Telepresence Realization with Virtual Reality

- A controlled live video streaming device with hybrid of microcontroller and head mounted display[HMD]

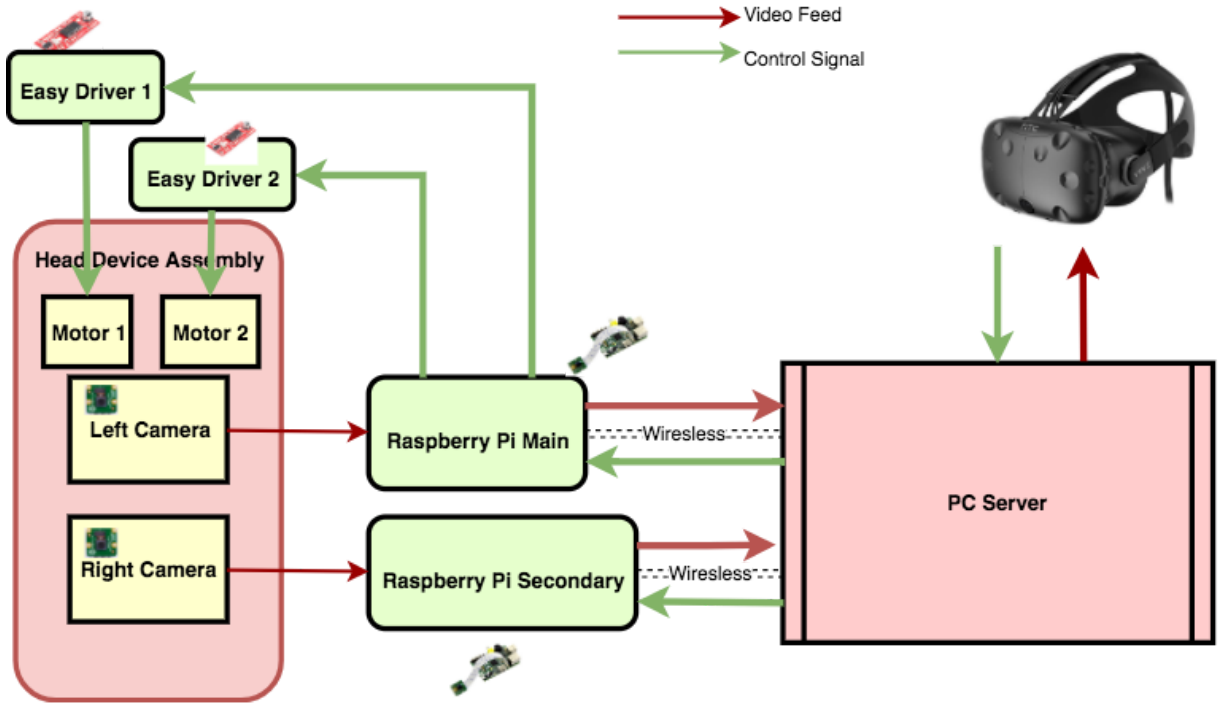
Team members:

ZheKai Jin Cooper Union 2019' ECE
YiFei Shao Cooper Union 2019' ME

2017 FALL

Table of Contents:

Flow Diagram



Abstract:

This project aims to build a simple realization of the abstract idea “Telepresence” by the aid of microcontroller(Raspberry Pi in this specific case) and the Head Mounted Display (HMD) (HTC Vive) and supplementary hardware. The idea is to make a wireless connected device, mimicking the observer’s perspective to capture its ambient environment and feeding it into observer’s HMD, which therefore allow the observer to Fully perceive device’s surrounding as if he/she is in the device’s position. The application can range from basic monitoring to highly expendable like bomb disposal with an extended machine arm. The implementation involves web development, server setup, data transmission & encryption and hardware control. The report is presented in a way that one can follow the writer’s work from here and extend it to any other more specific appliance with need.

