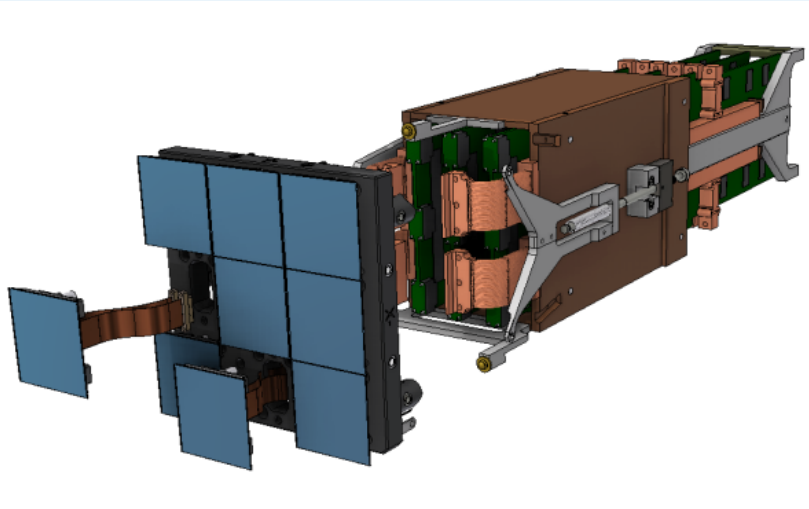
Development of an algorithm to find edges of CCD sensors optimized for the I&T of LSST

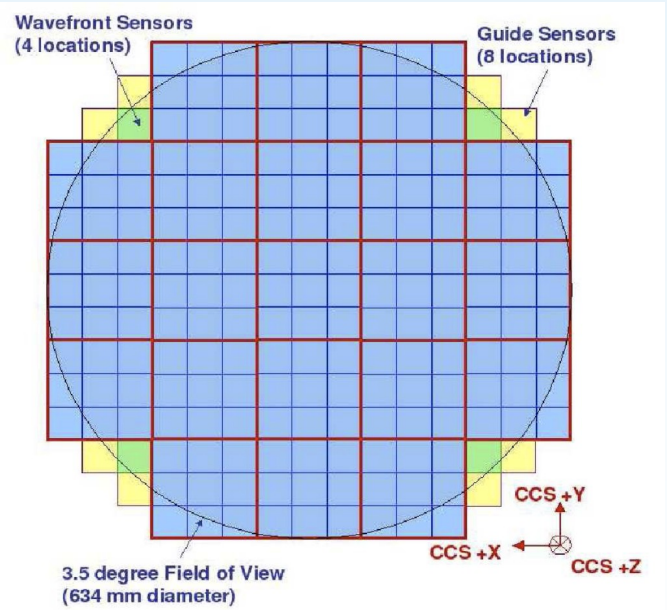
Giorgio Dho

Supervisor:
Scott Newbry

Introduction

- The LSST camera is made of modules: rafts
- The camera focal plane is made of mainly 21 rafts
- The camera focal plane is realized putting the raft one next to the other
- Each one is made of 9 CCDs of 16 MP each and their electronics



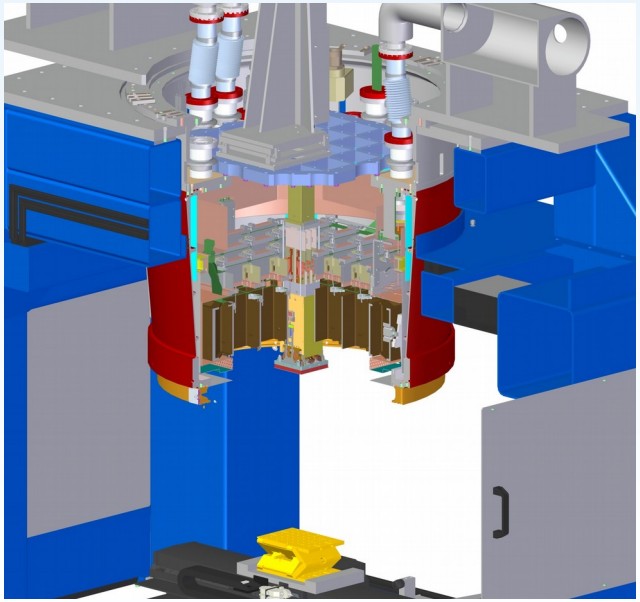


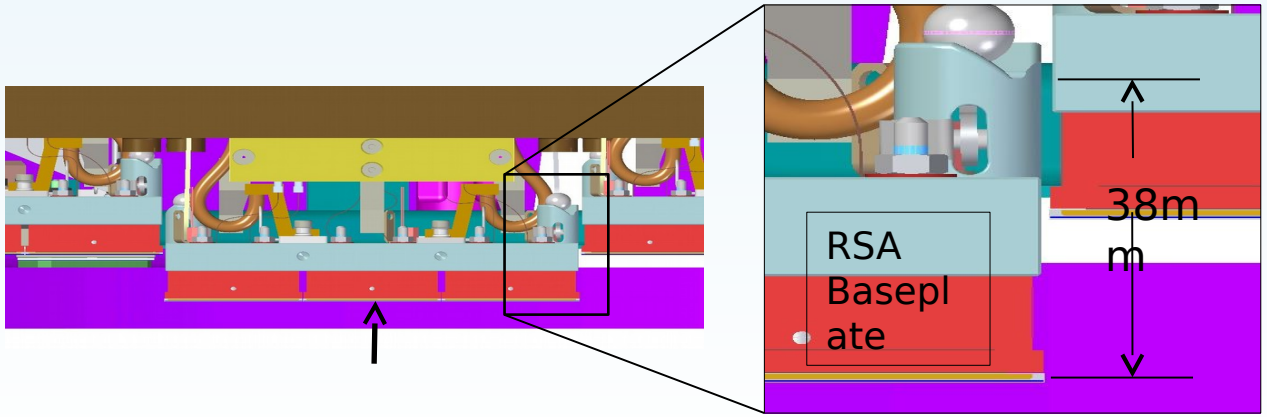
Introduction

- During the integration each raft will be pulled up in position one next to the other

- The rafts must be placed with extreme precision to avoid one to bump into another one causing:
  - Loss of great amount of money because of their cost
  - Delays in the construction of the camera

- The main features of the measurements are:
  - Gap width down to 125 μm (expeted 500 μm)
  - The depth of field needed is 38 mm
- All these make it a tough measure

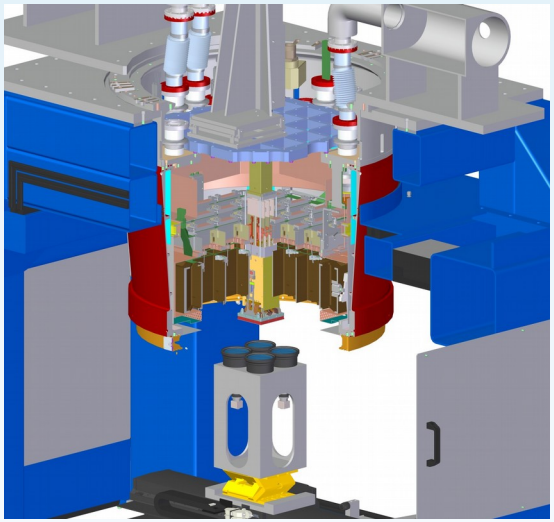




Introduction

- An optic system is needed to properly guide the integration of the rafts and to make sure they do not collide while being installed

- The group came out with the idea of four cameras equipped with telecentric lenses looking at the corners of the raft to be installed, since therre was not any commercial item to do this measurement



