Is a Raspberry Pi and PiCam NoIR V2 suitable for Prospective Optical Gating?

2193036C

March 20, 2019

Abstract

ABSTRACT TEXT HERE

1 Introduction

1.1 Aim and Motivation

The aim of this project was to investigate the suitability/effectiveness of a Raspberry Pi and available camera attachment, the PiCam V2 NoIR for Prospective Optical Gating. The end goal is to replace the reference camera and computational components of the setup with the RPi and PiCam, as shown in **Figure 2**.

Currently the optical setup is custom made for the application, and so is not easily replicated elsewhere. Replacing the aforementioned components with the Pi and PiCam allows for a cheaper and more easily replicable setup. This, along with a 3D printed light sheet generator (the focus of another undergraduate project), would allow other research parties to create their own setups.

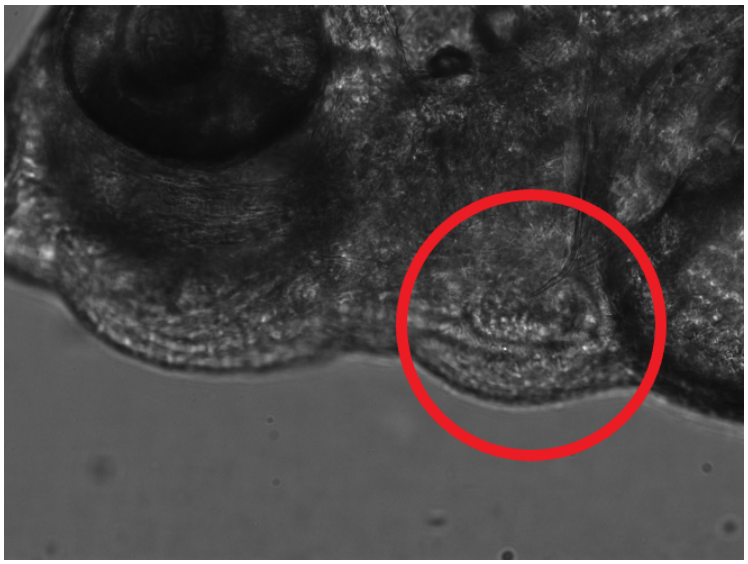

Figure 1: Example of a reference frame, with the heart circled. The portion of the image containing the heart and some of the surrounding area is 350x250 pixels

1.2 Background Theory

1.2.1 Light Sheet Microscopy and Prospective Optical Gating

Light sheet microscopy is a technique that uses a light sheet to excite fluorescent cells in a 2D plane of the subject, with a 3D image being constructed from multiple 2D scans. However, tissues can be bleached or damaged due to the incident high intensity laser light so continuous exposure is unwanted.

Prospective Optical Gating is an imaging technique that allows for the 3D imaging of an object with periodic motion, being tested and developed for imaging the hearts of live zebrafish embryos [REF]. The aim is to capture one 2D slice at the same phase in every heartbeat, so as to reduce exposure time of the subject, and remove the need for continuous laser exposure. This works by first using a set of brightfield reference images, captured continuously during the imaging process, to find an estimate for the heart period. This is then used to predict when the heart will be at the same phase in subsequent beats and trigger the laser at these times.

The frame is compared to the reference images by calculating the Sum of Absolute Differences (SAD) for each reference frame. This finds the absolute difference between each pixel in one image and the corresponding pixel in the other, and sums them to give a single value. The SAD is a measure of how similar two images are; the larger it is the less similar.

