GitHub: koopaduo2
Optical Engineer, M.S.
Repository link: https://github.com/koopaduo2/Beam-GUI.git
Reddit: u/koopaduo

Laser Beam Profiling with a Raspberry Pi

Table of Contents

1. Introduction

This document serves as a technical introduction and manual to the Raspberry Pi (RPi) image acquisition, processing and laser beam profiling hardware and associated Python code. As a disclaimer, please know the hazards and understand laser safety when working with lasers. I am not advocating anyone to work with lasers and will not be held liable for any reason. Do so at your own risk.

I designed this system as a means to characterize laser beams on my home optical bench, where I cannot afford the cost or complexity of commercial beam profiling systems. While some end-users may try to adopt the system for use in a laboratory setting (academia, industry, etc.), I cannot guarantee the efficacy of the system that some users may need. It is up to the end-users to find that the beam profiling and processing complies to their specification and it solely up to them to qualify its use. Furthermore, the software is open-source under the MIT license, so it may be modified as seen fit.

That aside, I hope to provide open source beam profiling with decent performance.

2. Hardware

The hardware consists of a Raspberry Pi compute module, RPi HQ camera module, camera connector and a variable ND filter. The variable ND filter is required to prevent camera saturation in general. It should be understood that a variable ND filter is essentially two crossed polarizers and may introduce unwanted effects depending on the polarization of the beam. End-users can replace the ND filter with a fixed ND filter or suitable beam attenuation techniques. The best compute performance will be achieved with the latest RPi 4B. While the program is not memory intensive, it does require significant compute power which translates directly to image processing throughput and refresh rate of the images.

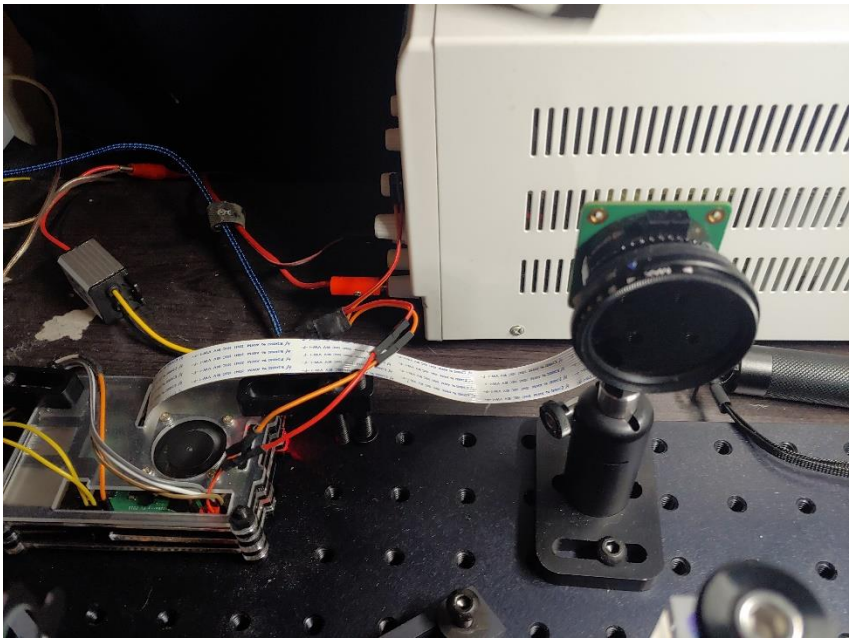


Figure 1. Beam profiling hardware. RPi, connector, mounted camera and ND filter

To properly use the system for laser analysis, an optical breadboard and mounting components are usually required. That is beyond the scope of this document, as anyone interested in using the system will already be familiar with such hardware. I will briefly discuss mounting the camera module, but otherwise optical mounting and alignment techniques are assumed.

The parts list will be available via the Bill of Materials (BoM) spreadsheet. To assemble the camera, first remove the CS-mount cap of the HQ module revealing the inner sensor.

