Calculating Beam Sizes

From Screen Images

on VELA

Duncan Scott

ASTeC: Accelerator Physics Group

August 2014

**Abstract**

This note details some procedures for calculating a beam size from a screen image, with particular reference to VELA. As well as of general interest this is a necessary procedure required to calculate the emittance using ‘*quad scan’* techniques where multiple beam images are recorded for different upstream quadrupole settings. From the beam size vs quadrupole focussing strength data and a model of beam transport through the accelerator the emittance (and other lattice functions) can be calculated. Essential to this process are robust methods to estimate the beam size. Another goal of this note is to examine the suitability of different methods for an ‘*on-line*’ emittance measurement system. This system would give a quick estimation of the emittance, preferably using fast, simple and robust non-proprietary methods.

# Introduction

Only digital screen images from VELA will be considered and the practical aspects of acquiring and saving raw pixel values will not be discussed. The camera images of the screens are rectangular arrays of 1392 by 1040 pixels, each with 8-bit resolution, (higher resolution is possible but network bandwidth limits currently prohibit this). Correct setting of the camera gain to avoid image saturation is assumed, the total signal from the beam is not of interest here, just some measure of the beam size. Whilst searching the literature it seems that measured emittances are often quoted without an error bar, or at best statistical errors only. An estimation of all the systematic errors that can influence the emittance measurement, including the beam size, would be an area of worthwhile study.

## Outline of procedure

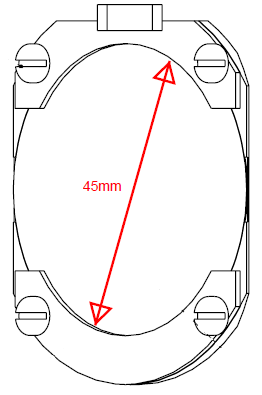
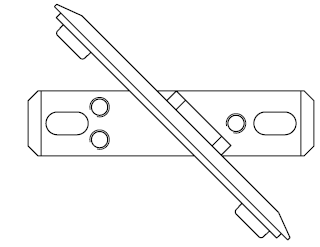
After a description of the basic layout of the camera and screen system some simple pre-processing of the raw data is required:

* The portion of the image that is the screen is selected
* A background signal is subtracted
* A background image is subtracted (i.e. one with no beam, but with dark current).

This pre-processed data can be used to estimate a beam size. Two main methods are discussed, those based on fitting models to the data, in both 1 and 2 dimensions, and calculating moments of the distributions. There are two reasons for investigating the moments of the distribution, first they will be computationally very fast to calculate, so (in some sense) “*free*” and can be used as a check, and they can be used as starting values for model fitting. Also, model fitting techniques require an optimisation algorithm and although very good ones are available on commercial codes such as Mathematica online measurements using non-proprietary software would require a different solution. This could be developing an optimiser, or using external libraries.

# Screen Layout

A general layout of the screen in it’s holder and angle (assumed to be 45o with respect to the cameras) is shown in Figure 1 [[[1]](#footnote-1)].An example camera image from YAG-02 is also shown in Figure 1.