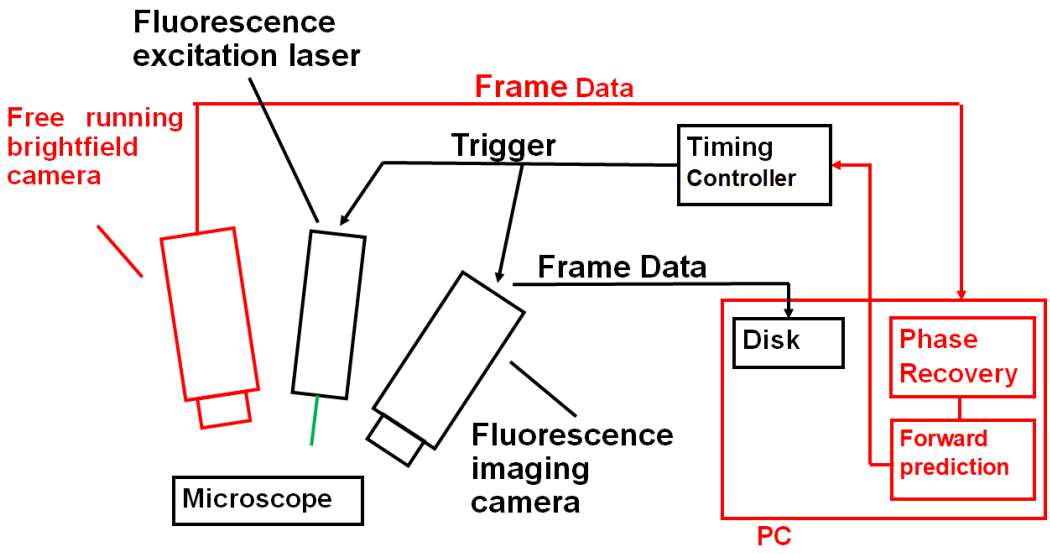

Figure 2: Diagram of a prospective optical gating setup. This project is investigating if the Raspberry Pi and PiCam could replace the highlighted components.

1.2.2 Hardware

A Raspberry Pi Model B+ is an ultra-small form factor computer designed for accessibility and cheap cost, and in this case had Raspbian, a Debian based operating system installed. The 1.4GHz 64-bit quad-core processor makes it a powerful computer for its size, and the general purpose input/ouptut (GPIO) pins mean it is well suited to prototyping and hardware control. The PiCam V2 NoIR is the second iteration of the camera attachment developed for the RPi with no near-infrared filter, enabling is to detect wavelengths from 800-2500nm [REF]. This version is used because the wavelength re-emitted by the fluorescent cells is in the near-infrared range.

The camera sensor is the Sony IMX219PQ CMOS sensor, which is back illuminated meaning the metal wiring is underneath the light sensing layer, increasing efficiency compared to sensors with wiring in front. A rolling shutter is used to read pixel values row by row from top to bottom. It reads then clears each row one by one. Field of view restriction is also available for capture of images of smaller resolutions, meaning they are captured by the centre of the sensor instead of using the full area.

Any image capture and display system has what

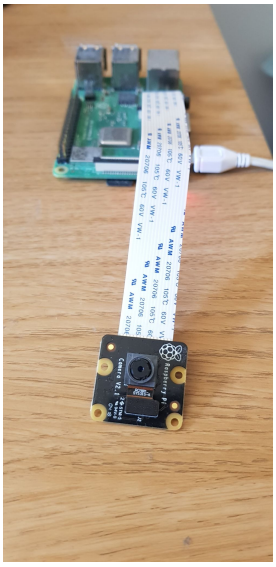

Figure 3: Photograph of Pi and PiCam

is know as an image pipeline. This is the set of components between the capture device, the CMOS sensor in the case of the PiCam, and the render, e.g. a monitor or television. An image is captured in Bayer format then processed, called de-mosaicing, into a raw image format e.g. RGB or YUV to be gamma corrected, scaled, rotated and encoded if requested.

1.2.3 YUV Colour Space

As is shown in section 4.1.1, YUV is the optima image format for rapid capture on the RPi. The probable reason for this is because after any required rotations, rebalancing, noise reduction etc. the Bayer image is de-mosaiced into YUV image, which is then processed and resized by the GPU.

