Building an Autofocusable Synchrotron Radiation Sensor Prototype for the Integrable Optics Test Accelerator

Leonardo Javier Rodríguez Gutiérrez[a]
[a]Fermilab, SIST. University of Illinois at Urbana-Champaign

Abstract. In this summer project we built a prototype of IOTA's synchrotron light monitor with an automated linear stage that allowed us to test several autofocusing algorithm to model the behavior of the actual monitors that will be located inside the Integrable Optics Test Accelerator in Fermilab. It was found that Luminosity $\mathcal{L}$ and $\frac{1}{ab}$ were the best beam parameters to estimate the best focus location of the monitors.

1 Introduction

The Integrable Optics Test Accelerator (IOTA) is a new machine located at Fermilab Accelerator Science and Technology (FAST) Facility which will allow us to perform new experiments using both protons and electrons at energies of 2.5 MeV and 150 MeV respectively. The focus of this summer project was to make a new model prototype for the synchrotron radiation station that will be placed inside the ring and develop different algorithms that would allow for autofocusing the images of the beam based on the previous work. The development of autofocusable algorithm's main objective is to make experiments more efficient by automatizing the preparations of the experiments to take place at IOTA, since the offline focusing is not precise enough and manual focusing would require a circulating beam, which is prohibited due to safety regulations.

2 Design

The design consists of an LED light located at the point where orbiting electrons will emit synchrotron radiation; a diffuse mask was applied on the LED to simulate a Gaussian beam shape. The radiation will travel through a series of tubes used to shield the beam from external light pollution, and will be reflected by two mirrors located inside the vacuum chamber. Moreover, the radiation

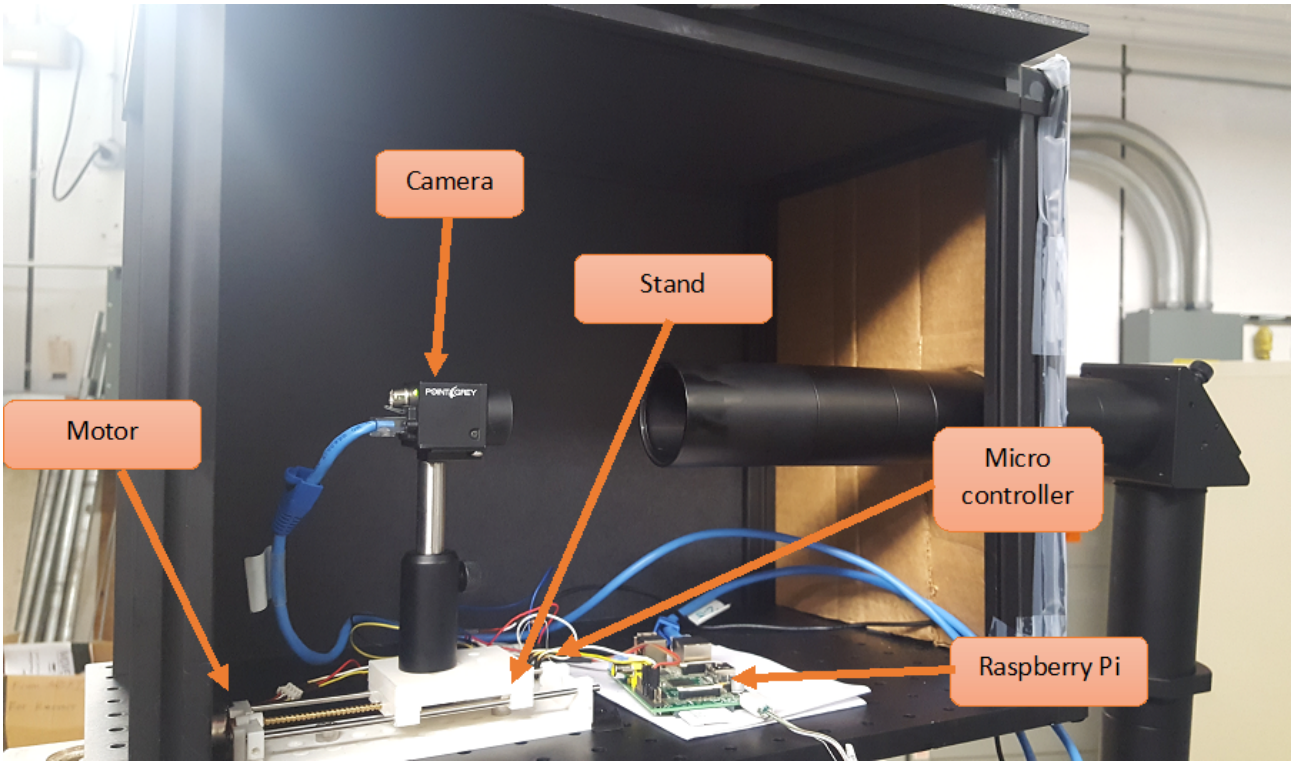


Fig 1 Picture of project setup.

will go through a lens with a focal length of 400 mm and a diaphragm. Finally, the beam will registered by the camera, which is located on top of a 3D-printed linear stage.

The main difference between the prototype and the actual experiments that will take place at IOTA is that the radiation that we are using in this project to study the focusing algorithms comes from an LED. This is an important difference since the light coming from the LED is monochromatic, whereas synchrotron radiation is not. This will be adjusted by placing a green filter on the path of the electromagnetic radiation. The sensors would be located at each dipole (i.e. bending) magnet which change the direction of the electrons, thus making them produce electromagnetic radiation, as shown in the following figure.

ADD PICTURE OF DESIGN and distance to lens

The prototype uses a generic green LED as the source of the beam, located at the entrance point