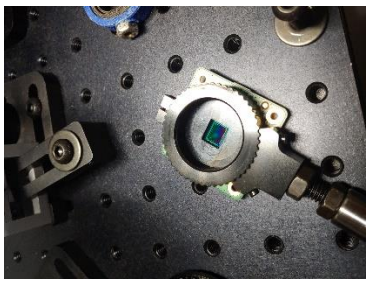
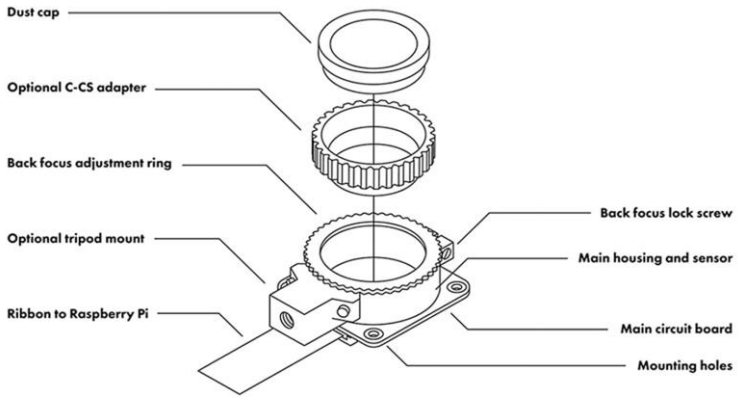


Figure 2. RPi HQ camera module with Sony IMX477 CMOS sensor

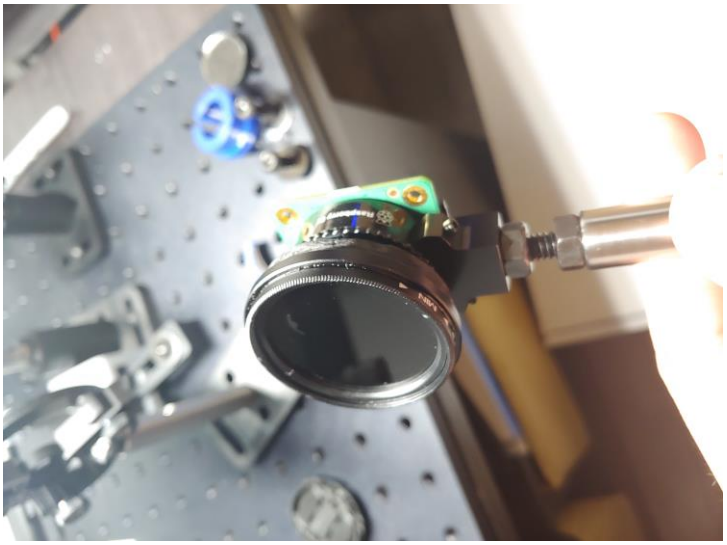
The CS-mount cap comes in two pieces, a threading ring onto CS-mount (C-CS adapter pictured below) and the actual dust cover which threads onto the ring.



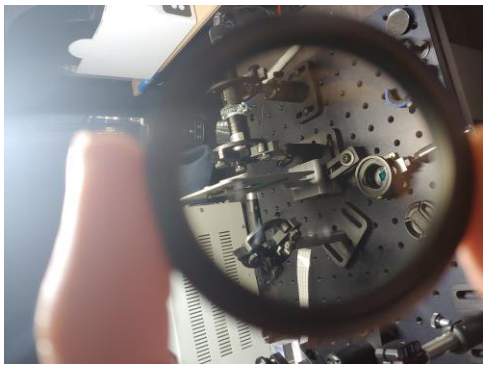
Remove the dust cap from the adapter ring, it won't be needed. I was not able to find a variable ND filter which threads onto CS mount. If you can find one, use that. Otherwise, use a phone variable ND filter and affix it as follows. Cut the clip which normally connects it a phone. You should now have a threading ring for the ND filter. Glue the front side of the C-CS adapter ring to the backside of the ND filter threading ring. Let the glue completely cure before mounting, otherwise it may contaminate optics. The result is pictured below. Furthermore, the threading rings can be glued at an angle using a shim or similar method to minimize back reflection off the variable ND filter if desired. My initial setup overlooked that fact and they are glued flush.