YUV is a colour space that takes into account human perception, which allows for smaller bandwidth of the chrominance components. These channels are that contain the colour information about the image, UV in the case of YUV. Y is the luma component, meaning it contains brightness information [REF]. Consequently, if the U and V values from a YUV image are removed, the remaining Y values form a grayscale image. This is particularly useful for the application of Prospective Optical Gating because the algorithms that recover the phase of the heart only require a grayscale image.

The specific YUV encoding is YUV420, where the 420 represents the chroma subsampling; a process that reduces file size by reducing the resolution of the chroma information. Essentially, this means that for every region of 4 by 2 pixels, it considers only 2 in the top row, and zero in the bottom row, resulting in chrominance pixels the size of 2x2 image pixels.

2 Initial Investigation

Subsampling Rate	**Effective Frame Rate (fps)**
1/1	80.00
1/2	40.00
1/3	26.66
1/4	20.00
1/5	16.00
1/6	13.33

Table 1: Available Frame Rates

Unfortunately, due to the restricted availability of live fish, the Pi could not be tested in the physical setup. Instead an emulator that had been written previously in Python was used to find what was needed for the Pi to work. Since it only emulates the physical system, the parameters that could be altered in the emulator were limited. It was used to determine the minimum frame rate of the reference images required to produce an accurate phase calculation and period estimation.