- Need to recognize from the each image location of the edges and position when compared to the others (angle and distance)

- Need of an algorithm to find edges of sensors from image to process and get information

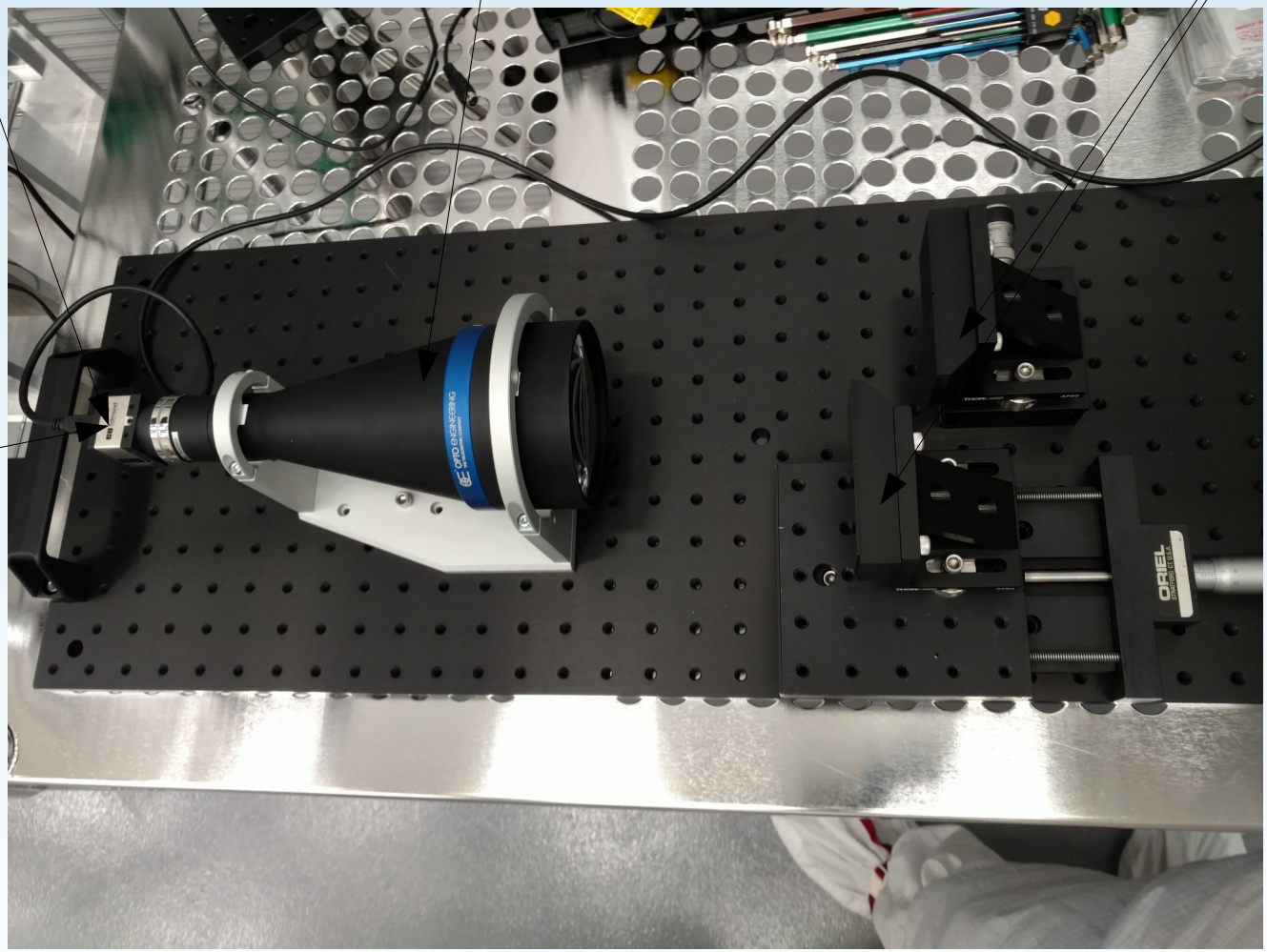
Initial instruments

Gray scale camera
(RT-mvBF3-2051aG)

Telecentric lenses
(TC23064)

