USB Webcam setup and algo logs

4th March 2024

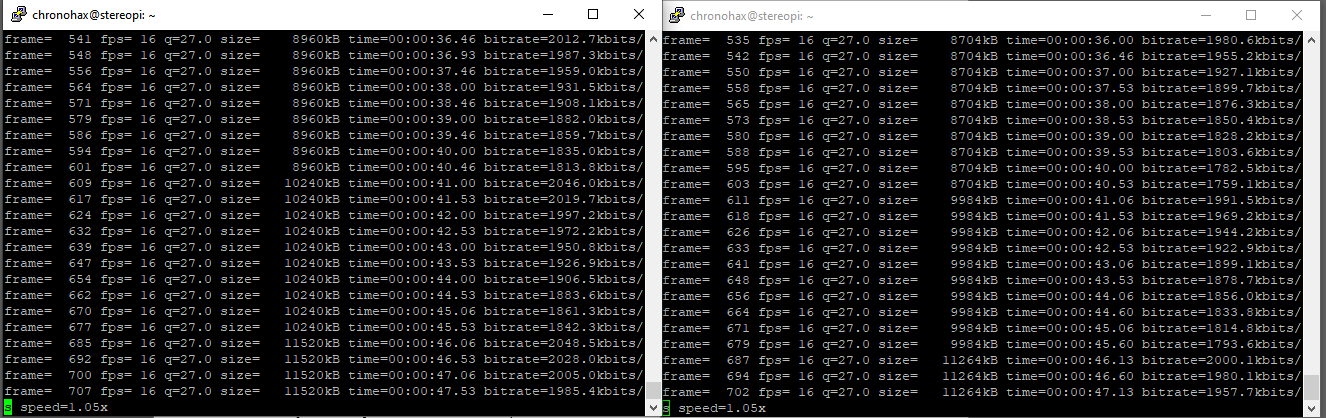
Setup:

* 2 x USB Webcam (Microsoft LifeCam HD-3000) borrowed form B16 University Labs
* 1 x 8GB RAM Raspberry Pi 5 single board computer (or better, need enough processing speed and RAM for stereo recording at high fps)
* Keyboard if not using SSH (PuTTy etc)
* Micro HDMI to HDMI cable adapter for monitor display to check video stream quality
* 5V/3A USB-C power adapter and cable, (recommended is 5V/5A but only available from official Rasp Pi adapter and not usable anyway if using this as portable camera which is the point, the difference in amperes meant the pi cant drive some usb peripherals like drive but so far usb webcam seems to work fine [Power consumption of usb webcam on a Raspberry Pi 4 - Raspberry Pi Forums](https://forums.raspberrypi.com/viewtopic.php?t=343144), refer testing part)
* Compile ffmpeg manually ([Compiling FFmpeg on the Raspberry Pi - Pi My Life Up](https://pimylifeup.com/compiling-ffmpeg-raspberry-pi/)), for some reason, using apt install ffmpeg gives EGL not initialized error when using ffplay etc.

Initial fps testing, recording both camera concurrently, results into 24+ stable fps, this is a good sign! Using command: ffmpeg -f v4l2 -framerate 15 -video\_size 1280x720 -input\_format mjpeg -i /dev/video0 -preset faster -pix\_fmt yuv420p out-test0.mkv

From [video capture - recording from webcam using ffmpeg at high framerate - Stack Overflow](https://stackoverflow.com/questions/47292785/recording-from-webcam-using-ffmpeg-at-high-framerate).

To check current connected usb webcam devices, use v4l2-ctl –list-devices command.



# Algorithm setup:

<https://chat.openai.com/share/1158d1f9-11a1-4962-bd06-8234f7d0a43e>

import subprocess

import time

from datetime import datetime

while True:

# Generate timestamp for the filename

timestamp = datetime.now().strftime("%Y%m%d\_%H%M%S")

# Define filenames for each webcam

filename\_cam0 = f"video\_cam0\_{timestamp}.mkv"

filename\_cam2 = f"video\_cam2\_{timestamp}.mkv"

# Define the FFmpeg commands for each webcam with optimized settings

command\_cam0 = f"ffmpeg -f v4l2 -framerate 15 -video\_size 1280x720 -input\_format mjpeg -i /dev/video0 -preset faster -pix\_fmt yuv420p -t 20 {filename\_cam0}"

command\_cam2 = f"ffmpeg -f v4l2 -framerate 15 -video\_size 1280x720 -input\_format mjpeg -i /dev/video2 -preset faster -pix\_fmt yuv420p -t 20 {filename\_cam2}"

# Run the commands using subprocess.Popen() to start each process

process\_cam0 = subprocess.Popen(command\_cam0, shell=True)

process\_cam2 = subprocess.Popen(command\_cam2, shell=True)

# Wait for both processes to finish

process\_cam0.wait()

process\_cam2.wait()

print("Automated recording started for both webcams...")