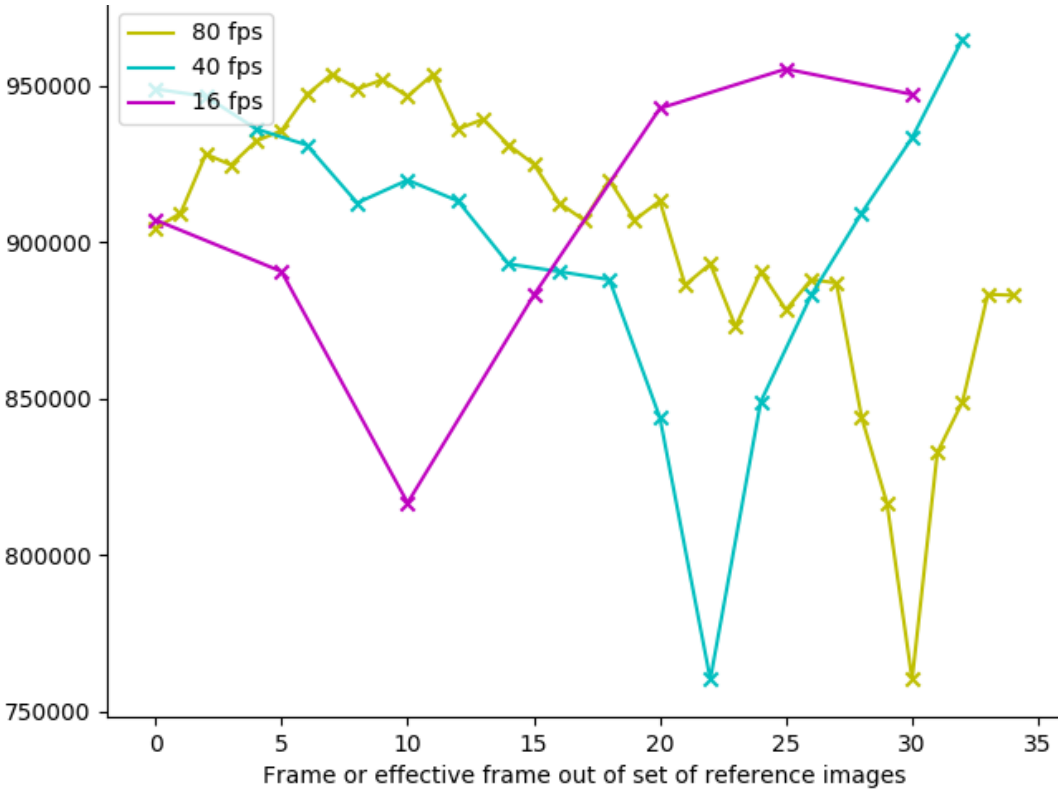

Figure 4: Effect of lowering reference frame resolution on SAD plots

2.1 Findings from the Emulator

A set of pre-captured reference images was supplied, captured at 80fps. A simple algorithm to sub-sample this set was written, which extracted 1 in every 2, 1 in every 3, etc. images to simulate lower framerates. The emulator was then run using these lower frame-rate images, and the outputs were analysed. The initial algorithm to provide a period estimate failed to run at 13 fps, but worked at 16fps. This gives an absolute minimum required frame rate of 14.67 ± 1.33 fps.

The phase recovery algorithm effectively finds the minimum of the SAD plot shown in **Figure 4**, and so the sharper the 'V' in the plot, the more reliably the phase can be recovered. **Figure 5** shows the degradation of the phase extraction as framerate is reduced. The ideal phase plot is a perfect sawtooth, with values ranging smoothly from $0 - 2\pi$. The plot mostly retains its shape at 40 fps, and even down to 26.67 fps, however it loses its form almost completely at anything lower. This sets the framerate goal for the RPi, i.e. if it can capture and process images at a rate faster than 30 fps it is theoreticall fast enough.

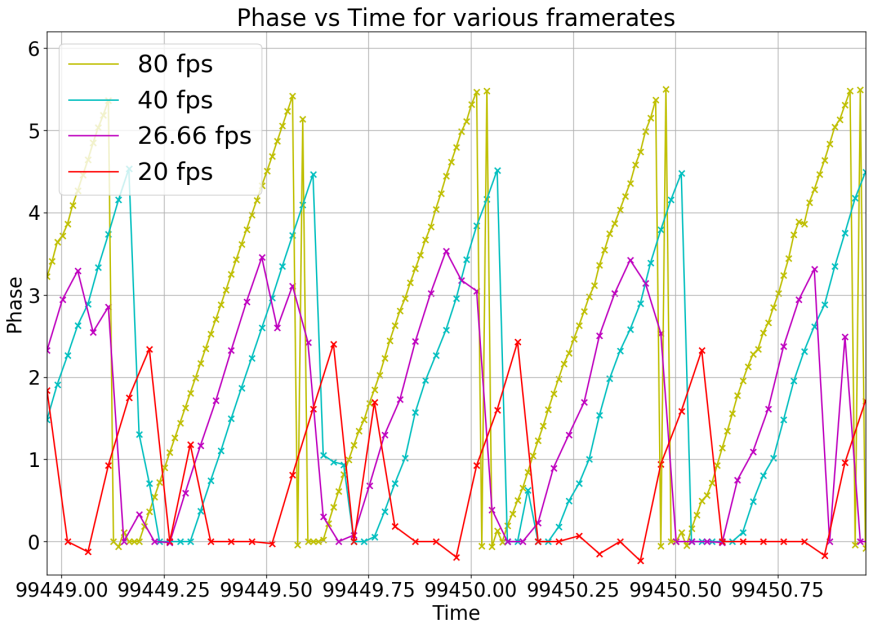

Figure 5: Plot of recovered phase against time. The units of time are irrelevant here as it is a comparison of the shapes of the plots. The ideal shape is a sawtooth pattern.
N.B. *There appears to be a systematic downward vertical shift to all of the phase values, which is likely an artefact of the algorithm. This suggests that it could be adjusted and improved for lower frame rates.*

3 Camera Mount

To use the RPi and PiCam in the current setup a camera mount was designed in CAD. The current optical setup is built using Thorlabs components, so the mount had to be compatible with its rail system. The final design, shown in **Figure 6**, had to be simple enough to print (no sharp overhangs etc). The camera is screwed into the indented area, mounted to the rail system with the corner holes.