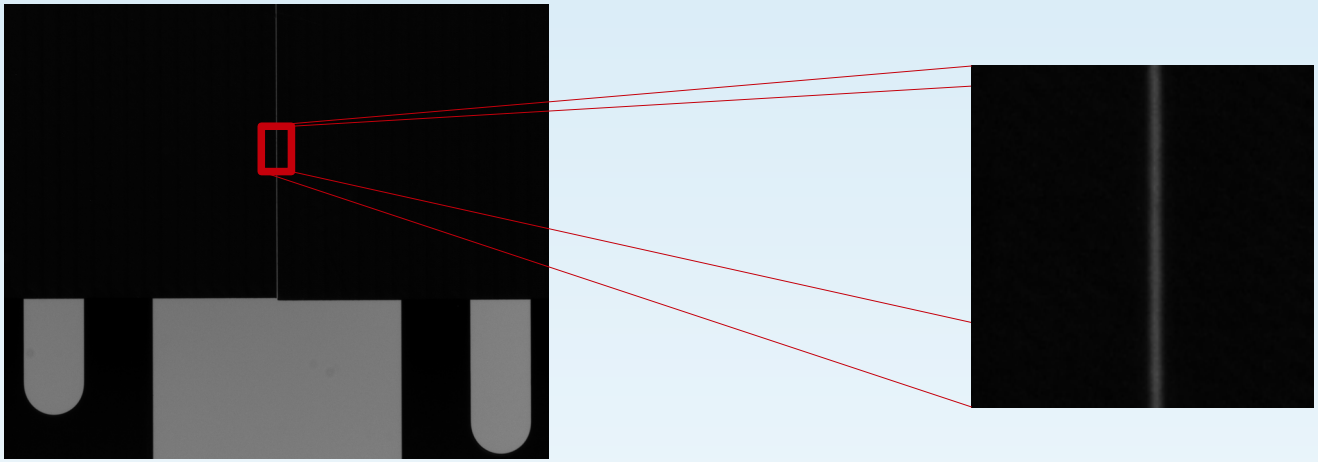
Plastic blocks

Colour
camera
EO-10012C

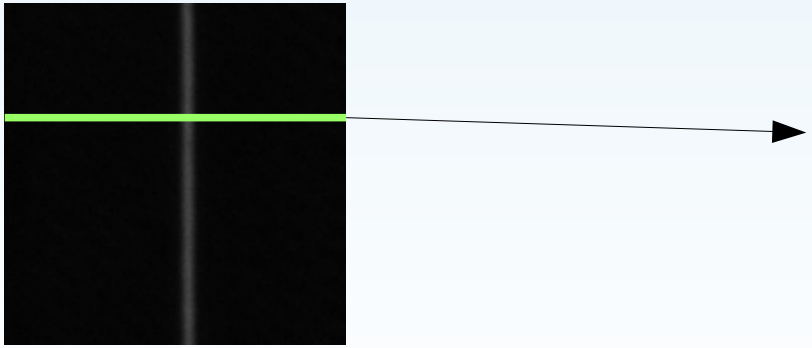


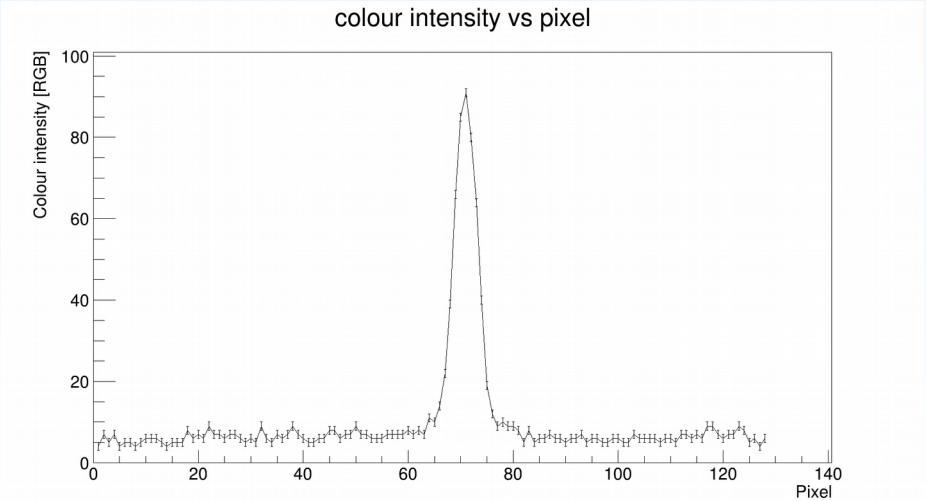
Work with blocks – image analysis

The image is cropped by an image editor down to 128x128 pixel picture.



Then it is translated into RGB values to get information





Work with blocks – function choice

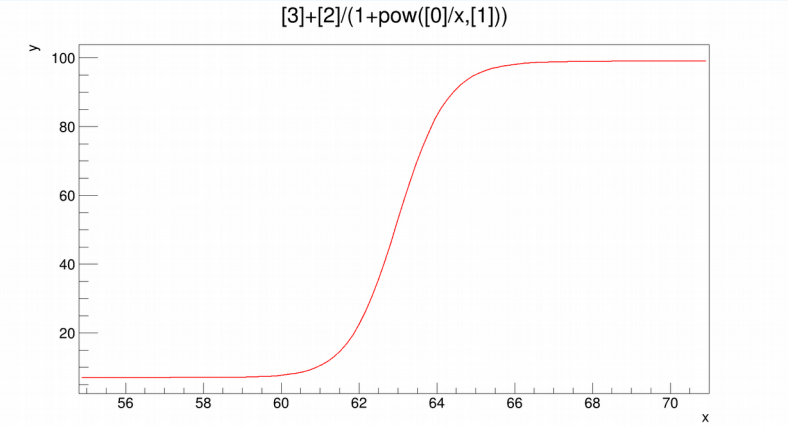
- The edges of the blocks are expected in the middle between bright and dark regions as an effect of the camera resolution and light diffraction

- For the program to be automatic and not knowing the exact function the data follow several functions were tested (gaussian integral, RC filter function..)

- In the end the most versatile was shown to be the S-function

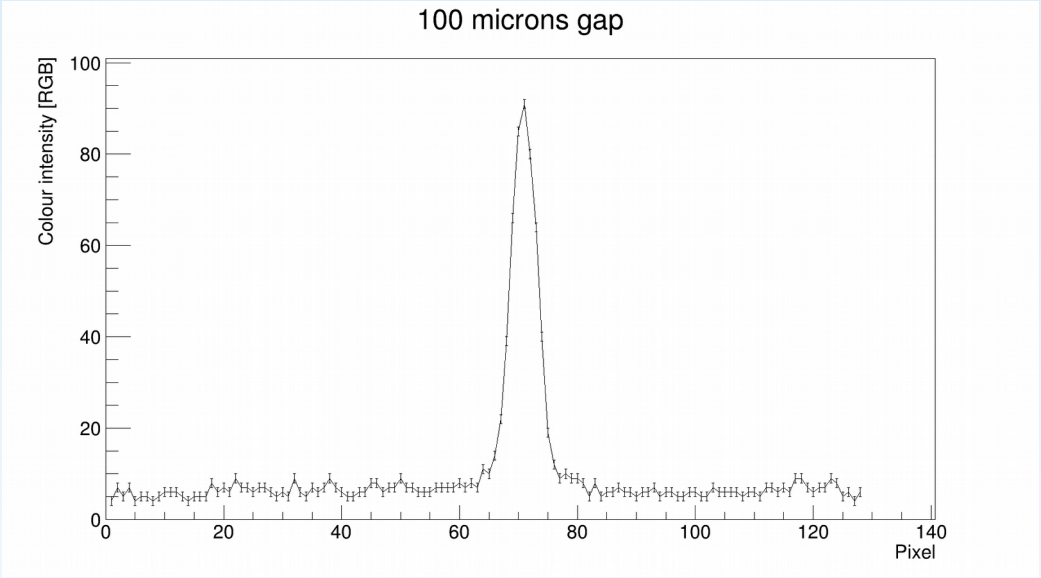
$$f(x) = a + \frac{b}{1 + \left(\frac{c}{x}\right)^{d}}$$

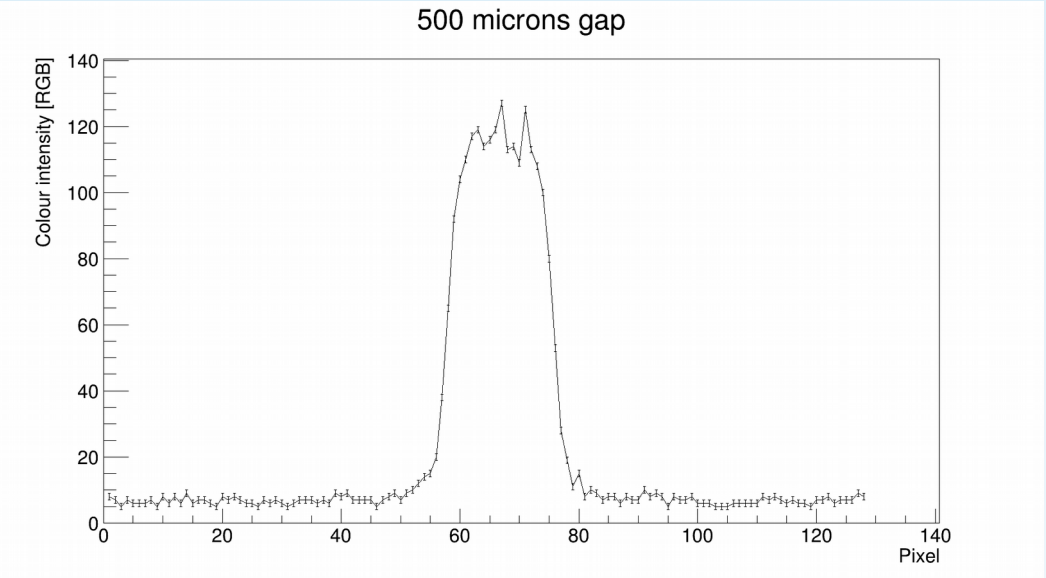
  - a: vertical offset
  - b: amplitude
  - c: x value corresponding to where half of the slope is reached
  - d: shape factor which determines how steep the curve is



Work with blocks – Algorithm

- According to the distance between the blocks the shape of the data changes:

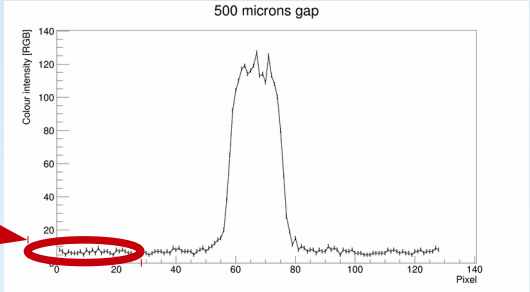




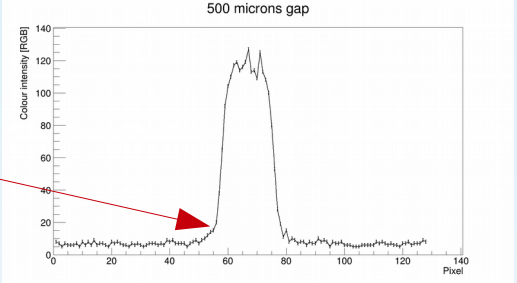
- Necessary to decide whether there is a kind of plateau: problem solved
  - Finding where the maximum is located, being sure it is not a noise
  - Then counting the number of points around it with RGB values close “enough” (calibration)
  - If that count is more than 8 data are considered to have a plateau