Camera

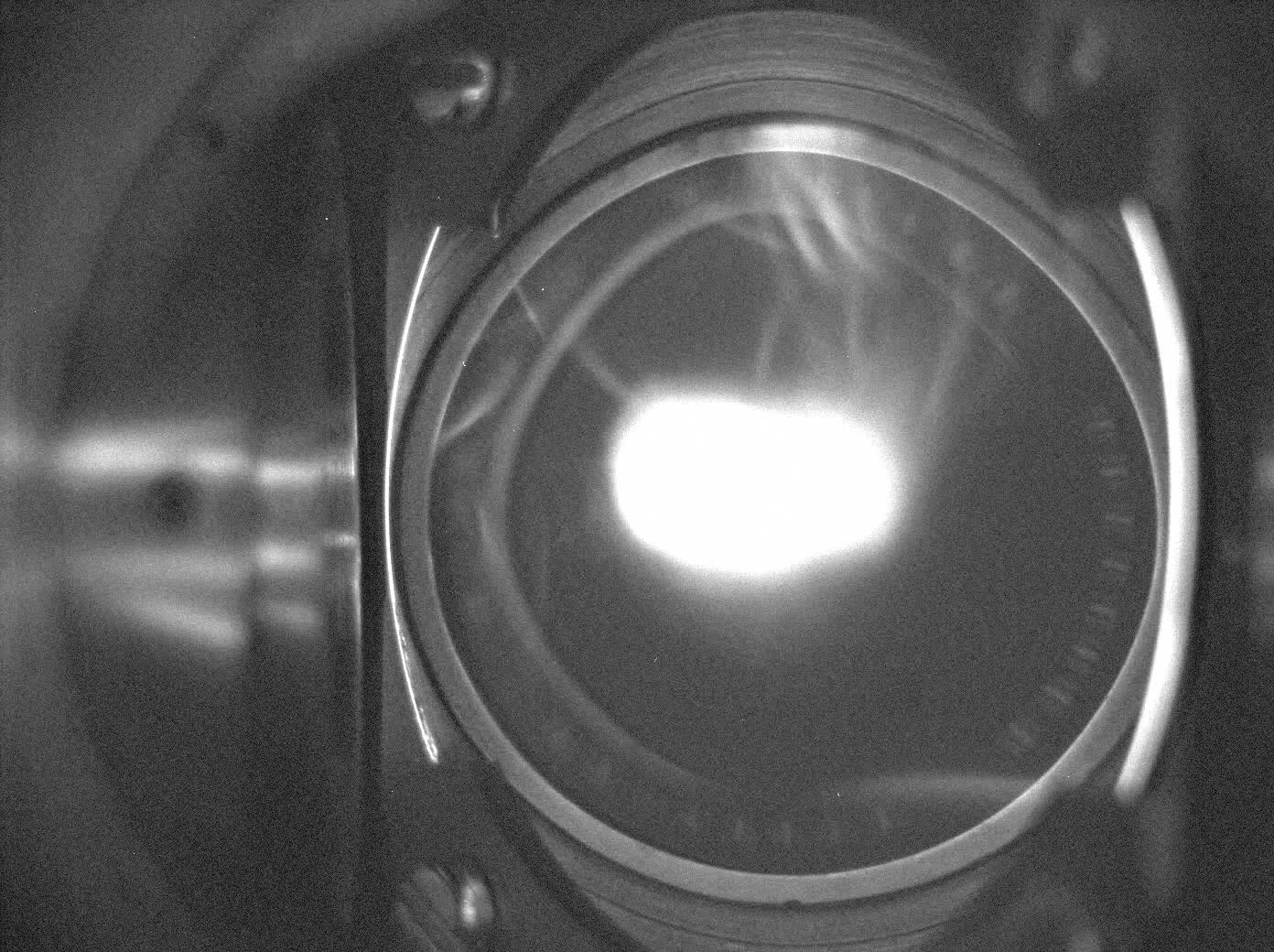
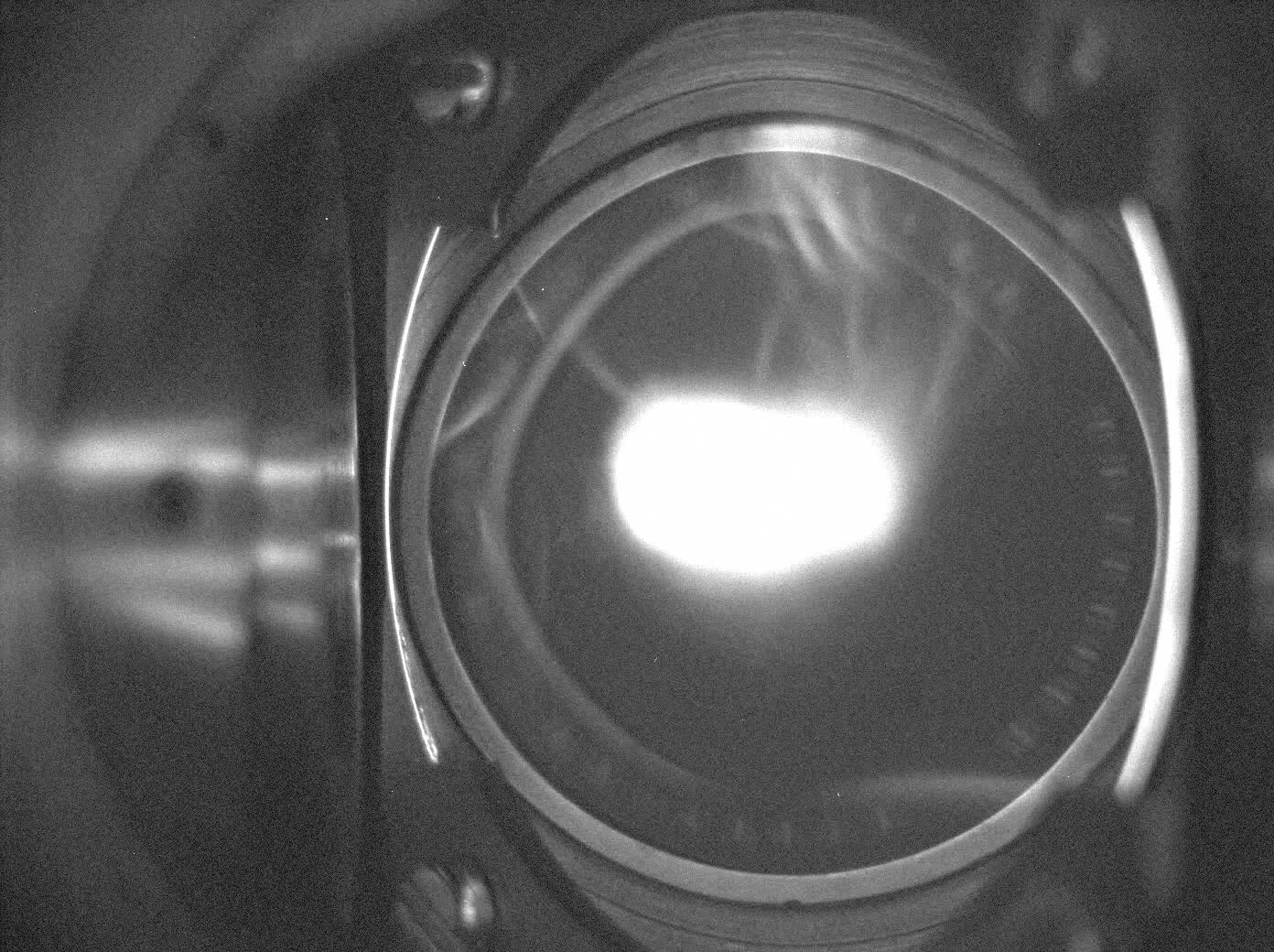