Now you have an adapter piece which mates to the CS thread on the camera, which the ND filter then threads onto. Below is a picture of the glued adapter threaded onto the HQ camera module.

You can now thread the ND filter onto the assembly.



The camera assembly has a 1/4-20 thread on the bottom for mounting. You can use a 1/4-20 set screw and two nuts to affix it to an optical mounting post also having 1/4-20 thread as shown above. The ND filter also comes with its own dust cap which can be used to cover it when not in use. The ND filter at minimum and maximum transmission are shown below.



The detector setup as above does not include a lens for the camera. In doing so, the beam profile is an unscaled image. This allows us to measure the physical dimension of the real beam. If separate focusing optics are required, they can be included as needed.

The camera module uses the Sony IMX477 CMOS sensor with Bayer color filter array. The sensor characteristics and spectral sensitivity of the camera are also shown below. It has peak response in green near 532 nm, a common laser emission line.

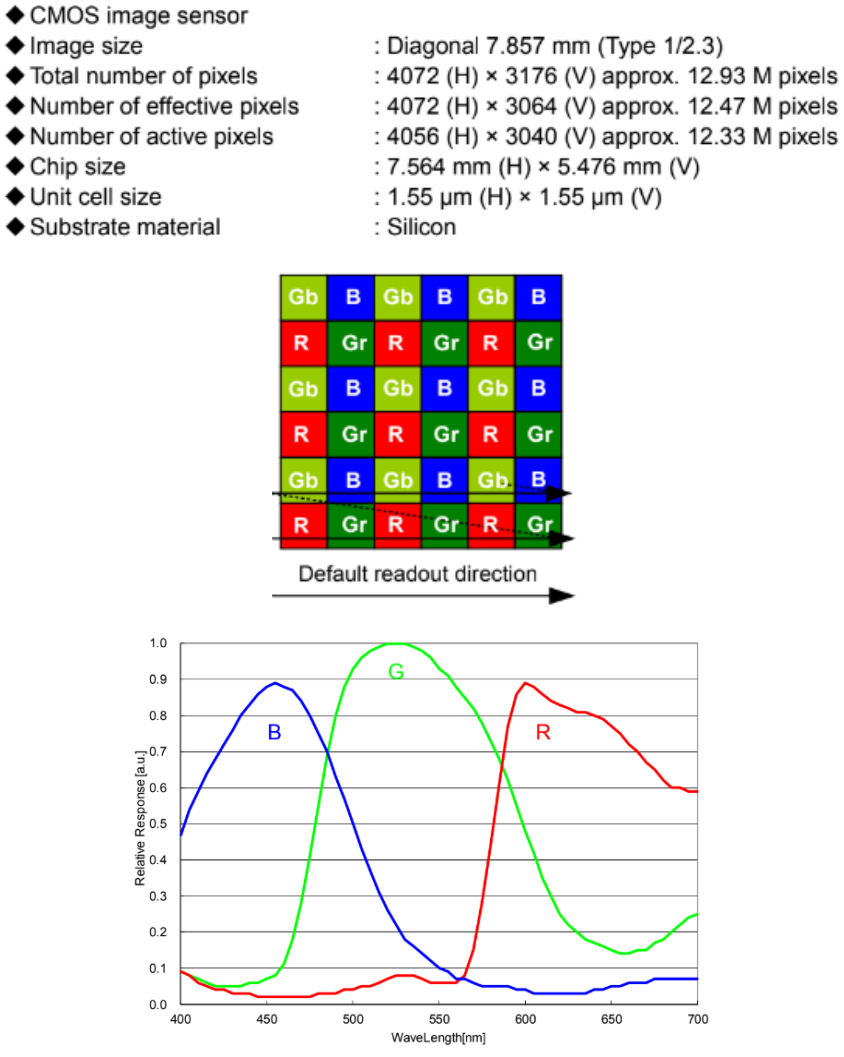


Figure 3. Camera device structure, CFA format and spectral response

If different detector characteristics are needed, the camera can be swapped out for other devices which may include monochromatic, larger sensor, USB or other cameras. The code has two separate functions for camera initialization and image capture. If these functions are rewritten for the different cameras, then the rest of the code should support the new camera. This would have to be integrated by the end-user for their use case.