Work with blocks – Algorithm with plateau

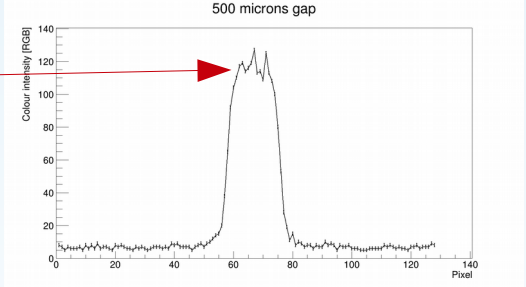
Calculate baseline and Std



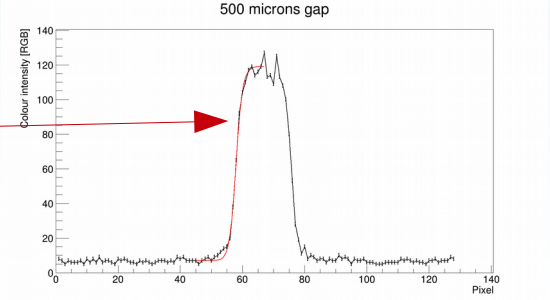
Find first three
consecutive points
out of a number of sigma
from baseline
as start to ROI (avoid spike noise)



Find first three
consecutive points
getting back inside the
double number of sigma interval