Figure 1: YAG screen in holder at 45o with respect to the camera and an example YAG-02 image.

# Pre-Processing Raw Data

## Selecting the screen image

To select the screen image cuts can be made to a fully illuminated background image (showing the screen) to select precisely where the screen is. In principal this could be done by hand, and perhaps in a environment where the screens and cameras do not move relative to one another this would be the best solution. However, currently this has been done using an elliptical *mask*. Assuming the screen is an ellipse, centred at {x0, y0} with a major and minor axis aligned with the ‘*horizontal’* and ‘*vertical*’ axes, with ‘*radii’ (lengths)*  rx and ry a 1392 by 1040 array is created with all points inside the ellipse equal to 1 and all points outside the ellipse equal to zero (shown in Figure 2). This mask can then be multiplied by any image to give all the data on the screen, and zeroes outside the screen. The image can be cropped between x0 - rx and x0 + rx, and y0 - ry and y0 + ry to reduce the amount of data to be analysed. Once the mask is computed it is simple and fast to multiply it by the image, speeding up the image cropping process. An example of the ellipse used to define the mask, with parameters, and the cut applied to the background image is shown in Figure 3. There is clearly some judgement in where to make the cut, for this analysis the cut is deliberately smaller than the screen, this is because reflections from the beam spot show up around the screen edge giving spurious data, although this does mean that some of the beam data is cropped in some images.