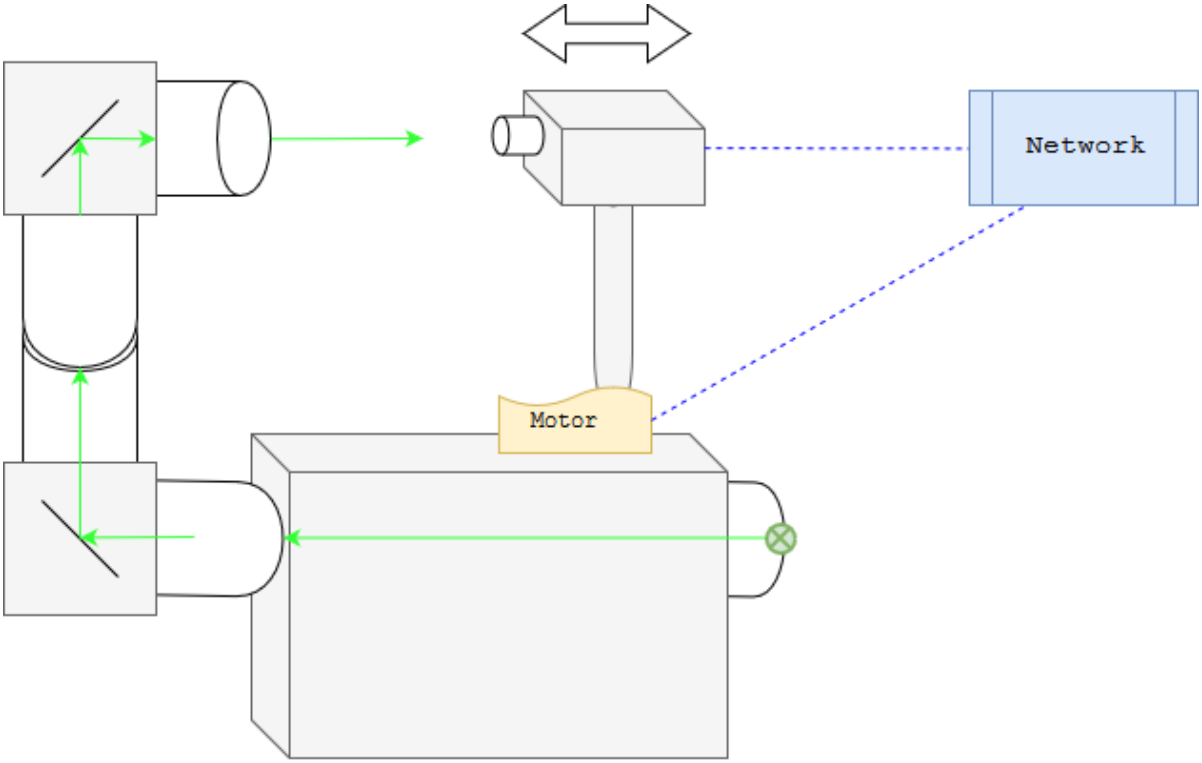


Fig 2 Design of prototype.

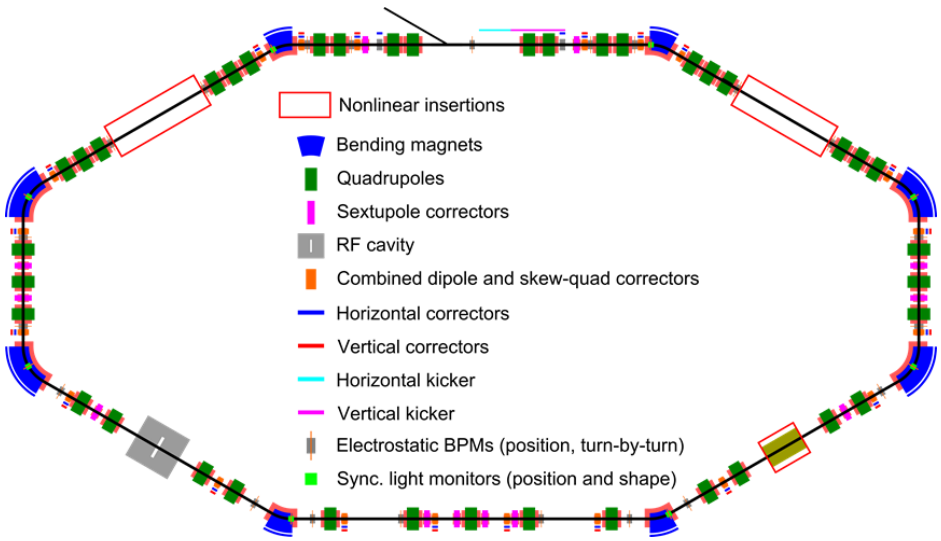


Fig 3 Diagram of the IOTA ring with the location of its components.

Fig 4 PointGrey BlackFly-PGE-23S6M-C Camera used for the prototype.

inside the dipole magnet, and travels a total length of depending on the position of the camera lens, going through a diaphragm and a focusing lens with focusing length of $f = 400\mathrm{mm}$. The camera is attached to a 3D-printed linear stage made to fit the motor's dimensions.

2.1 Camera

The camera used for the IOTA project is the PointGrey BlackFly-PGE-23S6M-C. This particular model was chosen because of its size, which is small enough to fit inside the black box without interfering with any other components. Moreover its sensors are very good, with a quantum efficiency of 50%, and good pixel capacity. Thus, the noise of this camera is small and it allows for more precise measurements. Moreover, the connection is through a 1 Gbps ethernet cable, which also powers the camera.

2.2 Motor

The motors used to move the camera were microstepper motors with the following specifications:

Table 1 Motor Specifications

Motor diameter	20 mm
Motor height	14 mm
Motor weight	22 g
Output shaft	1 mm (with a 5 mm diameter copper gear)
Output shaft length	8 mm
Fixed pitch	28 mm
Step angle	18 degrees
Rated voltage	12 V dc
Lead	4 line (phase four line)

Fig 5 Micro stepper motors.

Previous to this project, piezoelectric motors were used to focus the camera. These presented the problem that their repeatability was not constant, so the same movements would shift the camera by different amounts. The advantage of the microstepper motors is that they present a better repeatability and it is possible to know exactly the movement of the camera by knowing the amount of steps.

Fig 6 Raspberry Pi 2.

2.3 Computer

We chose a RaspberryPi 2 to control the motor and host the server responsible to communicate the motor with the client via a micro controller. This was chosen for its accessible price and power, since it can process the instructions fast enough, and has all the necessary communication ports such as Ethernet, USB and General Purpose Input/Output pins. Its price is $35 dollars a piece.

2.4 Micro controller

The microcontroller BIQU A4988 Compatible Stepper StepStick Motor Driver Module is the one in charge of directly controlling the motor. With this module, the step resolution can be set to 1, 1/2, 1/4, 1/8, and 1/16 of a step. This module presents a cost effective solution with all the necessary functions for only about $2 per unit.

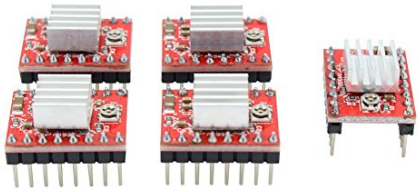


Fig 7 BIQU A4988 Compatible Stepper StepStick Motor Driver Module.

3 Network

There are several programs connected among each other that permit the cameras to communicate with the client and the motors.

3.1 BPMServerHard

This program works to obtain the information of each pixel directly from the cameras and it fits a 2D Gaussian profile to the data obtained from the beam. This program also calculates several characteristics of the beam such as the position x and y, luminosity, and pixel with maximum luminosity. This program was coded in C++.

