        image_data = subprocess.check_output(libcamera_command,
                      shell=True)

        end_time = time.time()
        rtt = end_time - start_time

        # Convert the image data to a NumPy array
        image = cv2.imdecode(np.frombuffer(image_data, np.uint8),
                    cv2.IMREAD_COLOR)

        # Convert the frame to JPEG format
        _, buffer = cv2.imencode('.jpg', image)
        frame_data = buffer.tobytes()

        headers = {
            'Content-Type': 'application/octet-stream',
            'Frame-Width': str(image.shape[1]),
```

```
            'Frame-Height': str(image.shape[0]),
            'Client-Timestamp': str(time.time())
        }

        response = requests.post(server_url, data=frame_data,
                    headers=headers)

        # Parse the response JSON
        response_data = response.json()

        # Extract the information you want from the response headers
        processing_delay = float(response_data.get(
                                    'Processing-Delay', '0'))
        Network_Delay = rtt - processing_delay
        server_to_browser_delay = response_data.get(
                                    'Server-to-Browser-Delay', '')
        total_frames_processed = response_data.get(
                                    'Total-Frames-Processed', '')
        total_frames_received = response_data.get(
                                    'Total-Frames-Received', '')

        # Print and write the data to the CSV file
        print('Round Trip time (Drone-Server-Drone):', rtt)
        csv_writer.writerow([rtt, processing_delay, Network_Delay,
                    server_to_browser_delay, total_frames_processed,
                    total_frames_received])
        print("Network Delay (rtt-processing):", Network_Delay)
except KeyboardInterrupt:
    pass
except Exception as e:
    print("Error:", str(e))
finally:
    csv_file.close()
```