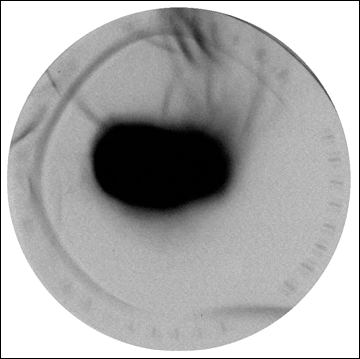
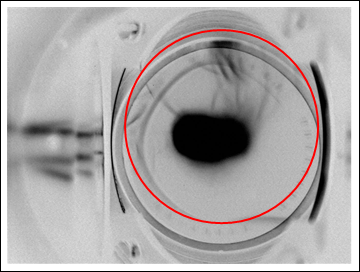


**{x0,y0}**

**rx**

**ry**

Figure 2: elliptical mask definition.

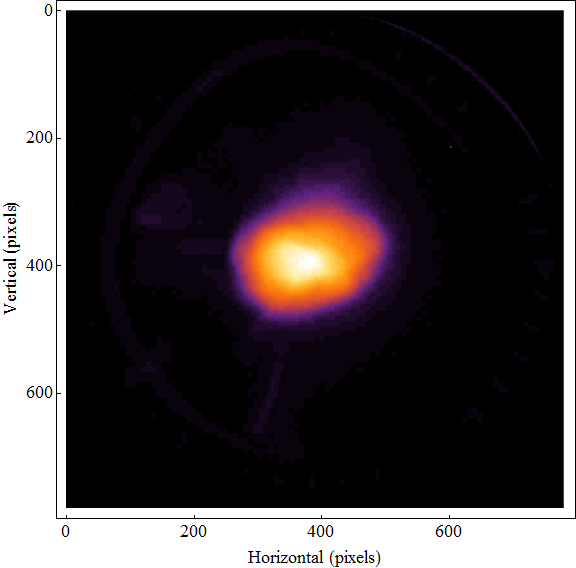
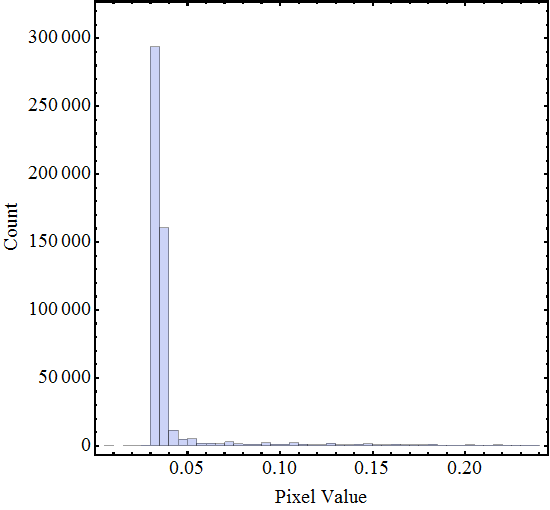


|  |  |
| --- | --- |
| Parameter | Value [Pixels] |
| x0 | 865 |
| y0 | 560 |
| rx | 390 |
| ry | 390 |

Figure 3: Example mask, with parameters applied to background image.

## Background Signal Subtraction

To remove the background it was assumed most of the data is background and not beam. Of the 500 000 active pixels in the cropped image the commonest values are assumed to be the background levels, a histogram showing the distribution of the pixel values for a typical image is shown in Figure 4. For this analysis the commonest pixel value was subtracted from each pixel and negative pixels were set to zero (the beam has no ‘*negatively’* charged component). It would not be too difficult to implement a more sophisticated analysis looking at the distribution of background pixel values and fitting a suitable function, but this would be more computationally expensive. Figure 4 shows a typical beam image after the cuts and the background is subtracted. There are still plenty of artefacts that are attributed to reflections, applying the screen mask and background subtraction. Further study should look at reducing the number of these artefacts.



Artefacts

Figure 4: Histogram of pixel values in typical cropped screen image Example typical beam image after simple pre-processing.