3.2 BPMServerCommon

This program obtains data from all available hardware servers and sends necessary processed data to the clients. This allows all the clients to have one contact point to all available BPMs. This program was coded in Java.

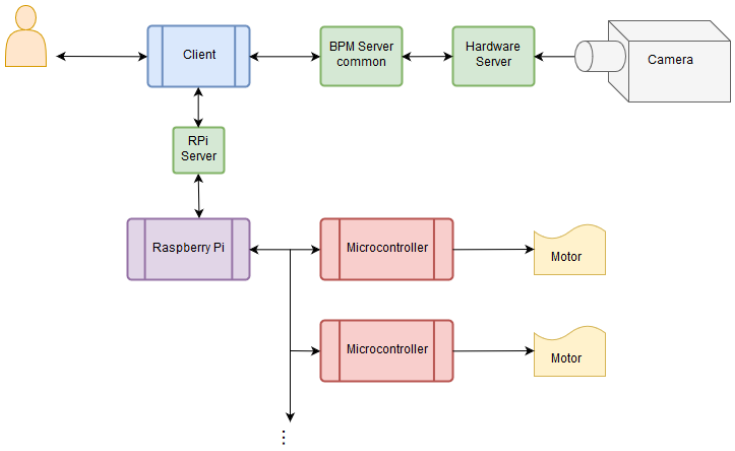


Fig 8 Network diagram.

3.3 BPMClient

This program shows the captured and processed image of the beam as obtained by the camera and processed by BPMServerHard. It displays plenty of information regarding the current frame as well as allowing to save the individual frames. This program was coded in Java.

3.4 Main Client

This program communicates the camera (i.e. the BPM programs) with the user and the Raspberry Pi Server. From its GUI, the user can manually move the camera, obtain plots that display the behavior of several different properties of the beam, and execute the autofocus method. This program was coded in Java.

3.5 Raspberry Pi Server

This program is responsible for receiving the instructions from the Main client and execute the appropriate movements of the motor, as well as requesting more instructions once it is done with the current task. This program was coded in Java.

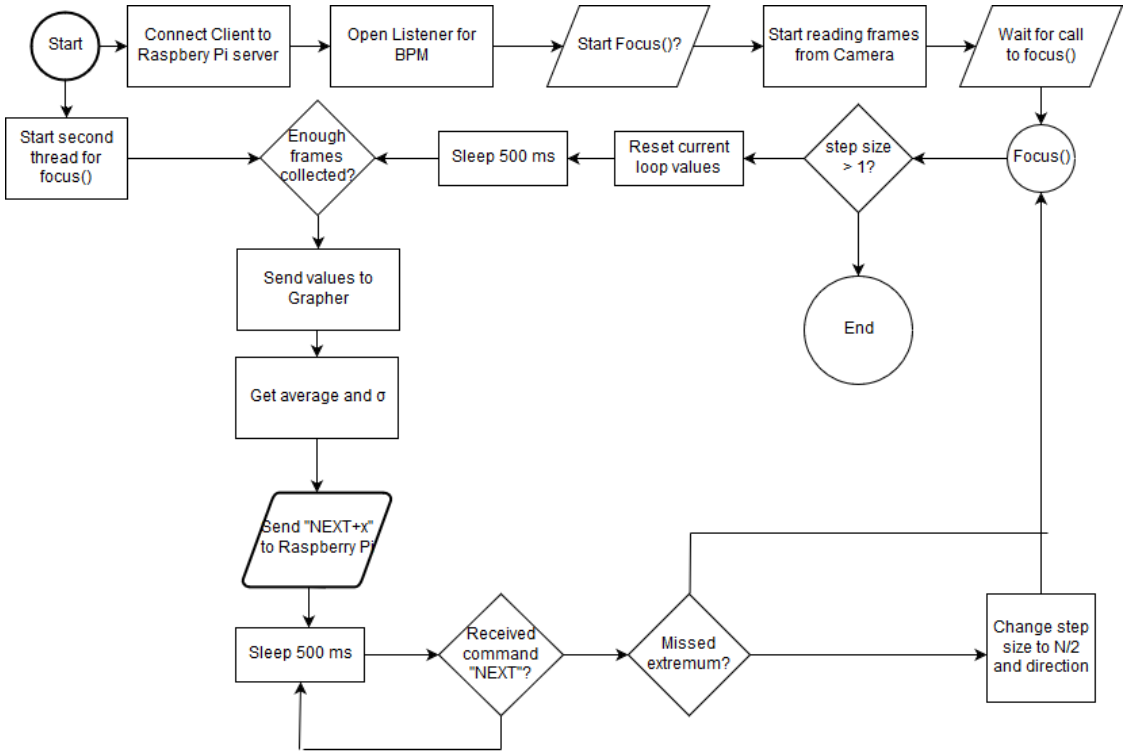


Fig 9 Flowchart diagram of client software.

3.6

4 Software

The software developed for this project was designed using Java and it is divided in to three main parts: the client and General User Interface (GUI), the motor server and the camera server. This last one is composed of the BPM programs.

4.1 Client and GUI

The purpose of the client is to display the GUI, control the communication between the user and the servers, as well as well as mediating the communication between the two servers. This program was coded in Java.

The client begins by initializing a second thread to run along with the GUI so the plots can be

updated while the camera is being focused in real time. This second thread will just wait for the instruction to start the focusing algorithm. Then, the main thread will start receiving frames from the camera server. When prompted to start, the focusing algorithm will be checking that the step size is greater than a threshold (e.g. 10 steps or more). It will reset the current loop's values and wait until a specific amount `N` of frames is collected from the camera. Once the desired amount of frames is collected, the values will be analyzed and sent to the Grapher object to update the plots. The next step is sending the Raspberry Pi server the `MOVE` command with `x` steps in direction `D`. Next, the client will wait until it gets the `NEXT` command from the server to continue. Assuming a smooth function with one extremum for focus, it will test for the criterion and, if the camera is moving in the right direction, it will just loop again. If the camera missed the extremum, it will change the direction to that opposite to `D` and reduce the amount of steps by the desired amount (e.g. `N/2` or `N - 10`) and loop again.

4.2 Raspberry Pi server

The purpose of the server in the Raspberry Pi is just to control when to move the camera and to ask for more instructions.