The only other setup steps are to connect the camera via the standard flex cable, load the RPi OS, power the RPi and connect to it. This will not be covered here as it is well detailed on the web.

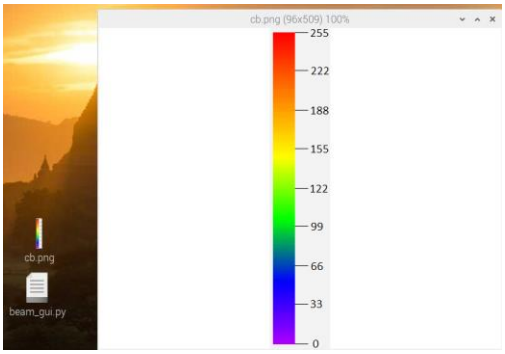
3. Software

The beam profiling software itself is a simple python script *beam_gui.py* which runs on the RPi; it utilizes PyQt5, OpenCV, numpy, matplotlib and picamera. The needed packages are listed below and available at the top of the code. They should be installed with pip or similar prior to running the script.

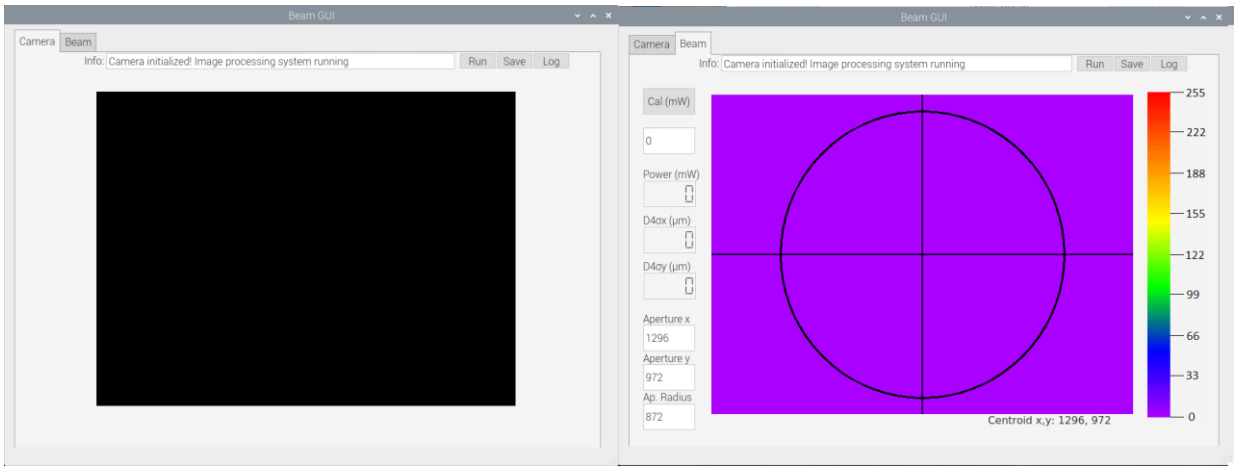
```
#required imports
from PyQt5 import QtCore, QtGui, QtWidgets
from PyQt5.QtCore import QThread
import numpy as np
import matplotlib.pyplot as plt
import cv2
import os
import datetime
from picamera.array import PiRGBArray
from picamera import PiCamera
import time
import math
```

Figure 4. Required Python imports

The only other source file required to run the script is a png graphic for the color bar called *cb.png*. I found this to be the easiest way to integrate the color bar. If an adjustable color bar, different heat map or other functionality is required, the user will have to integrate this themselves.



The graphical user interface (GUI) is shown below. It consists of two tabs, one to display the live camera image and one to display the beam profile and computed statistics.



The program consists of two main threads: one for handling the GUI and one for handling image acquisition and processing. With my test unit (RPi 4b and HQ camera module), I achieve approximately 1

FPS which is not terrible but is an obvious limitation of the computer. This most likely can be increased by the end-user by optimizing the code or removing undesired functionality.

