OSL Python Project

Live Streaming using Python

Abstract:

This project was developed to be deployed on Raspberry Pi, so as to serve as eyes of a smart car. As the main project was in planning stage, this python project intended at streaming the data live from the camera to the Android device which served as a controller

Two Methodologies:

The implementation could be done in two ways:

1. Running two scripts, one for sending image data and other for receiving user commands.
2. Multithreading.

Smart Car Concept:

The real scope of the project lies in giving the car smart features. This would require the use of image processing, audio processing and use of neural networks. The Pi will not be just controlling the motors but also the camera and possibly the audio recorders. Introducing audio is an idea that seems difficult to implement with shall be dealt with towards the end.

Steps in Live Streaming the data:

1. The camera device hosts a server and awaits connection.
2. The controller device connects to it.
3. The camera device then converts images into bytearray and sends it over the socket connection.
4. The Controller device decodes and displays the image.

Steps in Sending Commands:

1. The Receiving device hosts a server and awaits connection.
2. The controller device connects to it.
3. The controller device then send command numbers with a specific mapping that also lies with the receiver.
4. The receiver then maps and decodes the command and acts accordingly.

Using PyGame:

Due to time constraints, motors could not be interfaced. So to display the effects, an image was shifted accordingly on the screen so as to simulate the car moving.

Steps:

1. Initialize the socket and game variables.
2. Whenever command received update the screen
3. Four types: Up,Down,Left,Right.
4. For Up simply change the Y-coordinate by -5.
5. For Down simply change the Y-coordinate by 5.
6. For Left simply change the X-coordinate by -5.
7. For Right simply change the X-coordinate by 5.
8. Update Screen.

Using Multithreading would simply create two threads, one for sending and other for receiving the data. The problem encountered here was the once the image thread started, the command receiver failed to receive data and hence, the two script idea was implemented.