First, the server will open the socket to allow the client to connect. When a client is connected, it will activate the GPIO pins in the Raspberry Pi and put them in stand by mode so current is not flowing through the motor, since it can overheat and melt the stand. The next step is starting the loop, whose first step is to wait for input from the client. Once it receives the `MOVE` command from the client, it will enable the pins. Depending on the direction indicated by the `MOVE` command, the server will turn on or off the pin in charge of controlling the direction of the motor's rotation. Next, the motor will move one step and loop the necessary amount of times, as indicated by the

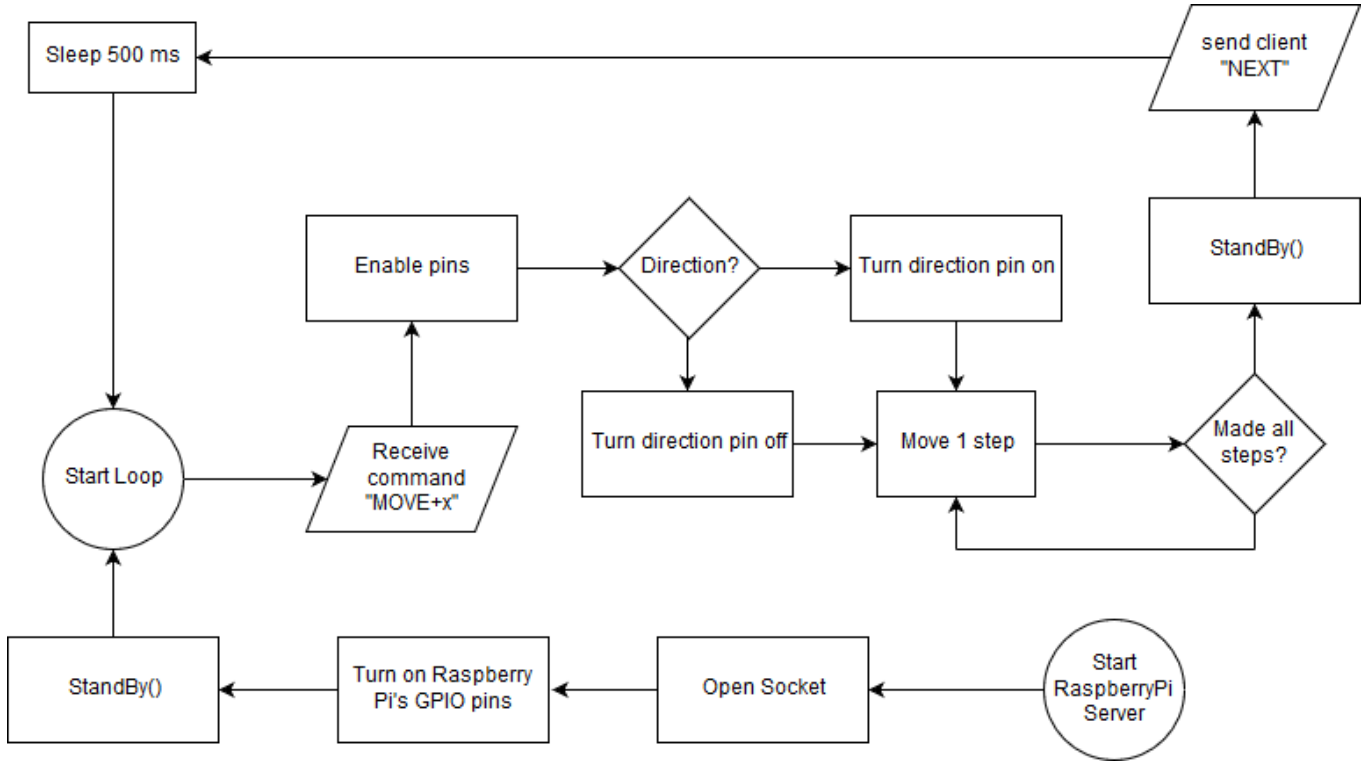


Fig 10 Flowchart diagram of server software.

Client's command. Once all the loops (i.e. the steps) have been performed, all the pins will be set in stand by mode, and send the Client the authorization to continue with the next step. Finally, the server will loop and wait for the next command.

4.3 Protocol

The Client and the Raspberry Pi server communicate via a simple string-based protocol, the commands are:

4.3.1 `MOVE+x`

This command is composed of 3 parts. The first part of the string is the substring `MOVE` which tells the server that the instruction will be to move the motor somehow. The second part is a symbol; either + or −, depending on the direction. Finally, `x` is just the number of steps to be performed.

An example of a complete command would be `MOVE+20` or `MOVE-1`.

4.3.2 NEXT

This command is sent from the server to the client. It tells it to continue with the next step and break the sleep loop.

4.3.3 SHUTDOWN

This command is sent from the client to the server. It simply tells the server to shutdown the pins, close the socket, and shutdown the program.

5 Algorithms and Measures

This system uses passive focusing techniques to obtain the sharpest image possible in the sensor. Passive focusing techniques are especially useful since they depend uniquely on the information of the images collected by the monitor and no external devices are needed, as is the case with active focusing techniques. (1,2) There were several algorithms chosen for this project to compare their efficiency.

5.1 Preliminary analysis

To assure that the experiment was running properly, we ran a test to plot the constant quantity

$$K = \mathcal{L}ab$$

where $\mathcal{L}$ is the luminosity of the beam, and a and b are the minor and major axes respectively.