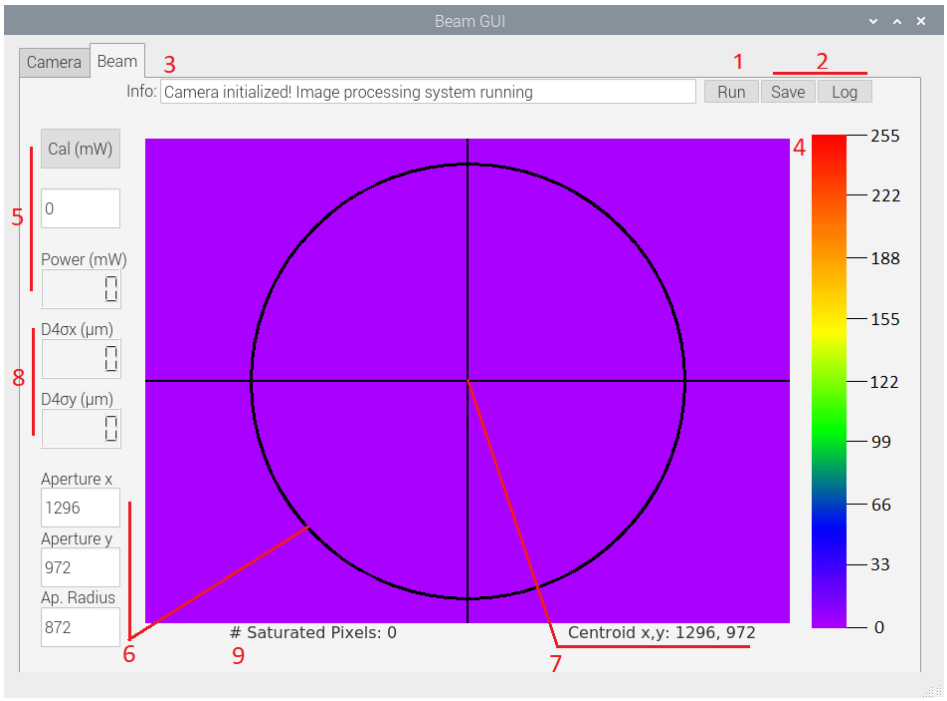


Figure 5. Features of the beam profiling tab explained below

The beam profiling GUI supports the following features:

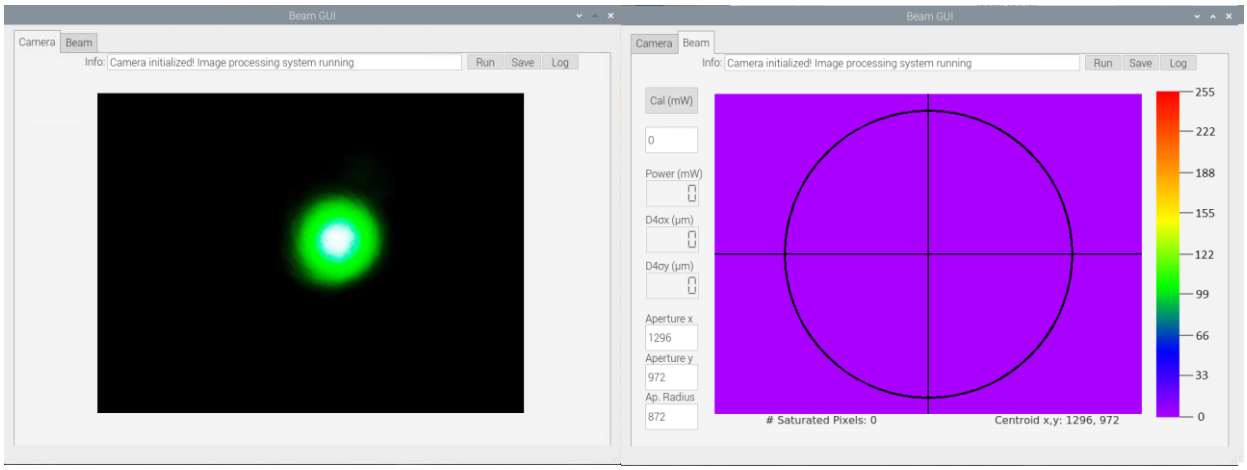
- *Camera* tab: display live images from the camera
- *Beam* tab: display the beam profile and statistics
- *1. Run* button: start the image acquisition and processing
- *2. Save/log* buttons: *save* instantaneous camera png image, beam profile png image, separate x and y beam profile plots along the centroid crosshairs, and beam statistics csv file. Continuously *log* or stop logging data. By default, the files are saved to the root directory + *saves* folder
- 3. *Info bar*: the info bar will display status messages such as: root directory, camera initialized, files saved, power meter calibrated, power meter failed calibration
- *4. Heat map* and *color bar*: the intensity of the beam is profiled on a rainbow heat map which scales from purple (0) to red (255).
- *5. Cal* button and *Power* display: calibrate the estimated power incident on the detector. This method simply correlates the pixel intensity sum with power measured from a real power meter. The user inputs a measured power reading in the text box under the *Cal* button then presses *Cal.* Calibration will fail if the image is blank or saturated. Saturation is defined as having more pixels at maximal intensity (255) than the variable *sat_num_allowed,* which by default is 20. This functionality is not meant to be an accurate power meter, only a real time diagnostic.
- *6. Digital aperture:* the digital aperture can be specified by setting the x and y center coordinates and the radius (pixel units px). The x and y default to the center of the image and the radius defaults to half the vertical image height minus 100 pixels. The aperture limits any beam profiling computations to its defined region using a mask.