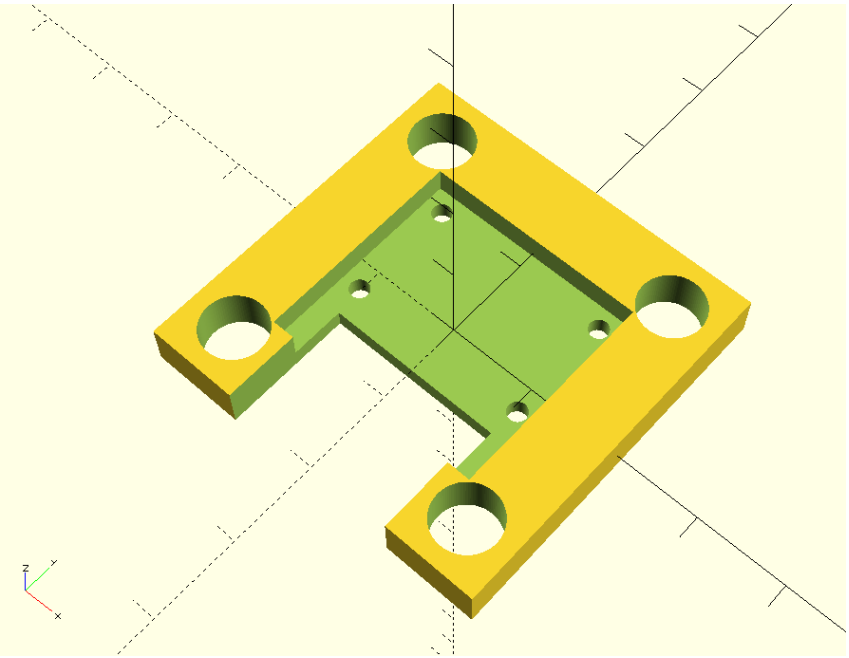

Figure 6: CAD render of the final camerea mount design

4 Investigating the RPi

4.1 Analysing the Camera Input

Since the primary goal is to determine whether the PiCam can be used to capture and process reference images, a large portion of the investigation was focussed on how it could be used to do this.

4.1.1 Which Image Format to Use?

The image format to be used for testing had to be decided. Process time to capture images of various resolutions, shown in **Figure 7**, was measured, as well as the variance between a set of images taken of the same still scene, shown in **Figure 8**. It is evident that for images of resolutions less than 600 by 600 pixels YUV is the optimal format.

4.1.2 PiCam API

The PiCam has an extensive API that includes a multitude of capture methods that allowed images and videos of various formats to be captured as a number of filetypes and Python objects. Video and still image capture was available, however it was inconvenient to apply the image processing to video capture. Therefore the focus of the investigation was on still image capture, and which would be the optimal method.

The first tests were conducted using the **picamera.capture()** class, which allowed capture to either a file, or to an object with a **write** method. The time values shown in **Figure**

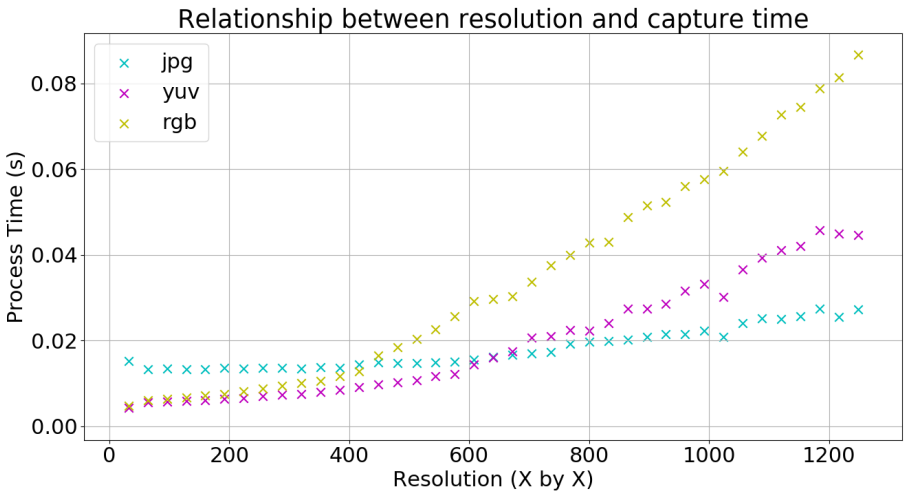

Figure 7: Process time to capture a single image vs resolution for the three primary image formats.