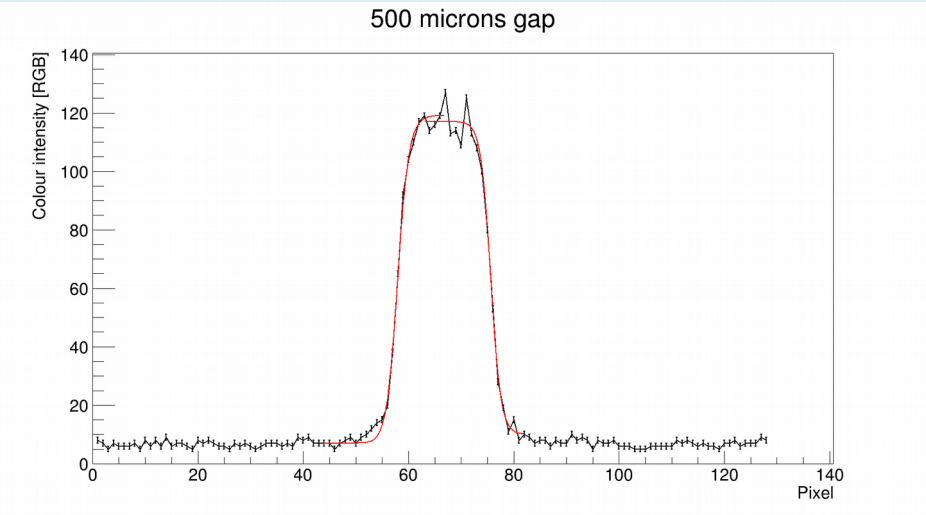


Perform the fit of the points
inside the area chosen

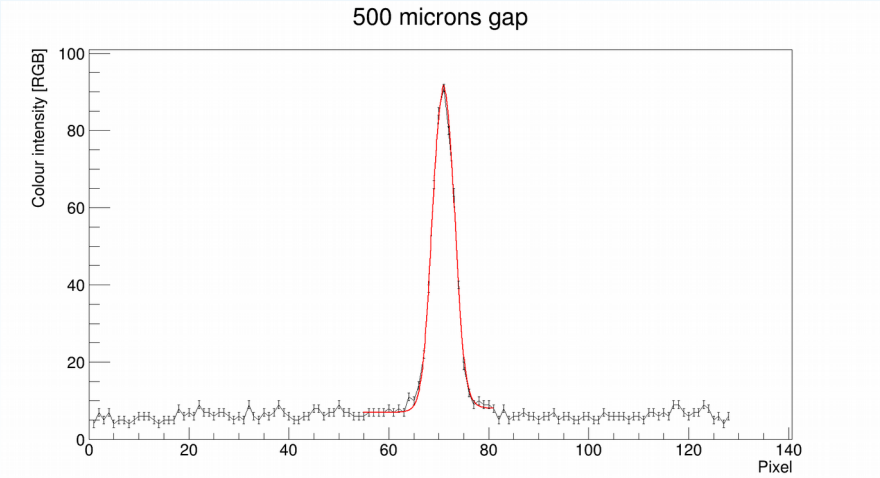


Work with blocks – Algorithm

- Repeat same kind of commands to perform fit of the other edge

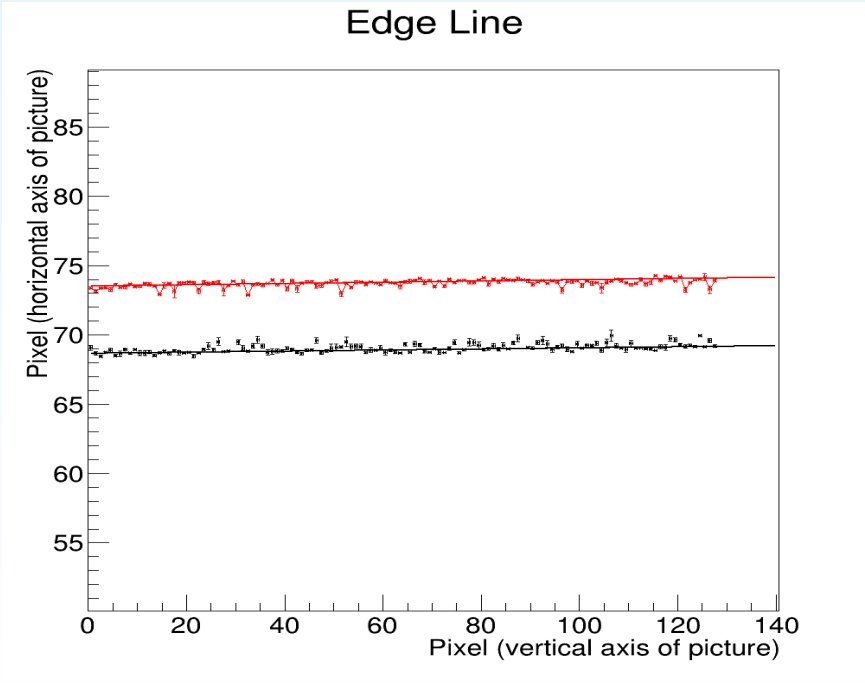


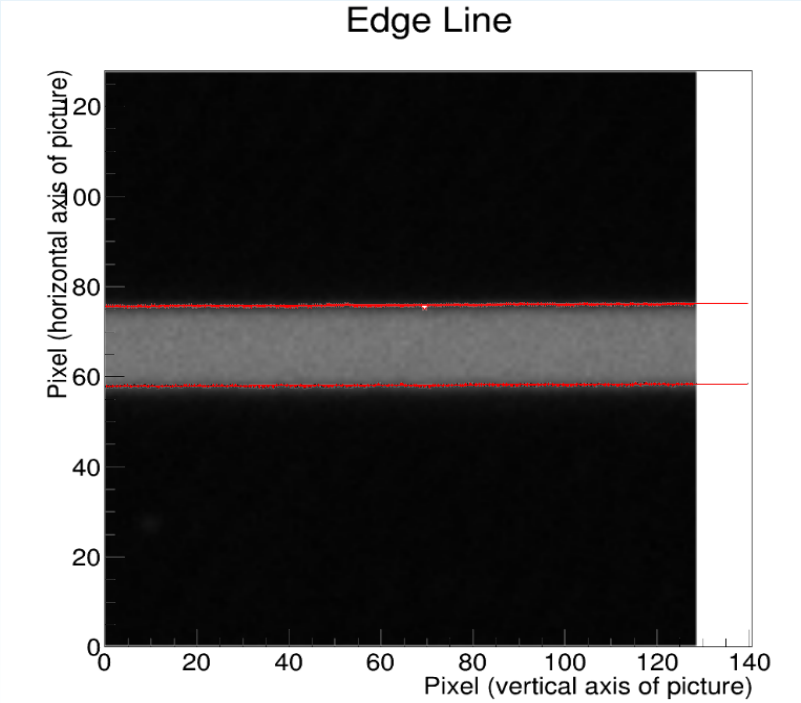
- In case of absence of plateau the max is where the first fit stops and the second starts



Work with blocks – Algorithm edgeline

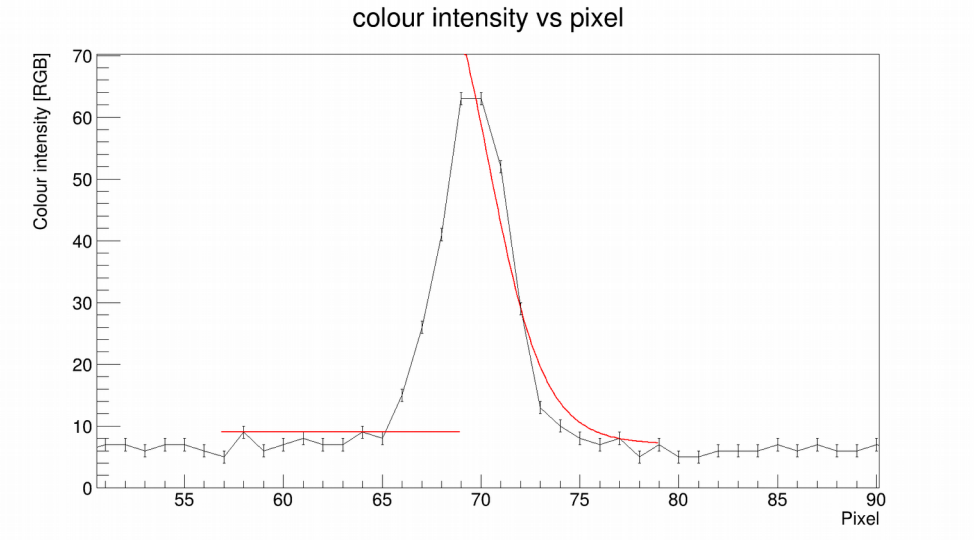
- The operation is repeated for each line of pixels in the direction orthogonal to the edgeline, or averaged bunch of lines.
- From this the parameter c is saved from each fit
- Having the position of the edge for each row, a linear fit can be used to find where the edgeline is located
- Once the two lines are obtained the minimum distance and relative angle can be calculated





Work with blocks – Algorithm failed fit

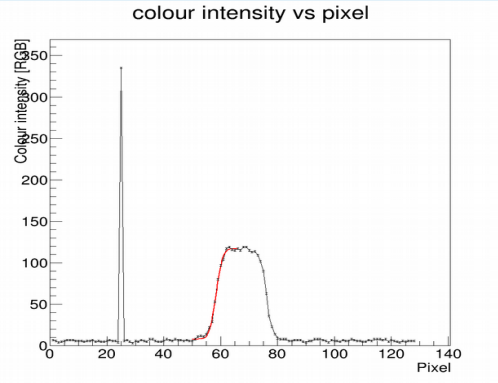
- Not always the algorithm to find the region of the fit works because of too much noise
- Sometimes points are too far from the S function
- Therefore some edge points are totally wrong

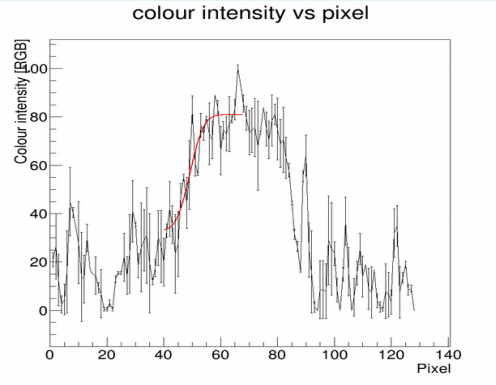


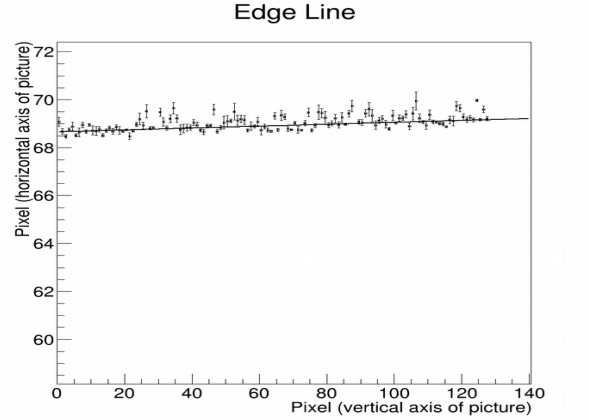
- Some controls are added to avoid these points when doing the line fit:
  - Chisquare/NDF
  - Fit status
- Number of failures counted to account for the quality of the edge line

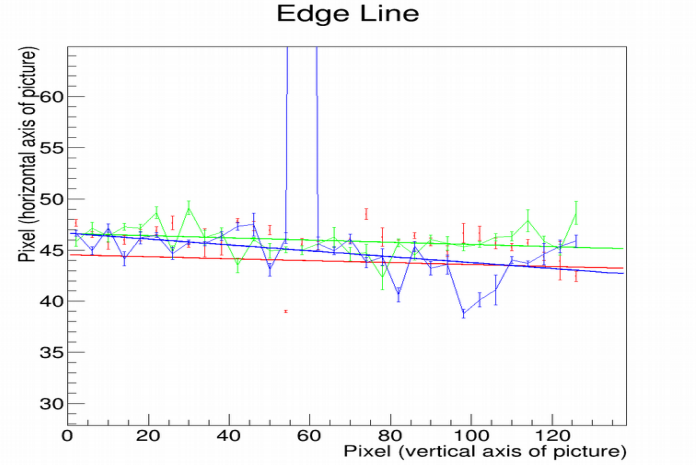
Work with blocks – Camera choice

- The colour camera has better resolution (pixelsize (1,67 μm) smaller than the greyscale one (3,45 μm))
- However the colour one is very noisy and it was hard to have results, so the gray was chosen
- With these lenses and camera 1 pixel corresponds to 25 μm