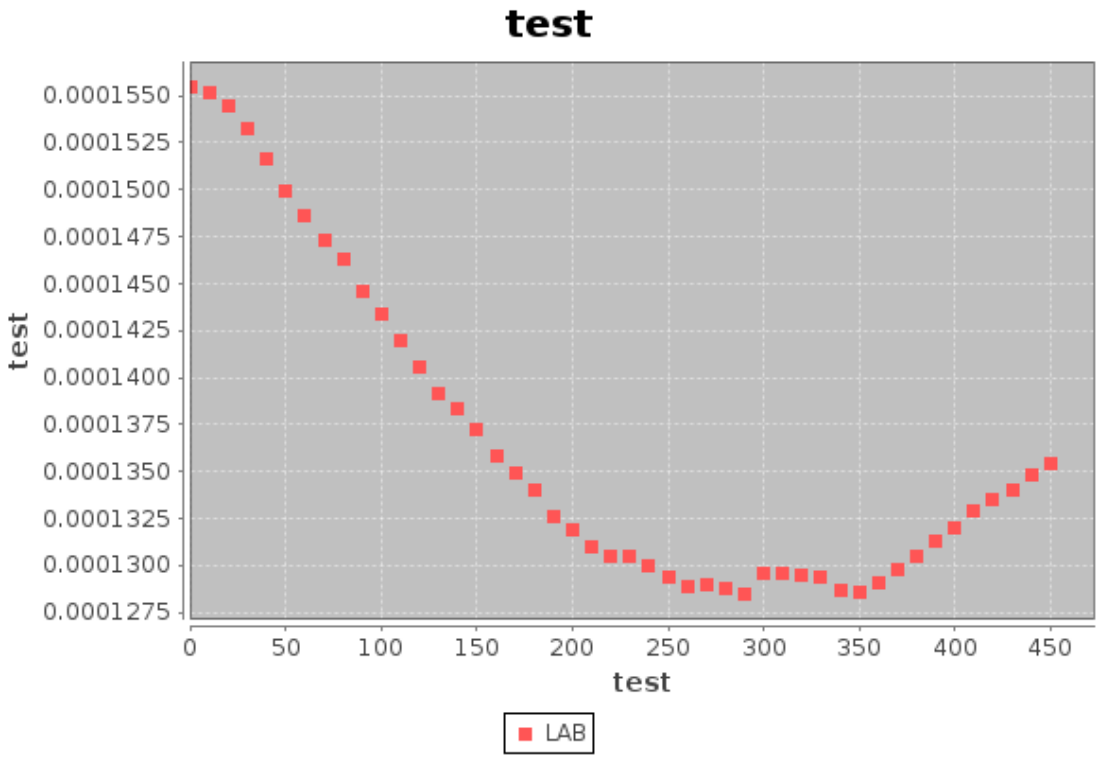


Fig 11 Plot representing the constant quantity $K = \mathcal{L}ab$

At first sight, this plot seems to represent a behavior which is all but constant. Nevertheless, upon closer inspection one can appreciate that the maximum difference for the values of K is $\Delta K_{max} \approx 2.75 \cdot 10^{-5}$. Thus, we can assume that the value K remains practically constant.

5.2 Main properties of beam

Some of the measures we took for the sharpness of the image were the main properties of the beam.

The minor and major axes were chosen because at the point of better focus, most of the light will be concentrated in a small ellipsoid somewhere in the image. Thus, the axes will be smaller at the sharpest location of the camera.

Similarly, and due to the same reason as above, the point of best focus will concentrate the most amount of photons in a small area, thus this image will present a higher luminosity than the other, less focused pictures.

Lastly, we opted for also obtaining two of the second moments of the beam according to the formulas

$$I_{xx} = a^2 \cos^2 \theta + b^2 \sin^2 \theta$$

$$I_{yy} = a^2 \sin^2 \theta + b^2 \cos^2 \theta$$

where θ represents the tilt of the ellipse with respect to the x axis and a and b are the major and minor axes respectively.

This decision was due to the fact that, when obtaining the values for the axes, the beam's ellipsoidal shape would change slightly in a wobbling manner, which at some point made the (original) major axis smaller than the (original) minor axis. Thus, showing the sudden shift in position on the plot. Contrarily to this, the plot of the second moments does not present this phenomenon.

5.3 Tenenbaum and Periwitt algorithms

The Tenenbaum algorithm is a 2D spatial measurement algorithm that uses convolution to study image gradients. This algorithm consists of convoluting the matrix A, in which each entry represents the measured intensity $I(i, j)$ of each pixel on position (i, j) of an image, with the Sobel matrices (also known as Sobel kernels).

$$S_x = \begin{pmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{pmatrix} ; S_y = S_x^T = \begin{pmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{pmatrix}$$

We declare the measure M of the quality of the image to be

$$M_{image} = \sum_{i=1}^{w} \sum_{j=1}^{h} G_x(i,j)^2 + G_y(i,j)^2$$

$$G_x(i,j) = I(i,j) \bigotimes S_x$$

$$G_y(i,j) = I(i,j) \bigotimes S_y$$

where h and w represent height and width in pixels respectively, and $\bigotimes$ represents matrix convolution.

The Periwitt algorithm works in the exact same way as the Tenengrad algorithm, but instead it uses the kernels

$$S_x = \begin{pmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{pmatrix} ; S_y = S_x^T = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{pmatrix}$$

5.4 Inverse product of axes

Another measure we used was that of the inverse of the product of the major and minor axes of the beam ellipsoid on the image, or

$$M_{image} = \frac{1}{ab}$$

From the previously mentioned constant relationship $K = \mathcal{L}ab$ it is expected that the behavior of M_{image} will be similar to that of measuring it with respect to the luminosity since in the equation

we can solve for luminosity, obtaining $\frac{K}{ab} = \mathcal{L}$.

5.5 Image Entropy

The image entropy of an image is defined by Shannon's information entropy as

$$S = \frac{1}{\log_2 n} \sum_{i=1}^{n} p_i \log_2 p_i$$

where the probability p of each pixel i is defined with respect to the pixel's intensity as

$$p_i = \frac{I_i}{\sum I}$$

for a total of n pixels on the image. The image's information content Info is defined as $\mathrm{Info} = -S$. The closer the position of the camera is to the focus, the more details and contrast appear on the image. Thus, the more information that will be present on the image. This means that the peak of the Info plot is where we should find the best focused image.

6 Results

The following plots represent the many results obtained when scanning the whole range of movement for the camera. In the following plots, the x axis represents the position of the camera, with 0 being the point furthest away from the origin of the beam, right next to the motor, and 460 being the opposite end of the stand (i.e. the one closest to the vertical tube).

From the standard properties of the beam, we found that the five of them have an extremum that indicates a point of best focus and have the shape of a smooth, unimodal function. The problem is mainly that the plots for a, b, I_{xx}, and I_{yy} are not quite clear with the position of the extremum,

since a big chunk of the plot has values really close to each other and the plot appears almost flat. On the other hand, luminosity $\mathcal{L}$ has a much more defined peak.

Both the Periwitt and Tenengram algorithms performed similarly well, suggesting the same position for the extremum as the previous plots. The difference being that this peak is much more defined.

Next, $\frac{1}{ab}$ did behave as expected, which was a similar plot to that of $\mathcal{L}$. Nevertheless, a much more defined maximum can be found in this function than in that of luminosity, also around the same point in the x axis.

Finally, Image information `Info`, presents a local maximum close to the extrema indicated by the other plots. Nevertheless, it also presents higher points close to the motor (step 0), which most surely would cause problems with the autofocusing algorithm, since it can interpret moving towards 0 as being moving in the right direction when the other plots indicated the optimal focus point to be located closer to step 300. At the end, there are four pictures showing the difference between a focused and an unfocused picture taken by the monitor.

Overall, it seems that Luminosity and $\frac{1}{ab}$ are the best parameters to find a precise extremum and, hence, focusing point for the monitor.