7 were obtained using this method. Subsequently, a new set of classes, themselves within the **picamera.array** class were found. Named **PiArrayOutput, PiRGBArray, PiYUVArray**, they allow capture of JPEG, RGB, and YUV images directly to a NumPy (Python's array library) array and so lend themselves towards image analysis. Within these methods is a stub method called **analyze**, which is a blank method that is called every time an image is captured.

The SAD calculation is the most computationally expensive part of the phase recovery, so this was used to find the maximum frame rate achievable by the Pi. A custom method was written in Python to take Luma (Y) values of each frame (since only luminance is required for phase recovery) and find the SAD between it and a set of pre-determined reference frames This could only achieve frame rates above 30fps for resolutions less than 50x50 pixels however. A C++ SAD algorithm that had been written for the emulator was available, so this was installed on the Pi and used instead, since it is considerably faster than Python code.

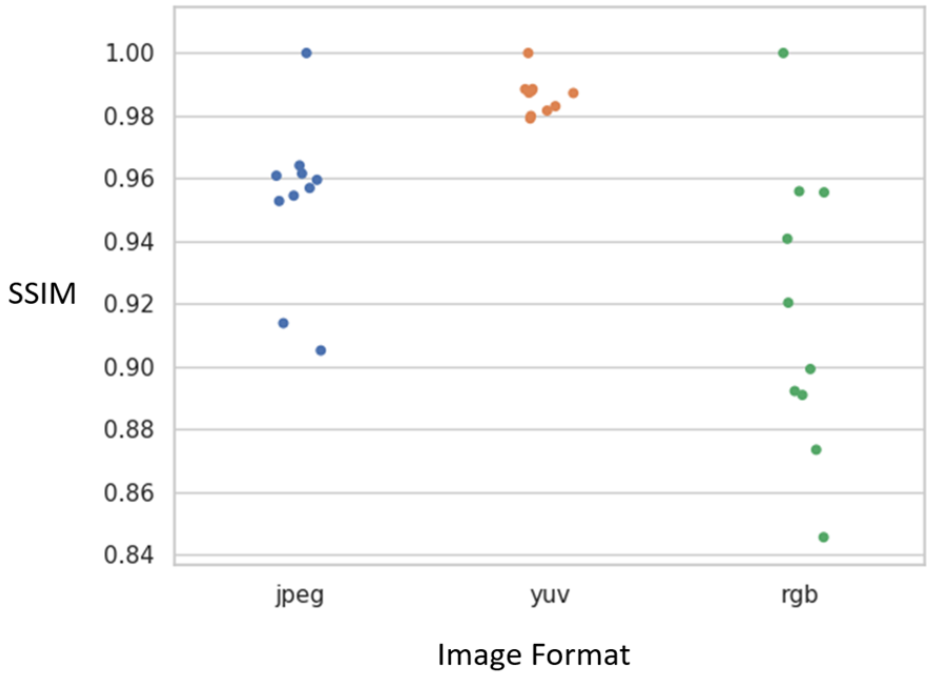

Figure 8: Variance between a set of images. The SSIM, an image comparison metric, was used where 1 is most similar, 0 is least similar.

5 Results

Figure 9 shows the achievable frame rate for continuous capture and SAD computations running on the RPi. The current capture resolution for the physical gating setup is 659x494, however the heart takes up a small portion of the image, so smaller resolutions are acceptable. The threshold frequency and relevant resolutions are marked. Any data points above 30 fps and 300x300 pixels represent resolution-framerate pairs that are within the goals set in Section 2. Three data points representing resolutions of 320x320, 352x352, and 384x384 pixels are in the desired region.

