USB Webcam setup and algo logs

4th March 2024

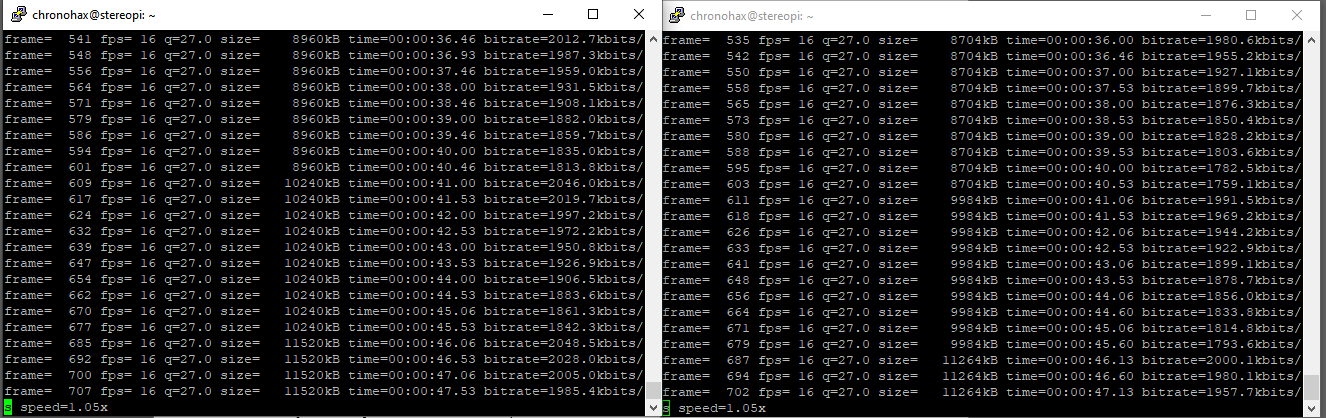
Setup:

* 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.

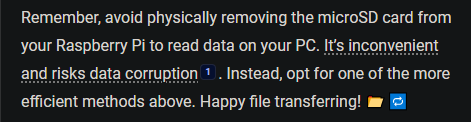
Proposed interval for now is to have picture taken every 5 minute and a 30 second video every 30 minute. (Calculation for file sizes needed.), maybe can increase or decrease interval based on that and necessity to get more data etc.  
Preliminary calculation (assuming 720p 24 fps at 8 bit colour, each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