Mission statement

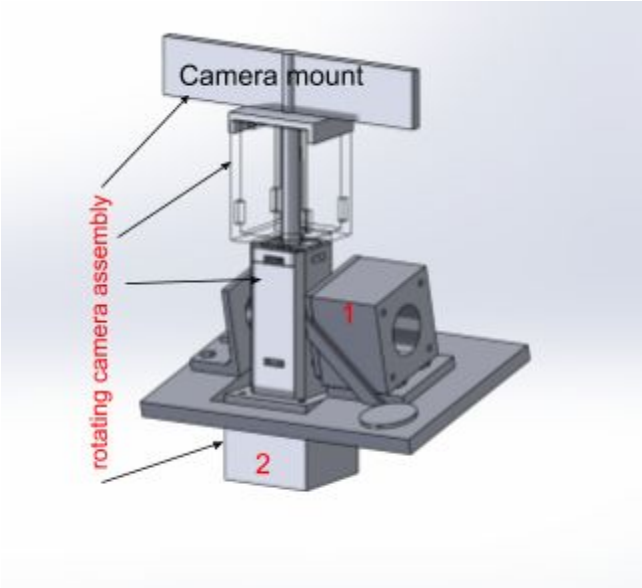
1. Capture stereo-image from cameras and feed a video stream into the computer
2. Display the Stereo-image into the HMD
3. Control the Rotation of the Camera with HMD orientation, or with subsidiary motion controller.

Team distribution:[1]

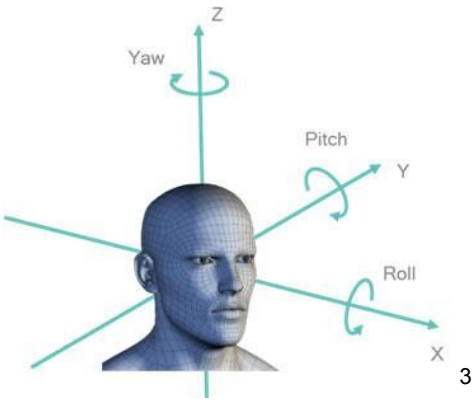
Research & Design:	Yifei Shao (Simon) & Zhekai Jin (Scott)
Web Development:	Zhekai Jin (Scott)
Server Setup :	Zhekai Jin (Scott)
Data Transmission:	Yifei Shao (Simon) & Zhekai Jin (Scott)
Environment Setup:	Yifei Shao (Simon) & Zhekai Jin (Scott)
hardware control:	Yifei Shao (Simon)
Code Maintenance:	Yifei Shao (Simon)& Zhekai Jin (Scott)

[1] Alphabet ordering

Hardware: Head Pan/Roll device



2



3

List of Hardware & Specifications:

- 2x NEMA 17 Motors 14 V 0.3 A
- 2x EasyDriver - Stepper Motor Driver[4]
- 8x 5mm long M3 Flathead Screws
- 7x 1 in long Wood Screw
- 1x 1 in long Aluminum Shaft ¼ in diameter
- 1x Ball Bearing No. R4, for 1/4" Shaft Diameter, 5/8" OD[5]
- Counterweight (quarters)
- Wood Base
- Jumper Wires

---

[2] Solidworks Rendered by Yifei Simon Shao

[3] https://www.qualcomm.com/news/onq/2016/08/11/device-motion-tracking-immersive-vr-freedom-wires

[4] https://www.sparkfun.com/products/12779

[5] https://www.mcmaster.com/#60355K503

The human head can generally move about three axis(Yaw/Pitch/Roll). In order to mimic the behavior of the head, three motors should be implemented to control camera motion. However, since the purpose of the robot is to explore new environment, the roll motion is not the most often used motion. Therefore, we omitted the roll motion to simplify the robot.

The rendering of the device is attached above. The device is composed of a few components: For size reference, a piece equal to the size of a US quarter is placed on the main platform. 2 motors each controlling a degree of freedom, motor 1 controlling the pitch(tilt), and motor 2 controlling the yaw(pan). Motor 1 is secured onto the main platform with a custom designed motor mount , motor 2 is mounted to the base of the rotating camera assembly. There is another mount on the opposite end of motor 1 for stabilizing purpose. It has a hole and a press fitted bearing in the middle for reducing friction during rotation. The rotating camera assembly is composed of (from bottom to top) motor 2, mounting base, the transparent counterweight housing, cap, and camera mount. The camera mount is directly connected to the motor via a long shaft. The counterweight mass can be adjusted by placing heavy metal inside.(We used 5 dollars worth of quarters) The cameras can be attached to the camera mount with drilled holes and small screws. The main platform has a cutout for the motion of the Rotating Camera Assembly. The cutout is big enough for the pitch range from -80 to 80 degrees. All components of motor mounts and rotating camera assembly are printed in the MakerSpace.