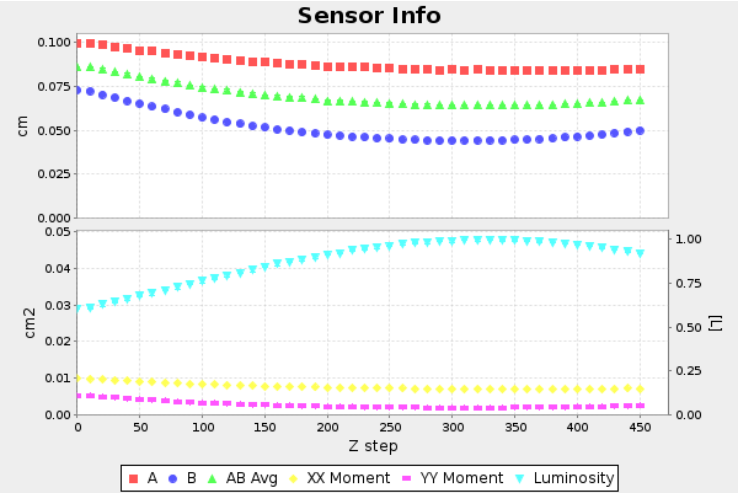
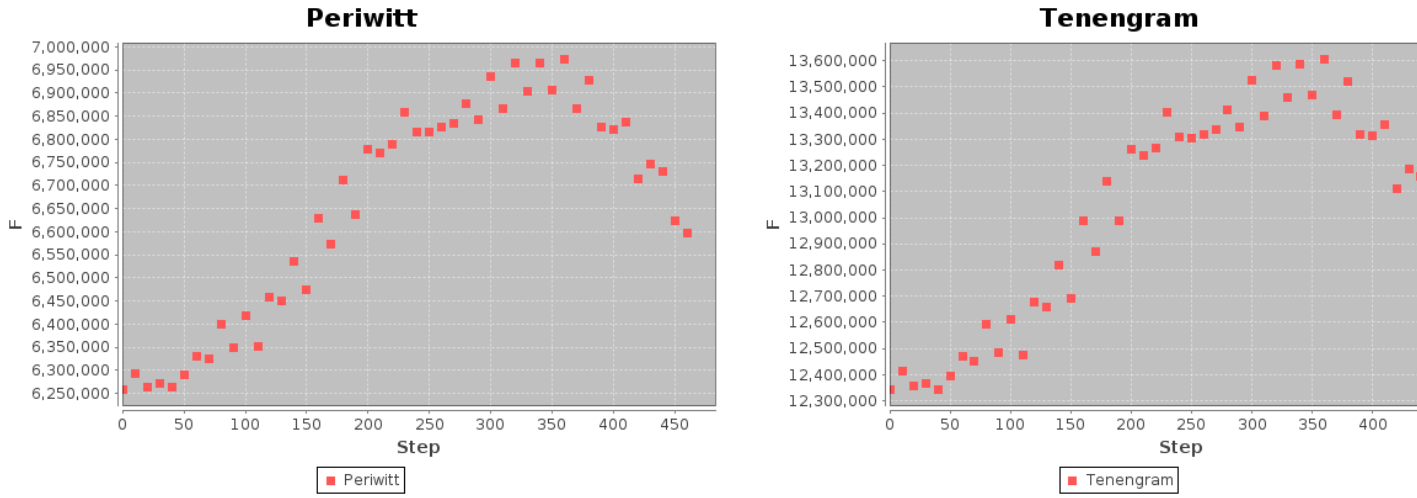


Fig 12 Behavior of a, b, $\mathcal{L}$, I_{xx} and I_{yy}.



(a) (b)

Fig 13 Comparison between Periwitt (a) and Tenengrad (b) algorithms

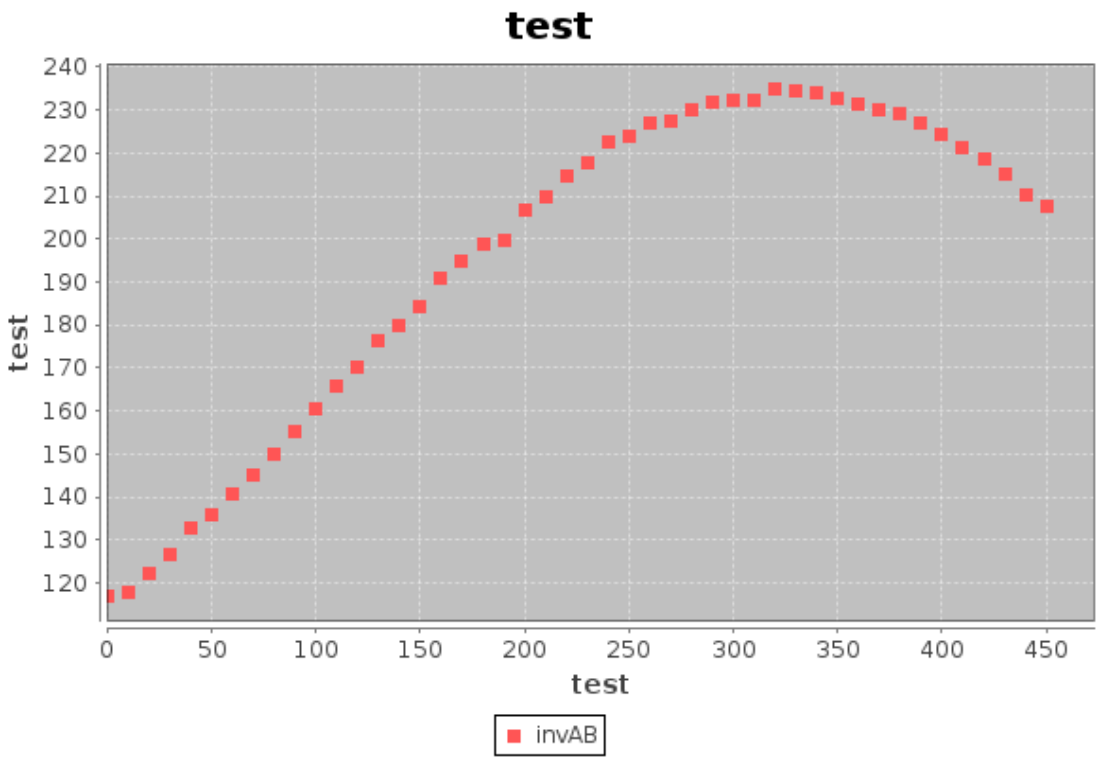


Fig 14 Behavior of $\frac{1}{ab}$.