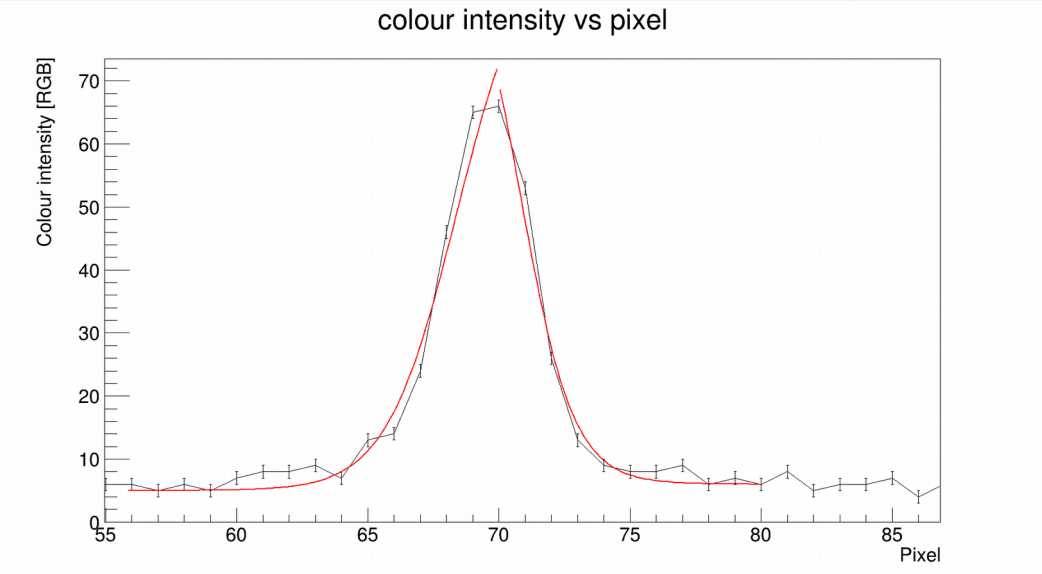






Work with blocks – Minimum gap width

- To see the minimum distance this algorithm can measure, pictures were taken with known relative increase of width of the gap
- The more the gap was shrunk the more the hypothetical functions data should follow overlapped, not showing all of their shape
- After studying the height of the plateau when the gap was wide enough to show it, the parameter b (amplitude) was forced to be close to that value
- The measurements were reliable down to a 50 μm gap



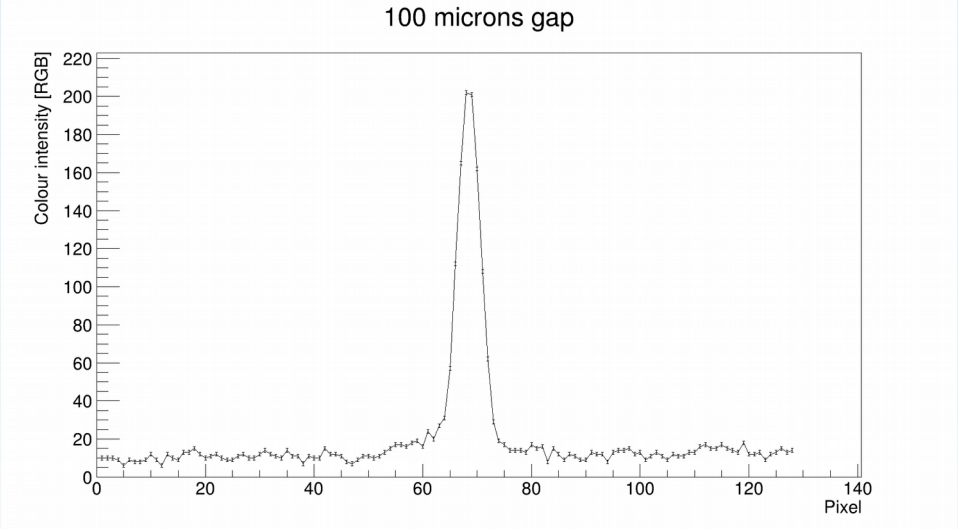
Example at 50 microns gap

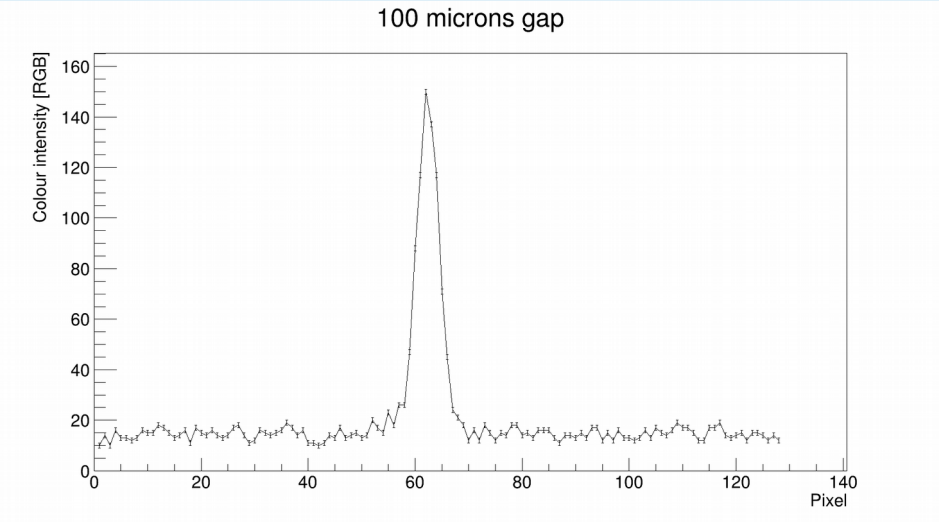
Work with blocks – Repetition of images

- To be sure that taking different picture in the same setup did not change result we tested it
- Trembling, little oscillations and camera noise are negligible to human eye but were detected by the camera, so each time the distance measured was different
- It would be better to average the results from a sequence of images
- The uncertainties on the mean value of the gap width is less than 1 μm
- The standard deviation of the distribution may give a better way to evaluate the region in which the raft should actually be. That is between 3 and 8 μm
- About the relative angle the uncertainties are close to 0,03°

Work with blocks – Light Effect

- To see the effects of illumination two pictures were taken with a light shining from behind inclined in two different ways



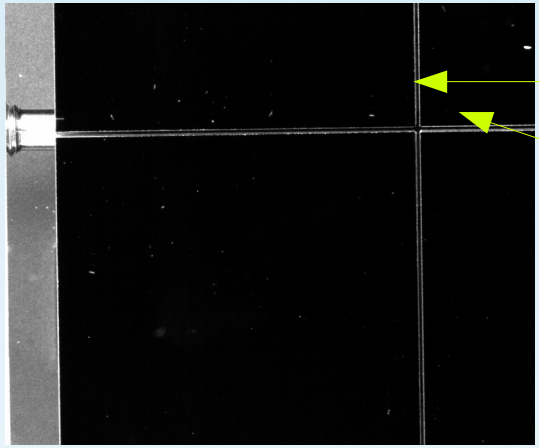


- The measured gap width did not change, but the position did by 150 μm

- From this it seems fundamental to control how light hits the sensor

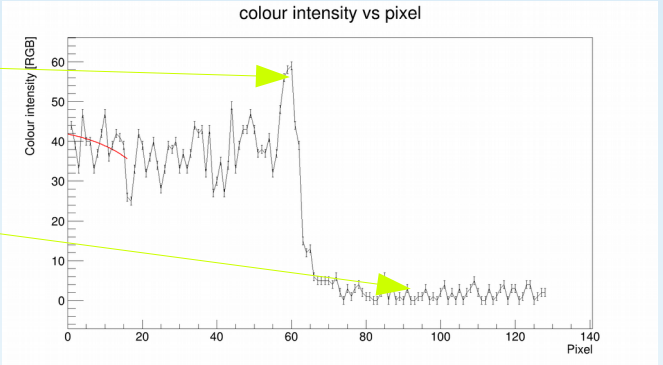
Work golden CCDs – Algorithm

- Presence of a golden, very reflective layer



Gold ring

CCD



- Images become noisy
- Easier to exploit the high value of RGB provided by the gold to locate the region of interest instead of the difference from the baseline
- High noise in bright region and thin area in which to define the point of the edge bring to the need of a blur
- Applied a gaussian blur to the image (openCV library)