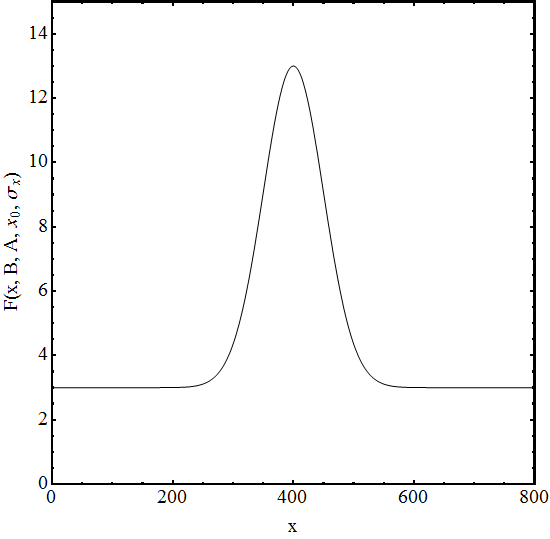
## Background Image Subtraction

Good quality data will have an associated dark current image. Dark current images should be taken with the same set-up but the beam off (e.g. with the laser shutter closed). These dark current images should be processed in a similar manner, with cuts and background signal subtracted, then subtracted from the beam image.

# Calculating a Beam Size

## Model Fitting Methods

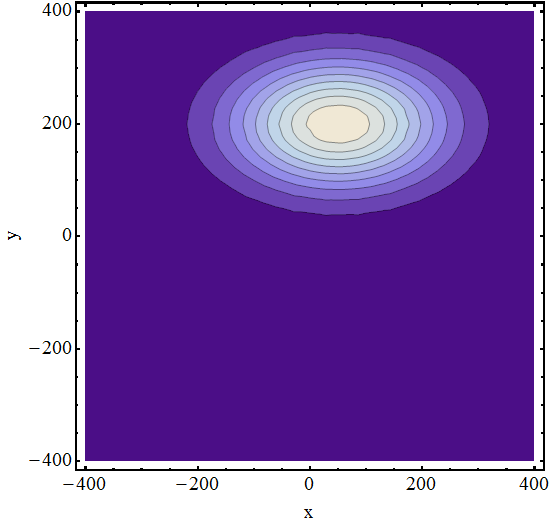
In these methods a least squares minimisation is performed between the data and a model function representing the expected distribution. If the pixel values are projected onto the x and y axes (i.e. summed along each column or row) then 1-D models can be fitted, or a 2-D model can be fitted to the actual data. Typically normal distributions are used for the model, three different models have been considered. For completeness, the parameters are defined in figures, and are an offset and scaling factor, and are the expected values and and are sample variances and is a clockwise rotation between the axis and the direction.