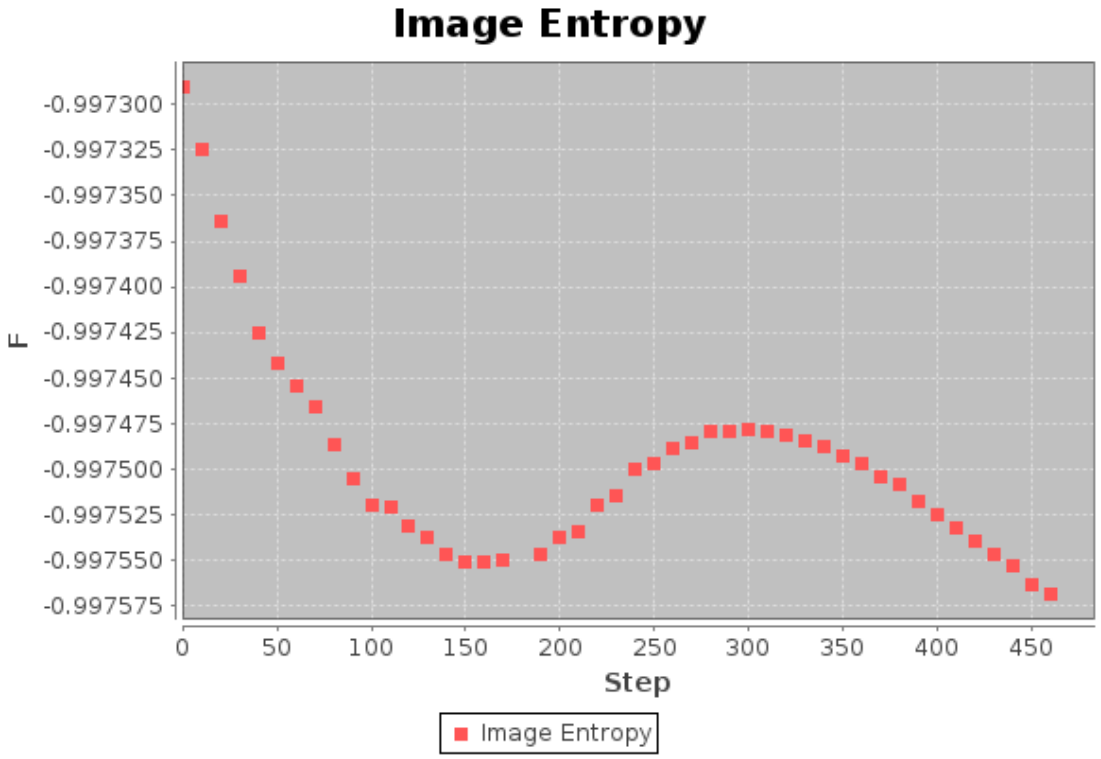


Fig 15 Behavior of Normalized Image Information from Image Entropy.

Fig 16 Unfocused image.

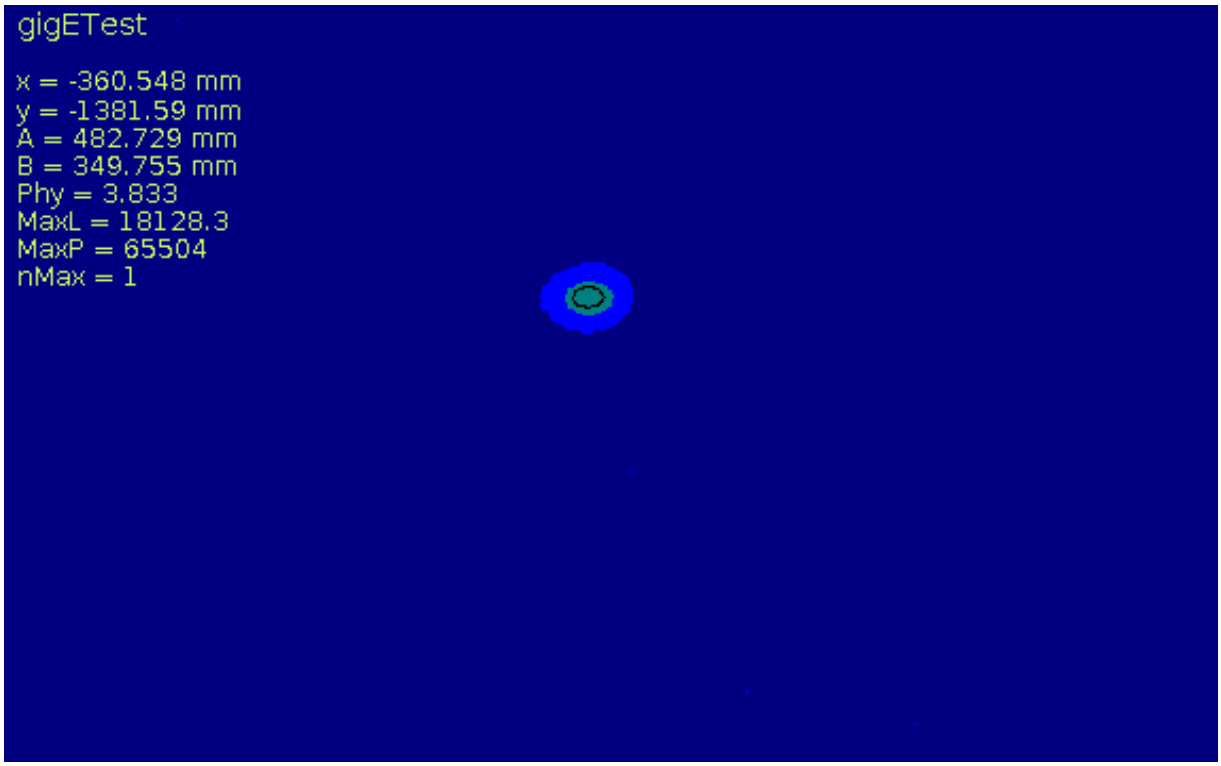


Fig 17 Unfocused image.