Work golden CCDs – Algorithm

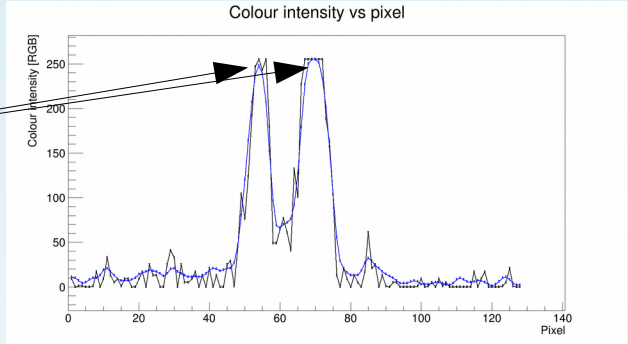
Reads directly from the image already cropped and applies a gaussian filter

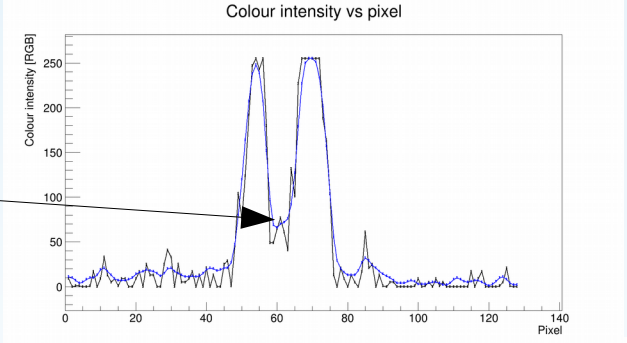
Finds the two maximum values separated by some pixels

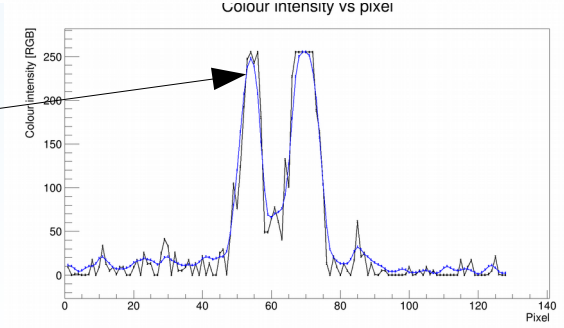
Finds the minimum in the area delimited by the two maxima

From the minimum it searches going backward where the derivate is increasing Than 30%, To find where to start the fit

These should be the two gold peaks in RGB graph.
The separation is imposed to prevent the second maximum to be part of edge curve of the maximum already found







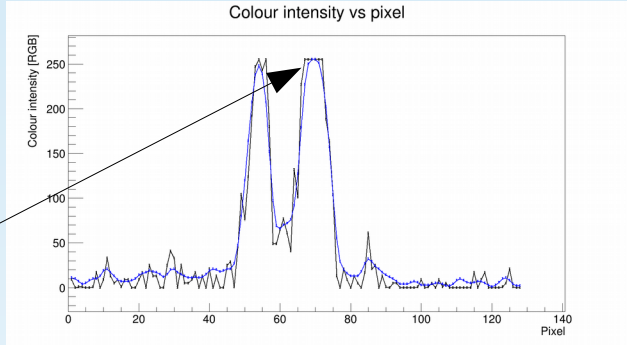
Work golden CCDs – Algorithm

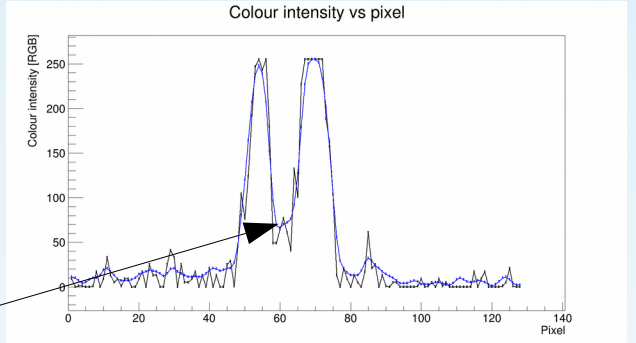
The start point can be placed
one point back
if it is close in RGB
value to the current start

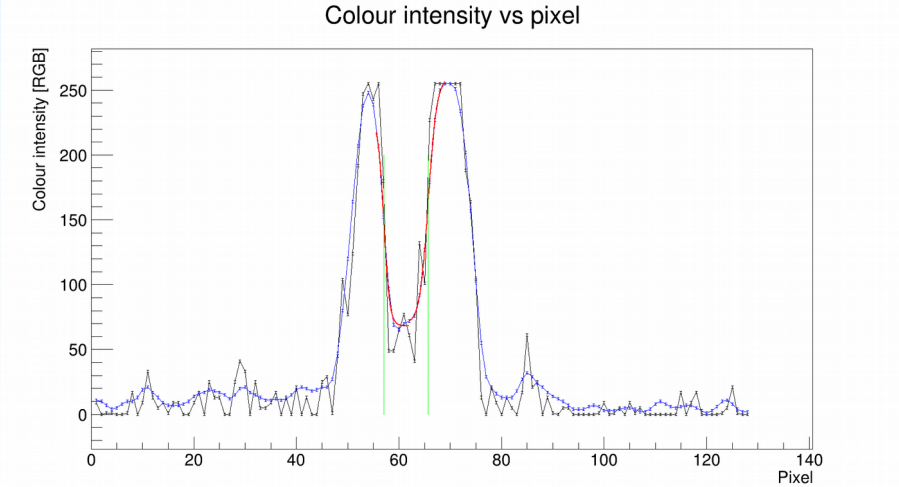
Does the same thing
searching forward
to find where to stop the second fit

Places the end of the first fit
and the beginning of the second
in the middle of the
area now defined

Fits, using the same
function as always,
the two regions







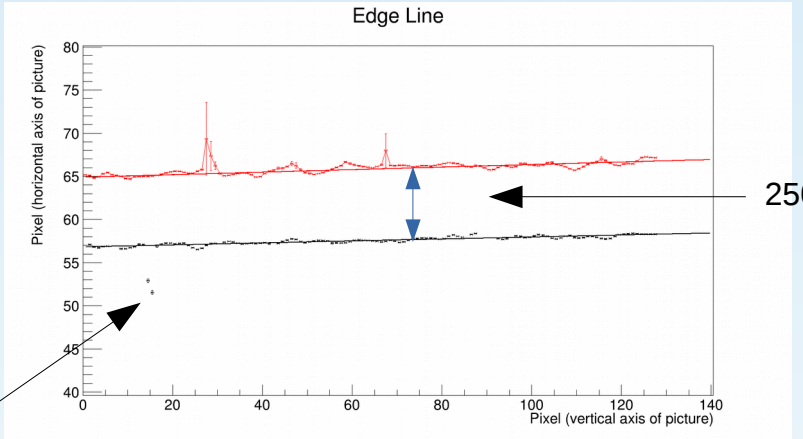
Work golden CCDs – Algorithm

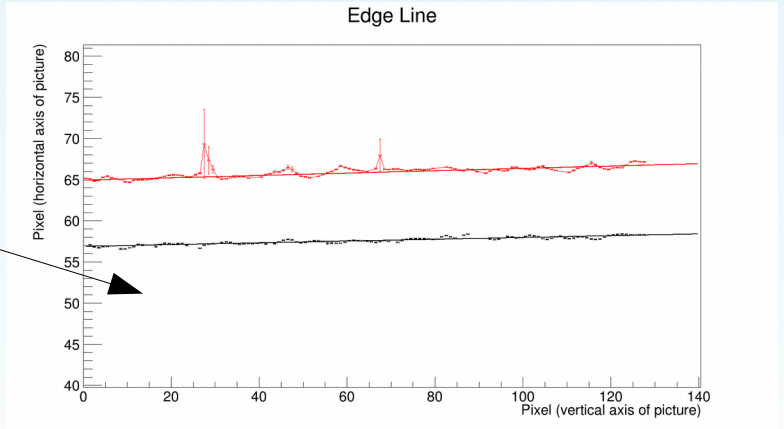
If a fit is too bad (chisquare)
or badly terminated,
shrinks the corrispondent region
and repeats(4 tries at maximum)