*B*

*A*

*x0*

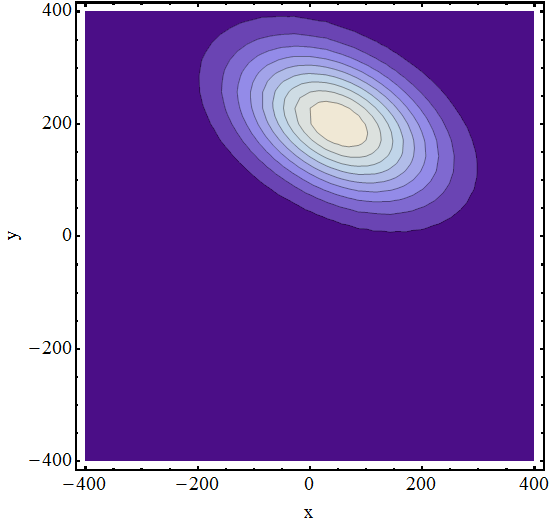


*x0*

*y0*

1-D normal distribution

2-D normal distribution



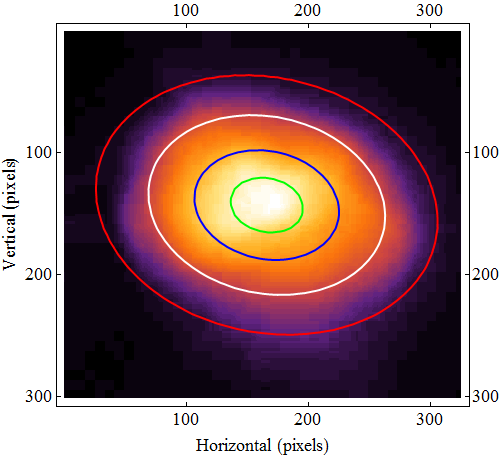
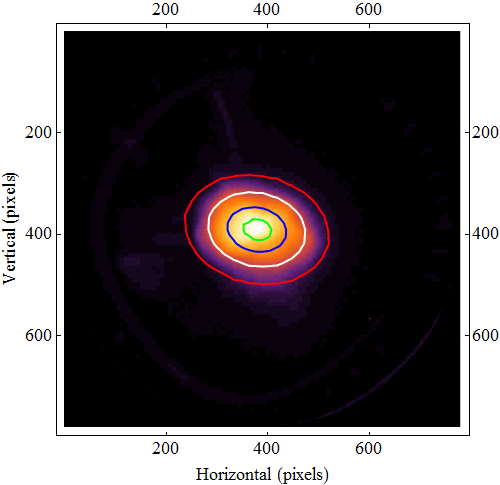
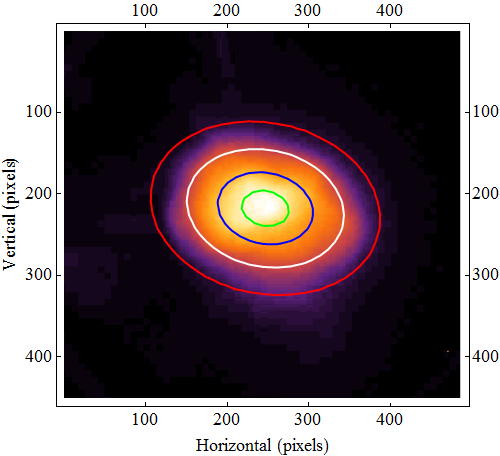
*y0*

*x0*

2-D rotated normal distribution

Fitting to the models is done by a simple least squares, or *chi squared minimization procedure*. For example, for the rotated normal distribution, a model with 7 free parameters, the sum that must be minimised is[[2]](#footnote-2):

where is the data value at . Good initial guesses for the means and sigmas are useful to decrease the number of iterations required in the minimisation process. Often they can be well estimated from the data, such as the amplitude A and the offset B. For higher dimensional fits good estimates can come from fitting 1-D projections first. One advantage of the 2-D fits is that the beam size does not need to be entirely contained on the screen to get a good fit, Figure 5.



|  |  |  |
| --- | --- | --- |
|  | σy | σx |
| Bottom Right | 48.1 | 70.3 |
| Bottom Left | 48.5 | 65.7 |
| Top Right | 47.4 | 64.4 |
| Top Left | 47.1 | 64.1 |

Figure 5: Four different fits to the same data cut by increasing amounts fitted rms values.

## Moments of the distribution

This is the fastest to implement, taking a few microseconds to calculate for a typical image, however backgrounds and noise can greatly influence the answer. Another problem with this method is that the entire distribution needs to be captured in the image, for model fitting this need not be the case. The projections on the horizontal and vertical axis are taken and from these 1-D distribution the variance is calculated using the well-known expression:

where is the distribution being considered, and the function is the expected value of .[[3]](#footnote-3)

## Maximum Likelihood Estimation

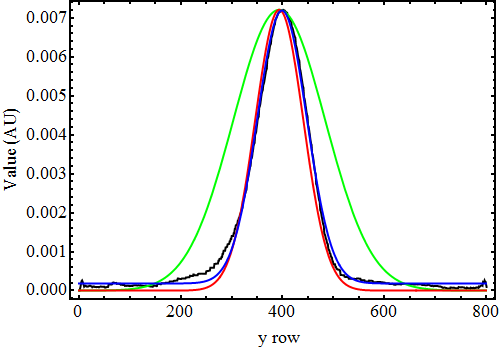
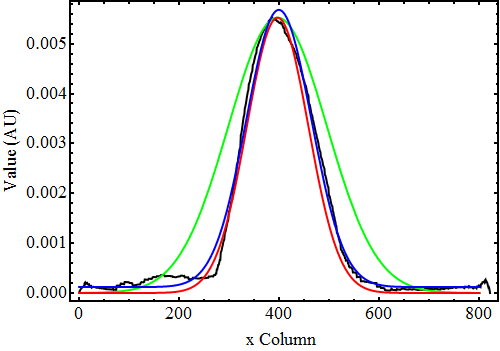
A multivariate normal distribution can be completely described by mean values and a covariance matrix. Therefore, finding these for the image data will give parameters for a normal distribution that are *optimal* in the sense of being *the normal distribution that is the Maximum Likelihood Estimation of the data*. The covariance matrix of the screen images is simple to calculate in codes such as Mathematica, and Appendix I gives an outline of solution that can be easily coded using just the raw data and simple matrix operations.