# Wait for 5 minutes before capturing the next set of videos

time.sleep(300)

something like that.

And have main.py that can run both manual.py and automated.py script together, refer to chatgpt link for code and more info.

Basic algo done for now, so moving to 3d mount real quick: Planning to use onshape.

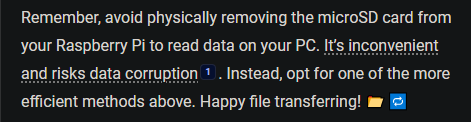
Proposed interval for now is to have picture taken every 5 minute and a 30 second video every 30 minute. (Calculation for file sizes needed.), maybe can increase or decrease interval based on that and necessity to get more data etc.  
Preliminary calculation (assuming 720p 24 fps at 8 bit colour, each picture/frame should take around 1MB):

Thus: Assuming infinite battery life, capturing picture when user is awake (0800 - 2400), 16 hours:

There will be 12 picture, and 2 x 30 second video every hour. Thus: 12MB + 2 x 30 x 12MB x 16 = 11532 MB which is around 11.532GB. Thus, every week 11.532GB x 7 = ~80.724GB is used, which means weekly backup and data transfer is needed. This is pretty ideal.

This excludes manual video capture using button.

# Exporting videos from Pi to main PC



I’ll be using FTP via FileZilla. [Transfer Files to and from Raspberry Pi using FileZilla FTP (Windows PC) | Random Nerd Tutorials](https://randomnerdtutorials.com/transfer-files-raspberry-pi-filezilla-ftp-windows/)

8th March 2024

Changed the preset to ultrafast, now it is very smooth! Also added image as well, skipping first frame cus that one is glitchy. Other than that, I noticed as expected hot temp throttle the smoothness very much.

To do next:

* Figure out remote ssh and wifi hotspot connection for the pi and autorun the run\_me.py script on boot.
* Optional QOL: add button and modify script to include manual video capture.

Using systemctl /system MD, autorun run\_me.py script and can use relevant commands to monitor/stop etc. The problem before is that script stops when ssh disconnected ie wifi out of range or webssh on phone stops running in background.

The service is stereo\_lifelog to run run\_me.py. followed this tutorial [How to Run a Linux Program at Startup with systemd (howtogeek.com)](https://www.howtogeek.com/687970/how-to-run-a-linux-program-at-startup-with-systemd/) /etc/system/system/stereo\_lifelog.service

Use sudo systemctl daemon-reload/enable/start/stop/status stereo\_lifelog in order to refresh, start, stop and status check.

Side note: don’t put pi in pocket, it overheats I think, tomorrow, try very fast instead of ultrafast and see if it heat throttles. NVM it overheats but don’t cause frame drops I think!

-ss 1 (skip 1 frame also needed for video so better thumbnail ie 1st frame)

^ sus, idk whether this or very fast preset or being indoor/lowlight cause stuttering/flickering(frame skipped?)

Need to investigate more tomorrow morning!

15th March 2024

Testing shows that overheating cause very bad throttling that can cause frame desync due to frame drops, thus its better to increase interval even when I want a lot of data to avoid that, cus or else the data will be useless anyway. Other than that, I also got a Raspberry PI Camera Module 3 for further testing on higher resolution at more stable framerate, and hopefully don’t have much low-light/overheating throttling issue. I also bought a USB microphone for simple audio. Modification to run\_me.py to include Boolean for options either USB webcam or CSI-MIPI camera is added, and using raspicam-vid/raspicam-still instead of ffmpeg, those uses libcamera linux as a base, and current settings seems to work well so far, need further testing regarding sync though. Other difference is that the file saves as .mjpeg instead of mkv so need to copy to mkv using 0.5run .py script in HOST pc.

[Raspberry Pi Camera Module: More on video capture - Raspberry Pi](https://www.raspberrypi.com/news/raspberry-pi-camera-module-more-on-video-capture/)

Another thing to keep in mind is distortion correction might be needed due to use of wide lens: [Correcting lens distortion using FFMpeg | Daniel Playfair Cal’s Blog](https://www.danielplayfaircal.com/blogging/ffmpeg/lensfun/v360/lenscorrection/fisheye/dodgeball/2020/03/24/correcting-lens-distortion-with-ffmpeg.html) [Raspberry Pi Camera Module 3 Comparison - {DPHacks}](https://dphacks.com/2023/01/18/raspberry-pi-camera-module-3-comparison/)