Therefore, under the conditions set in Section 2, this suggests that the RPi is in fact fast enough to be used to capture the reference images.

6 Conclusions

This project has shown that for smaller resolutions of 300x300 pixels the RPi is theoretically able to simultaneously capture a stream of reference images, and perform phase recovery computations. With some adjustments to the current setup and minor changes to the algorithms this should be suitable for capture of a large enough reference image (see **Figure 1**). The highly customisable nature of the Raspberry Pi, with many different operating

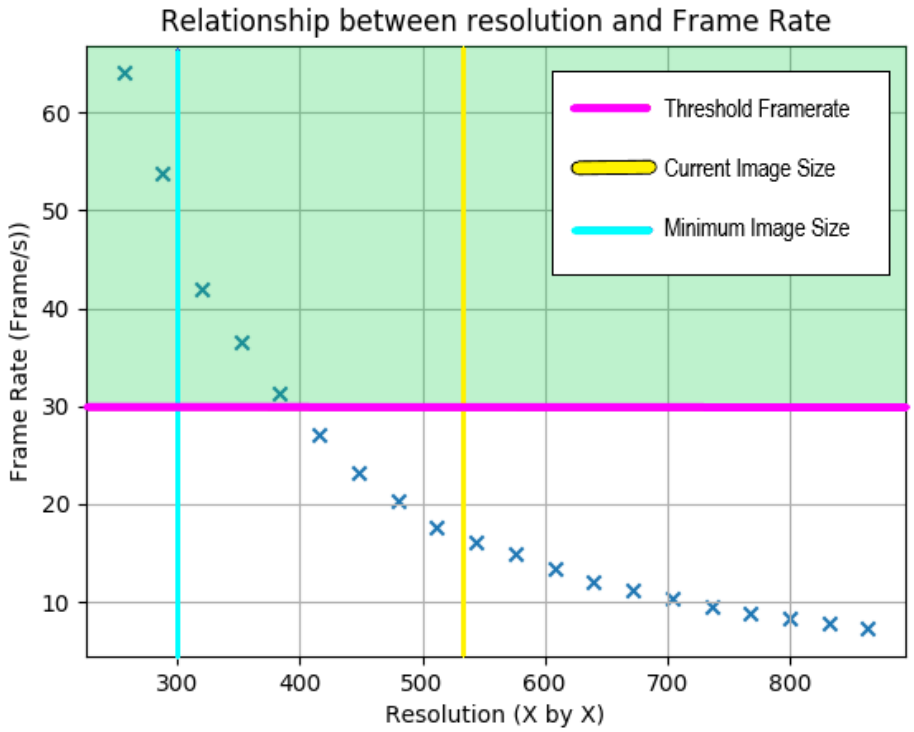

Figure 9: Achievable frame rate vs Resolution while running SAD calculations on RPi. Values above the red line are resolutions faster than the lower limit. Values to the right of the blue line are larger than the area of current images containing the heart.

systems available, it provides a versatile programming environment to write and/or install custom analysis methods. As discussed in the introduction an inexpensive and customisable setup is ideal for educational purposes (althought a lower power laser may be necessary). This could also be used for research in locations with fewer resources, for example.

6.1 Continuations

The first obvious continuation to this project woudl be to capture a set of reference frames with the PiCam, and repeat the investigations in Section 2.

Following this, practically evaluating the PiCam in the physical system with live fish will give a clearer idea of whether it is actually suitable for this application.

Should this be the case, modifying the entire setup to accommodate the Pi to trigger the laser could conclude this line of investigation. In this case the Pi may not be able to interface with the laser by itself, and may need to be paired with e.g. an Arduino, as is suggested in *M J Colville et al* [REF]

A more mathematical route could be taken, comparing for example the entropy of images taken by the expensive camera and the PiCam. This would be an investgation into the amount of information contained in an image, and how much of it is actually useful.