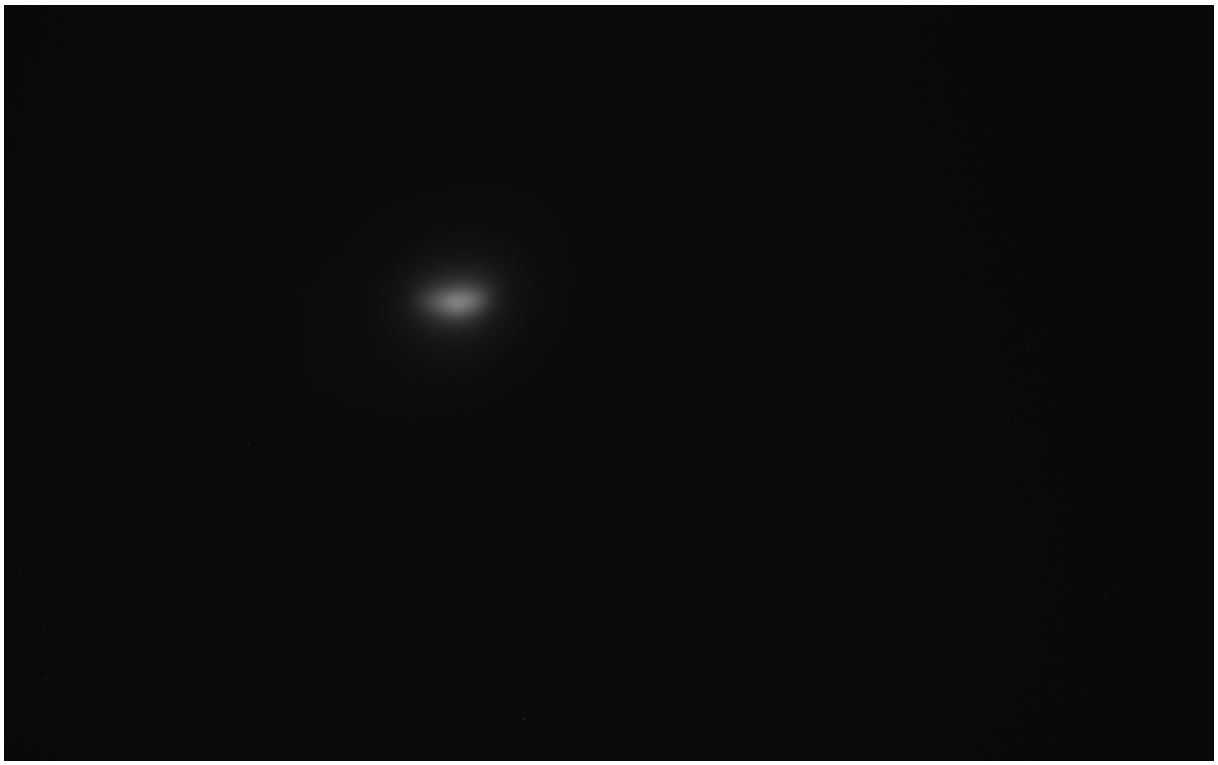


Fig 18 Focused image.

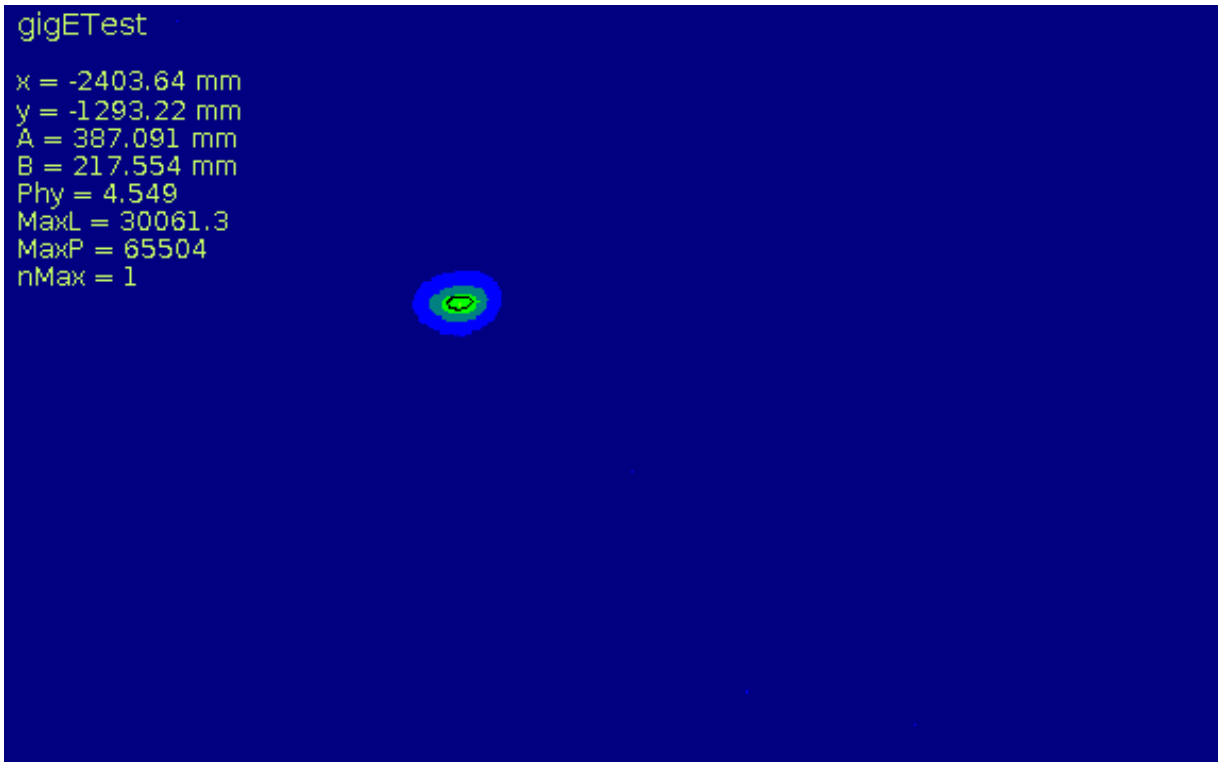


Fig 19 Focused image.

References

Acho, L., Alvarez-Borrego, J., Bueno-Ibarra, M. A., Chavez-Sanchez, M. C., (2005) *Fast Autofocus Algorithm for Automated Microscopes*. Optical engineering 44(6).

Mateos-Perez, J. M., Redonto, R., Nava, R., Valdiviezo, J. C., Cristobal, G., Escalante-Ramirez, B., Ruiz-Serrano, M. J., Pascau, J., Desco, M. (2012) *Comparative Evaluation of Autofocus Algorithms for a Real-time System for Automatic Detection of Mycobacterium Tuberculosis*. Cytometry. 81A.

Santos, A., Ortiz de Solorzano, C., Vaquero, J. J., Peña, J. M., Malpica, N., Del Pozo, F. (1997) *Evaluation of Autofocus functions in Molecular Cytogenetic Analysis*. Journal of Microscopy, 188.

Scarpelli, A., (2017). *Development of a synchrotron radiation beam monitor for the Integrable Optics Test Accelerator*. University of Ferrara.

Xu, X., Wang, Y., Tang, J., Zhang, X., Liu, X. (2011). *Robust Automatic Focus Algorithm for Low Contrast Images Using a New Contrast Measure*. Sensors. 11