When all the rows are analysed
it can find the edgeline
with a linear fit

After first linear fit edge point
20 sigmas farther from the line
are erased and linear fit is repeated

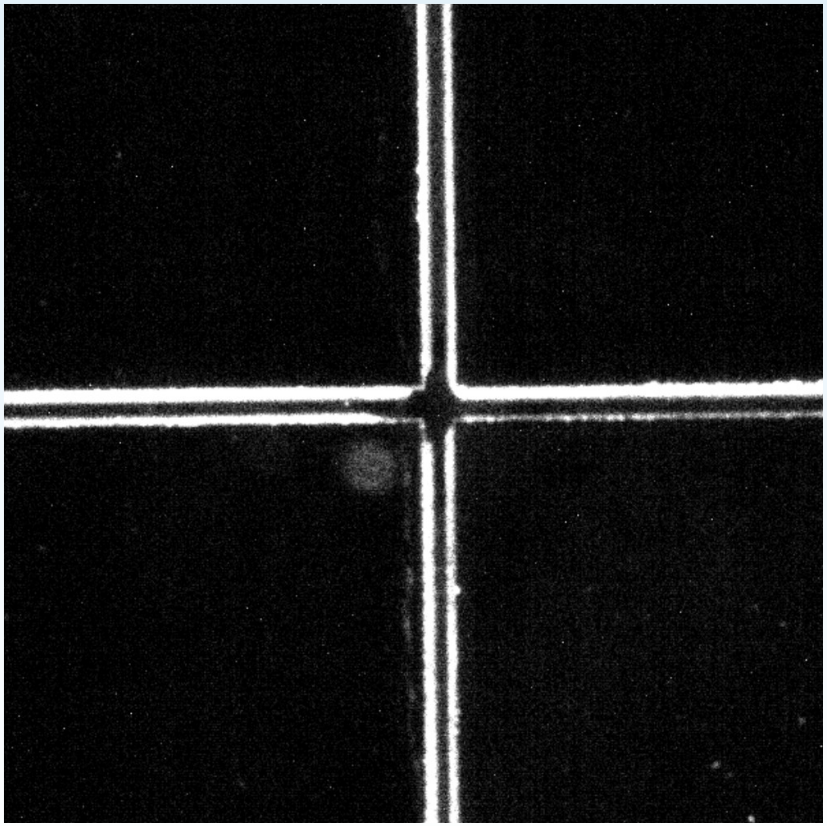
Repeats for all the images
of the same condition
and averages the results,
writing on a txt file

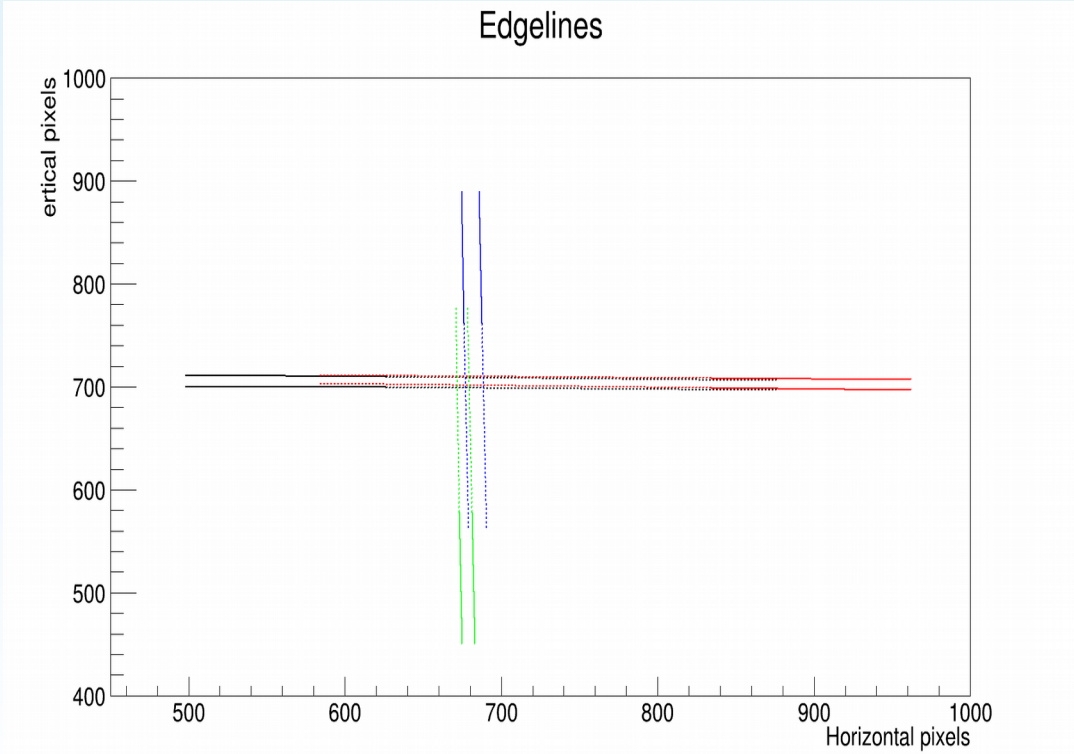




Work golden CCDs – Results

- This algorithm works either if the edgelines are vertical or if they are horizontal
- Averaging the results from the repetition of images of the same setup, angle and distance are found with the same uncertainties
- When looking at a corner





Work golden CCDs – Results

- As shown the extention of the lines, gotten directly from the points, do not match one another
- Zooming the picture, this seems to be correct

Major concerns:

- How light is shone on the detector affects how edges move on the picture
  - Suggested a light as homogeneous as possible
- Pictures are 128x128. If it is increased the dimension of the pictures more data points for defining the edges can be used

Conclusions

- Because of the need to place the rafts in the right position and avoid crashes during the installation and since the measurements required are difficult, an optical system is being built
- In order to provide the system with precise measurements, I developed a software to analyse images that gives information about position of the rafts
- Given the constraints needed to be considered useful (125 µm as minimum distance able to distinguish with 25 µm of uncertainties), the software seems to be on the right path to pass these tests
- This study reveals how much important is the way the sensors are illuminated
- Thanks to all the team that helped me with my work, especially, Scott Newbry and Kevin Reil

Thank you for the attention

Backup

Programs

- Some of the detectors will have black appearance because of the antireflecting coating looking similar to the blocks

- The way to find the edges should be quite similar

- Program used to deal with blocks:
  - Distanze.C
  - Testfunzionamento_algoritmo.C
  - Distanze.py
  - Testfunzionamento_algoritmo.py

    - Read image to be implemented in C macros
    - Fit filter, horizontal edges adaptation to be implemented in both

- Program used to deal with blocks:
  - Disttest.py
  - Testblur.py