On the hardware side, there are also some required components to make the device function correctly. First of all, both motors are stepper motors which means they both need driver. We used a driver for each motor, called EasyDriver[6]. Also, a power supply of 12 V (15V for very reliable torque) and 0.6 A is required to run motors at the same time. For testing, we

[6] https://www.sparkfun.com/products/12779

used the lab bench power supply. Two 4-AA-battery-pack in series should be able to sustain 3 operation hours with the device in motion and a 5 V supply for Raspberry Pi 3 should be branched out from the battery pack to allow camera to be in function, which reduce the on-fly hours to 1 hour. Connecting battery in parallel is the temporal solution, but later a replacement of power supply should be satisfactory.

- Subscriber-end High-level Overview

The Pubnub Subscriber is a service that receive JSON communication from online. There are three parameters that need to be specified for proper communication: A Subscribe Key, a Channel Name, and JSON message. For our purposes, the robot is receiving just two integer data: degree in pitch and degree in yaw. Each time the robot receives a communication with a valid message, a callback function is executed to run the motors.

- Head Control & Communication

Upon the receiving the Pubnub communication, the RasPi runs the callback function to run the motor. Since the target value from the HMD can change quickly, it is important that the program does not run the motors pass the target. Therefore, each time a communication is received, the motors run 1 step towards the target position. If in the next time step the target angle goes to the opposite direction, the callback function runs the motor in the opposite direction by 1 step, avoiding any extra stepping. For our communication, the subscriber key is "`sub-c-e1ee380e-d3b8-11e7-b83f-86d028961179`" and the channel name is "scott".

A package[7] is used to simplify the communication to the EasyDrivers, therefore running the motors with ease. The creator simplifies controlling steppers to a function, taking *direction* and *number of steps* as parameters. Combined with the controlling constraint discussed in the previous paragraph, the callback function will step yaw and pitch motors both by 1 step, and

[7] https://github.com/davef21370/EasyDriver

wait for the next communication. Since the rate of publishing is 300 fps, the continuity of command is more than sufficient. However, there is a few problems associated with this method: Time lag is about 0.5 sec and stepping 1 step in each callback results in problematic jerky motion.

In order to improve the problems with this device, we should obtain motors with feedback, and implement a controller for each motor which will result in smoother motion.

Research & Design & thought experiment

The design process of this project was carried out with respect to the flow diagram using the common divide-and-conquer methodology. The project was divided into three parts or three hows: How to stream the Picamera's feed on to the internet? How to feed the received stream in to the HMD? How to let the observer's motion be linked to the device?

How to stream the Picamera's feed on to the internet?

The idea was first to develop a simple server written in PHP and then upload the stream using embedded development and web server setup to push the Picamera's feed into the localhost. Apache server was selected for this purpose for it is open-source and free. The site was designed with simple HTTP 5 & CSS syntax and was designed primitively only to render the feeding image and allow low delay time by minimizing the package size and maximizing the data rate in the allowable limit.

How to feed the received stream in to the HMD?

The process of feeding the stream on the apache server into the HMD stayed unknown for a long time. The first initiative was to use 3D Unity, the cross-platform game engine developed by Unity Technologies to develop a custom video streamer platform in steam platform and then utilize it in steamVR platform since it is a seamless platform already integrated in Vive technology. In this specific setting with HMD specification, a already-built

software virtual desktop was found to be most fit to this task, and its functionality will be explained later in this article.

How to let the observer's motion chained with the device or the motor?

The first idea for this question was to extract information using Vive's API(Application programming interface) to extract the head's position and then feed the information into the Raspberry Pi with the same channel used in the server route for video stream. Later in the development stage, a better method was found to utilize the Publisher/Subscriber mode with JSON object in transmission. And the head motion information in HTC vive was later found complicated to extract due to the lighthouse technology HTC used to monitor the HMD to save its cost. Thus, a subsidiary motion controller Leap Motion was used for this purpose and the API of it turned out to work with this idea.

Setup & Application

Though the project was developed in stages, the initialization and setup can be generalized to a few stages:

HTC Vive Setup:

The vive setup was a lot of trouble to set up. We went through a few steps and concluded that the linux version of SteamVR is just not ready to develop on. The hardware setup is not very difficult. First, there are two lighthouse that come with the VR headset to triangulate the position of the headset in the room. Connect those to the power source, and secure them on to the wall in a way that they can see each other and the headset at the same time. After that, connect the headset to the computer via a hub and the hardware setup is complete.