## Truncating Distributions

To get a better estimation of the beam size from the moments and MLE methods it has been found that truncating the distributions at ± the Full Width Half Maximum about the peak value gives *a better* fit. Incorporating this technique is still computationally inexpensive.

## Results

Results for the projections of the test image are shown in Figure 6, here the moments expected value and variance has been used as parameters in a normal distribution to help compare with the 1D fitting.



Black – data, Green – Moments, Red – Truncated Moments, Blue – Normal Distribution Fit

Figure 6: fitted normal distributions and raw data

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Projection on X | | | Projection on Y | | |
|  | 1D Fit | Moment | Truncated  Moment | 1D Fit | Moment | Truncated  Moment |
| rms | 64.7 | 98.4 | 61.4 | 48.6 | 90.3 | 47.4 |
| mean | 399.1 | 396.6 | 396.6 | 398.9 | 393.7 | 393.7 |
| error | 1.73 | 7.71 | 2.66 | 1.35 | 13.3 | 2.78 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 2D Fit | Rotated 2D Fit | MLE | Truncated MLE |
| x rms | 62.9 | 63.9 | 231 | 67.8 |
| y rms | 47.4 | 47.0 | 90.0 | 46.5 |
| x mean | 394 | 394 | 400 | 399 |
| y mean | 369 | 369 | 393 | 368 |
| ϴ | 0 | 0.2 | 0.004 | 0.0005 |

## Beam size Error Estimate

Currently no suitable method has been developed to generate an error in the beam size from the fitting or covariance matrix, this is the subject of further study. However, a statistical error can be obtained by looking at multiple measurements

# Converting from pixels to mm

With the current screens in VELA this is not easy. Practically a conversion is arrived at by assuming the inner diameter of the bezel screen holder is 45 mm and then points chosen in the horizontal and vertical directions that define this diameter. Screen rotations about the beam axis will give an slight error to this measurement. For the example screen here the bezel edges are past the field of view of the camera, the positions of the screws has been used to judge the extra number of pixels required. As a check if the screen was aligned at 45o to the screen then the number of vertical pixels should be a factor of greater than the number of horizontal pixels for the same physical distance. These measurements by eye are quite difficult and should probably be checked by multiple people.

|  |  |  |
| --- | --- | --- |
| Direction | Pixels | mm |
| Horizontal | 838 | 45 |
| Vertical | 1130 | 45 |

# APPENDIX Efficient Computation of the Covariance Matrix

The 2D array of data, is represented by a 1D vector , with each point normalised by the sum of all the data, is a vector giving the column number for each entry in , is a vector giving the row number for each entry in , i.e.:

The expectation values, , are now simply dot products:

from which it is easy to calculate the elements of the covariance matrix:

The variance of the distribtution are the Eignevalues of the covariance matrix, or for the marginal distribution the diagonal of the covariance matrix can be taken. The Eigenvectors can be used to calculate the angel of rotation of the distribution, and some care must be taken to work in the correct quadrant when taking an Arctan, especially when comparing to the rotated normal distribution presented here, which has a clockwise positive definition of rotation angle.

1. YAG Holder Assembly Drawing No A2-256-10097 [↑](#footnote-ref-1)
2. In Mathematica the minimisation is conveniently done with NonlinearModelFit function, which effectively calls Mathematica’s minimizer. [↑](#footnote-ref-2)
3. The expected value and square root of the variance make excellent initial guesses for the mean and of a 1D normal distribution fit. [↑](#footnote-ref-3)