- *7. Centroid tracking*: the computed centroid (px units) will be printed on the bottom right of the beam tab. The crosshair will track the centroid.
- 8. $D4\sigma$ width: the processing computes the ISO standard $D4\sigma$ width (in μm) of the beam and displays it in the GUI. The physical dimension is obtained by converting the measured px units to μm by multiplying by the sensor size (square 1.55 μm pixel). More info on how the beam width and centroids are computed can be found in d4sigma.pdf. They are computed using the *cv2.moments* function which is fast.
- 9. *# Saturated Pixels*: counts and displays the number of saturated pixels (intensity=255).
- *ROI* cropping: the camera ROI can be decreased if the user wishes to acquire zoomed-in images. In the *init_camera* function, change the value of *ZOOM_BOOL* to True and set *crop_factor*, *roi_start_x* and *roi_start_y* to the desired values.

In order to prevent camera saturation, on top of using a variable ND filter, the camera is initialized with minimum exposure and gain settings in the *init_camera* function. The camera initialization settings will be printed to the command line on running the script.

```
pi@raspberrypi:~/Desktop $ python beam_gui.py
qt5ct: using qt5ct plugin
qt5ct: D-Bus global menu: no
AWB is flash
AWB gain is (Fraction(0, 1), Fraction(0, 1))
Brightness is 50
Aperture is 0
Shutter speed is 114
Camera exposure speed is 0
Iso is 1
Camera digital gain is 0
Camera analog gain is 0
Camera v/h flip is True, False
Camera contrast is 0
Camera color saturation is 0
Camera meter mode is backlit
Camera crop is disabled
/usr/lib/python3/dist-packages/picamera/encoders.py:544: PiCameraResolutionRound
ed: frame size rounded up from 2592x1944 to 2592x1952
  width, height, fwidth, fheight)))
```

Some sample images are shown below. Otherwise, that is it! Have fun, be safe and feel free to contact me if you have any questions, comments or updates.



4. Related Links

https://github.com/koopaduo2/Beam-GUI.git

https://en.wikipedia.org/wiki/Laser_safety

https://en.wikipedia.org/wiki/Laser

https://en.wikipedia.org/wiki/Laser_beam_profiler

https://www.raspberrypi.org/

https://picamera.readthedocs.io/en/release-1.13/api_camera.html

https://pypi.org/project/opencv-python/

https://numpy.org/doc/stable/user/whatisnumpy.html

https://pypi.org/project/PyQt5/

https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.html

https://en.wikipedia.org/wiki/Beam_diameter

https://en.wikipedia.org/wiki/Image_moment

https://en.wikipedia.org/wiki/Variance

https://www.arducam.com/sony/imx477/