The software setup is a lot trickier. First of all, we had a nvidia GTX-840, Which does not work for the VR system no matter what driver we used.[8] Therefore, we purchased GTX-1080 which worked for SteamVR setup and tutorial, but we had trouble connecting it via the development interface[9]. Therefore, we thought using a VR application in the Steam store may be a good idea. Other than that method, we also looked into using other operating system like Windows.

Web Server Wrapper Setup:

- Step 1: Install Raspbian on your RPi

[8] It always has error 306/307. A borrowed graphic card seems to work better for the purpose
[9] Running Vrcmd has error

- Step 2: Attach camera to RPi and enable camera support[10]
- Step 3: Update your RPi with the following commands:

```
sudo apt-get update
sudo apt-get dist-upgrade
```

Occasionally if camera core software updates have been done then a sudo rpi-update may be used to benefit from these before they become available as standard.

- Step 4:

  For Jessie Lite run sudo apt-get install git

  Clone the code from github and enable and run the install script with the following commands:

```
git clone https://github.com/silvanmelchior/RPi_Cam_Web_Interface.git
cd RPi_Cam_Web_Interface
./install.sh
```

Pubnub [publisher / subscriber service] setup:

Initialize the java code in IDE [JVM required] with leap motion on → The API will then publishing leap motion's data into the cloud service.

The python script in the Raspberry Pi was then ran to receive the JSON object about hand motion and then used to control the servo motors.

Raspberry Pi Stepper Setup:

- Step 1: After installing the stepper motors, plug them into the Easy-Driver.

[10] http://www.raspberrypi.org/camera

- Step 2: Plug in the driver on to a breadboard and for each pair of similar setup pins, connect them and then connect them to a pin on the Raspberry Pi (ENALE, MS1, MS2, GND).
- Step 3 : Connect the STEP pin on each driver to a PWM pin on the Pi and connect the DIR pin on each driver to a normal pin on the Pi.

The connected diagram looks like the following(Need to find the pic on the table)

The application of the project is extensive and broad:

- The ready-to-go application of this project is obvious with its tolerance to allow observer to see things beyond his/her biological ability and basically be anywhere around the world. Though phone calls and video chat has achieve that, but this project allow the realization of such a task to be mangable by a individual.

- The idea of using a device to be your “eye” or wingman is to avoid the risk and threats of being hurt in a perilous situation. Bomb defusal, Toxic environment exploration, and combat surveillance, etc, are all possible stages for it to perform.

- The extension ability with the servo driver, and a extendable machine arm with a high-definition allow some intricate and delicate task to be perform in industry, with the three-dimensional attribute and the concurrent chain between hand motion and machine arm, the range of task can be ranged from task small as wafer circuit inspection to large as robot control as shown in the movie Pacific Rim.

Publisher-end High-level overview

Framework included and resources needed:

- PubNub - Data Streaming Service. Free Sandbox mode for developers!
- Java SE - JDK 8 - Be sure to get the correct version for your operating system
- Leap Motion Visualizer and Java SDK - Again Be sur[11]e to get the correct package for your operating system PubNub Java SE SDK[12]- This needs to go into the project libs directory
- Java IDE[13] - Use your favorite, such as JGrasp, NetBeans, IntelliJ, or Eclipse

This project used the most efficient transmission model available for small package transmission: Publisher/subscribe. Publish/Subscribe is ideal for the point of interest since only the hand motion will be needed for the servo to make the corresponding move which will mostly be defined in the callback function in the subscriber's end>(Raspberry Pi). Leap motion is built as a tool to capture various kinds of hand motion with its API provides information about yaw , pitch and roll of hand. Then, with the degree of each motion, we can define the Picamera's relative location in a three dimensional model, just as the hands' motion. Therefore the chain between hand movement and Picamera's was tied and connected. And the publish/subscribe model can inherently support MIMO(multiple input, multiple receiver) mode, which allow many observer to use the channel together or one user to control multiple view together, depending on the definition and ID & data in the Json Object transmitting.

---

[11] https://www.pubnub.com/get-started/
[12] https://www.pubnub.com/docs/java-se- java/pubnub-java-sdk
[13] https://blog.idrsolutions.com/2015/03/the-top-11-free-ide-for-java-coding-development-programming/

RPi_Cam_Web_Interface

The web server installation and web page initialization was all adapted from a well-known project in Raspberry Pi community: RPi Cam Web Interface(RPI). RPI is a web interface for the Raspberry Pi Camera module. It can be used for a wide variety of applications including surveillance, dvr recording and time lapse photography, which highly used the API provided by the Picamera's library. It is highly configurable and can be extended with the use of macro scripts. It can be opened on any browser (smartphones included) with a easy edition on the web page syntax. Now multiple functions are supplied:

- View, stop and restart a live-preview with low latency and high frame rate. Full sensor area available.
- Control camera settings like brightness, contrast, ... live
- Record full-hd videos and save them on the sd-card packed into mp4 container while the live-preview continues
- Do timed or continuous video recording with splitting into fixed length segments
- Take single or multiple (timelapse) full-res pictures and save them on the sd-card (live-preview holds on for a short moment) Preview, download and delete the saved videos and pictures, zip-download for multiple files
- Trigger captures by motion detection using internal or external detection processes.
- Trigger captures by many scheduling-possibilities
- Circular buffer to capture actions leading up to motion detection
- Control Pan-Tilt or Pi-Light

- Shutdown/Reboot your Pi from the web interface
- Show annotations (eg timestamp) on live-preview and taken images/videos
- Supports selection from 2 cameras when used with a compute module

This project mainly utilized the interface's basic function of streaming, and upgrade the single camera's feeding into two to allow later space for stereo imaging and then change the server code to allow higher data transmission rate within the HTTP protocol.

For simplicity and a full screen rendering, a lt of function was omitted but nevertheless we can always retrace back to the former version and gain motion detection and other interesting functionalities. Here is a list of diff between the adapted version and the original one.

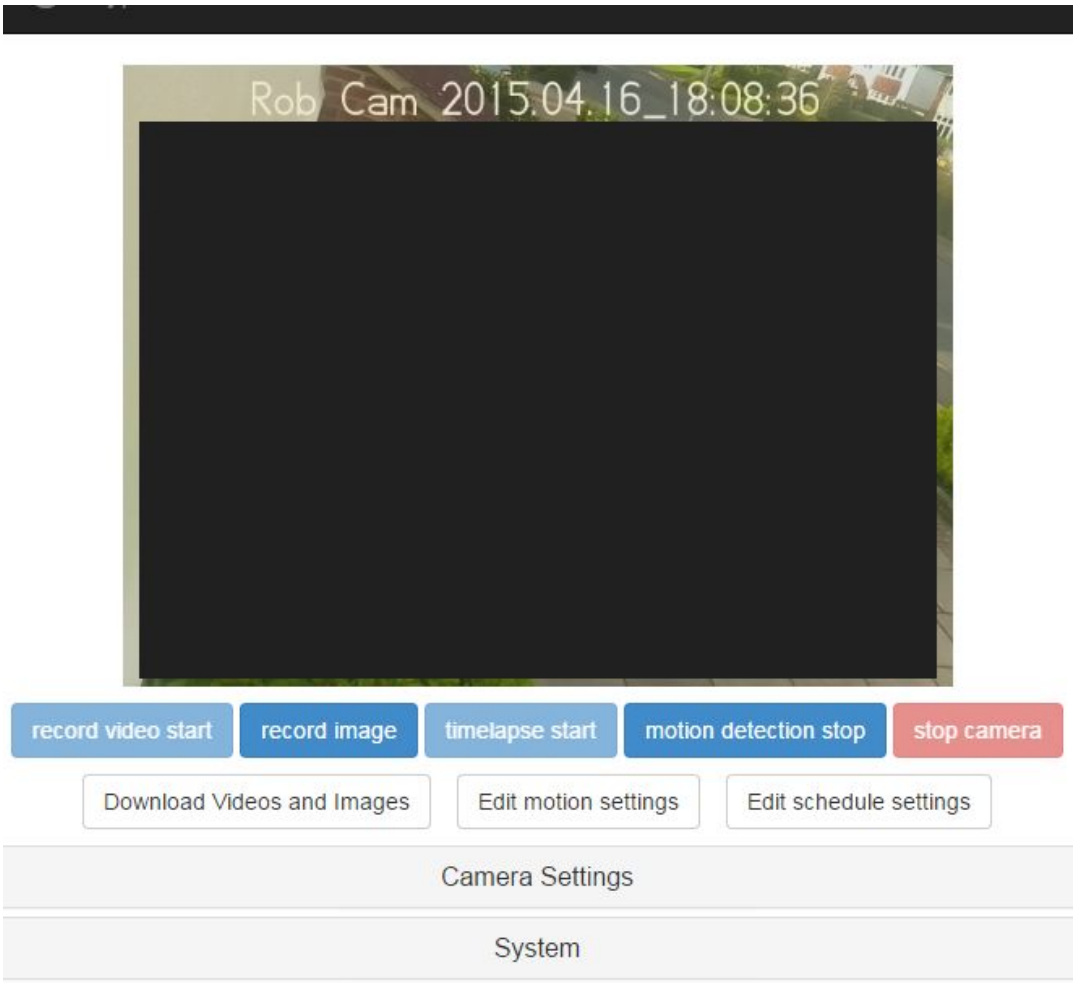
Older version:

- Video streaming manual start/stop button
- Camera's configuration allowed in web page
- Scheduler setting
- motion detection start/stop button
- video save cloud space for 50 MB

New version:

- video streaming autostart on power up and stop only when camera disconnected or power off\
- camera's configuration allowed in back end only [hard coded]
- scheduler setting disabled[since multiview already allowed]

- motion detection can be set but no OpenCV analysis include and not visible on webpage
- Video save disallowed
- Multiview added with up to 4 camera's space added.
- Live Pi bandwidth increased for better streaming
- Image overlay disabled to avoid confusion to users
- User buttons added to allow user to add extra macro function [undefined for now]

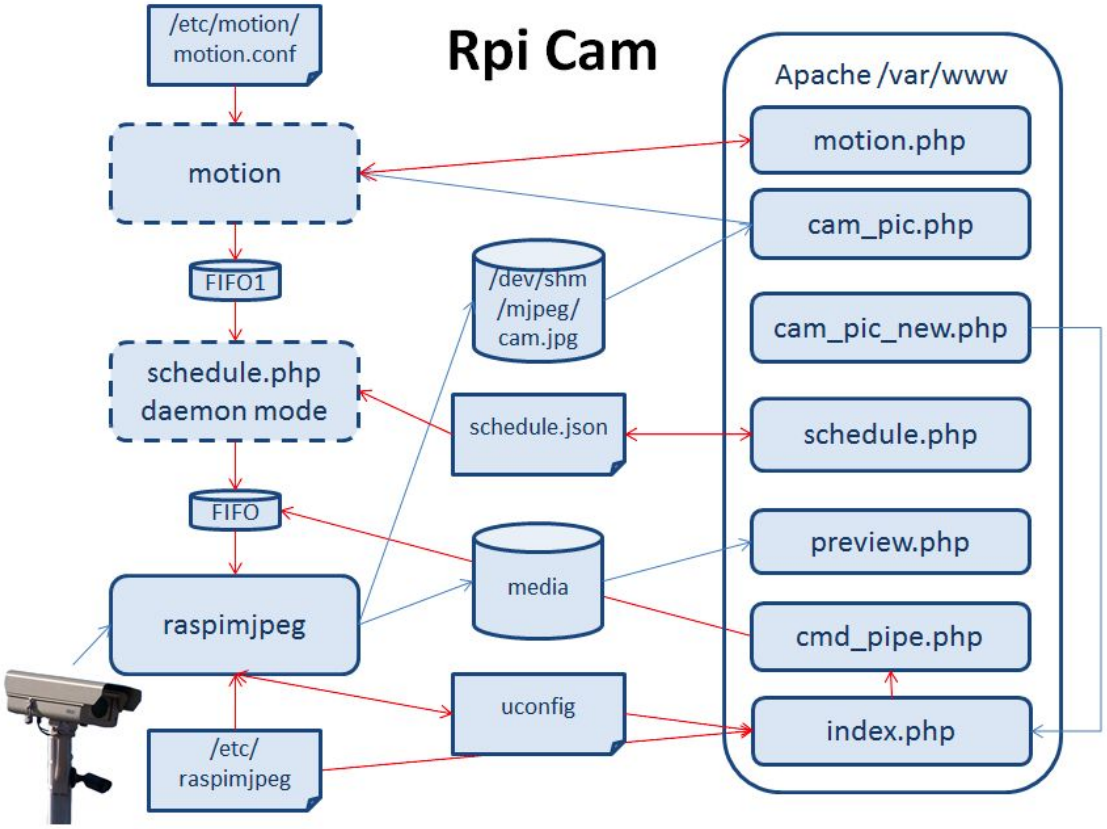


Single view version

Web development

The web development part of this project was mainly aimed to enhance the performance of RPI and to allow the rendering of the pi's feed onto the internet in a cheap and efficient way.

PHP WORK:



Picamera's development framework can be concluded in above graph and referred at github documentation.

WEBPAGE dev : simple HTML was used to plainly let the webpage to render the channel information(video streaming) based on the feeded json file. A better web page is

intended to be developed later for a better visualization and the button key should be avoided but changed to another form of control with user's hands occupied in the leap motion control.

Porting: ngrok was used as a tunneling tool to feed the localhost in its services webpage in a most cheap way. Though Apache was used locally , domain name is provided by the ngrok and a better and stable domain name should be maintained later solely for this project.

Usage:

```
$ unzip /path/to/ngrok.zip
$ ./ngrok http 80
```

HMD as a second monitor

The idea of Multi Monitor, also called multi display and multihead, is the use of multiple physical display devices, such as monitors, televisions, and projectors, in order to increase the area available for computer programs running on a single computer system. Now, however, with the application Virtual Desktop's[14] help, we are projecting our own monitor screen's image into the HMD in a curved way to render the effect of 3D, and also the control of mouse is handed over to the controller of HTC Vive.

Here is a list of features it offers:

- Hardware accelerated 360 video playback
- Ability to play/stream YouTube 360 videos
- Browse and view 360 photos
- MilkDrop support for music visualization
- 3D Side-By-Side video support
- Game launcher with voice commands
- Multi-monitor
- Environment Editor to create custom environments
- Steam Workshop integration for environments
- Dashboard integration so you can see/use your desktop in any VR game

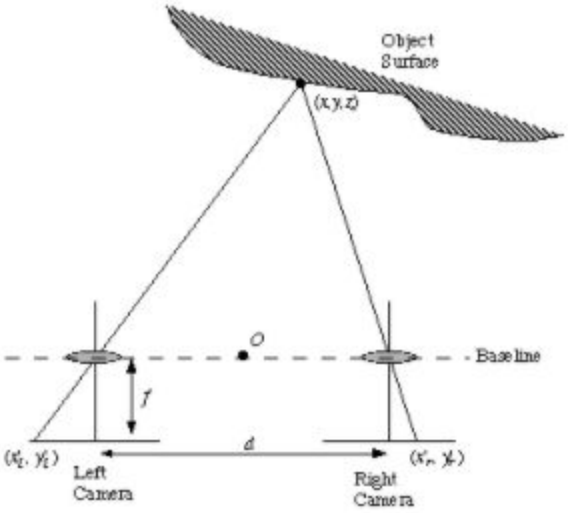
While Virtual Desktop is not a open source product, the technology behind it is believed to be involved with desktop visualization where all the components of the user desktop are virtualized, which allows for a highly flexible and much more secure desktop delivery model. In

14

addition, this technology supports a more complete desktop disaster recovery strategy as all components are essentially saved in the data center and backed up through traditional redundant maintenance systems. So anything happened in the HMD is not likely to hurt the computer/server with the data center is located at the server side. If a user's device or hardware is lost, the restore is straightforward and simple, because the components will be present at login from another device(HMD).

Future work (stereo imaging)

While this project for now supports a simple route from Picamera to HMD, the image is still flat and 2D, like what we viewed on a browser typically. To make the 3D HMD really perform its function. The stereo imaging can come into play where the multiview function of RPI explained before allow such a change.



The stereo imaging allows image to be post processed in a relatively fast rate to allow integration and miking, the image can be played in a playback fashion with a little delay, but in our project the real time data need a powerful processor at the server end to deal with the time lag and the quality of image. The exploration of this field is expected to be touched later.

Reference

1.https://users.cs.cf.ac.uk/Dave.Marshall/Vision_lecture/node11.html#SECTION00132300000000000000
2.https://en.wikipedia.org/wiki/Stereo_imaging
3.https://elinux.org/RPi-Cam-Web-Interface
4.http://www.instructables.com/id/IoT-Motion-Controlled-Servos/
5.https://www.pubnub.com/
6.http://blog.leapmotion.com/integrate-leap-motion-arduino-raspberry-pi/
7.https://developer.leapmotion.com/documentation/index.html?proglang=current
8.https://www.rs-online.com/designspark/building-a-raspberry-pi-2-webrtc-camera

Appendix

The code and documentation is maintained under this repository of Github service:
https://github.com/ZhekaiJin/VR-TELE