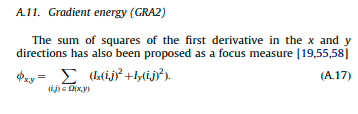
**Choosing a focusing regime for new PRISMS system**

Previous autofocusing algorithm

* Method: Passive contrast measuring
* Algorithum: EoG (Energy of gradient) (Partially chosen as fewer parameters)



(Pertuz et al., 2011)

* Gaussian Fit: 3 point fit
* Integration time: 60 ms
* ROI: 680x512 (Taken at the centre)

Old PRISMS Regime

1. Move the focuser as close as possible to being focused, then move back ~500 steps(This is to avoid false peaks being found).

2. Take image, and then take the blur metric. Depending on the out coming blur metric move the focuser forward/backwards and run the blur metric again. Repeat this process till either side of a peak is found.

3. When a peak in the blur metric is found, go beyond this point and see if the blur increases.

4. if so, fit a gaussian to these points to get the focus peak.

What regimes were tested

* Algorithms: EoG, Sum Modified Laplacian (SML), Tenegrad (Ten), Energy of Laplacian (EoL), Shaoe form focus (SF), Var
* Curve fits tested: 5-point Gaussian, 3-point Gaussian
* threshold between the optimum exposure time and a lower integration time
* ROI: medium size (680x512)
* Integration times between 30 ms and 80 ms

Shortcomings

* Did not account for blurring in odd corners caused by the focuser tilting the camera lens

**New autofocusing algorithm requirements**

Metrics to beat/match compared to old regime

* Speed/ time to completion
* Accuracy of regime

Limitations

* Speed of the Moonlite focuser
* sCMOS structure, so can’t use phase focusing
* Must be passive, don’t want to use laser ranger as additional equipment

Advantages

* Uses Crawford focuser so should not have tilting of the camera lens which led to blurring in odd corners.

Variables to chose

1. Contrast algorithm
2. Curve fit to the metric

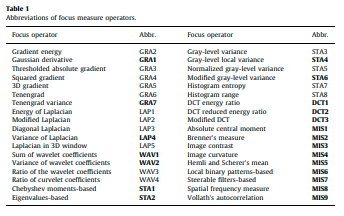
Additional Variables to consider

1. Noise filtering

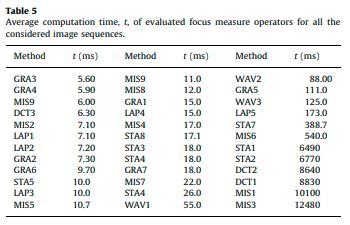
**1. Contrast algorithm**

There dozens of contrast algorithms to measure the blurriness of an image. Looking only at the algorithms that have computation time around the same as EoG, or greater if the accuracy is higher will be considered. Pertuz et al. 2011 hosts a broad consideration of algorithms so will be heavily depended upon. All of the images below are taken from this paper.

Algorithms Options



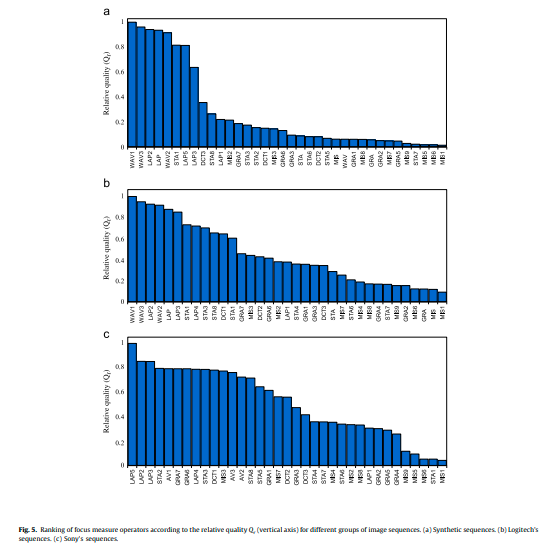
Computation time



Based on the computational time it would be safe to say only look at the functions that take less than 10s which could run as the focuser is moving to the next image. This leaves the below table of algorithums, with the green boxes faster than the current method, the yellow box being the current method and the orange boxes being slower than the current method.

|  |  |
| --- | --- |
| Method | Time (ms) |
| GRA3 Threshold Absolute gradient | 5.60 |
| GRA4 Squared gradient | 5.90 |
| MIS9 vollath’s autocorrection | 6.00 |
| DCT3 modified DCT | 6.30 |
| MIS2 Brenner’s measure | 7.10 |
| LAP1 Energy of Laplacian | 7.10 |
| LAP2 Modified Laplacian | 7.20 |
| GRA2 Gradient Energy | 7.30 |
| GRA6 Tenegrad | 9.70 |
| STA5 Normalized gray-level local variance | 10.0 |
| LAP3 diagonal laplacian | 10.0 |
| MIS5 hemli and scherer mean | 10.7 |

Accuracy

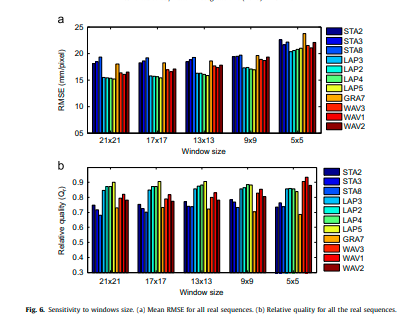


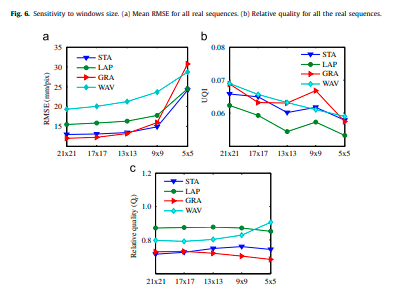
The Laplacian algorithms (LAP) consistently have a high accuracy metric compared to the other algorithms.

The current method will be tested as well as a few of the previous tested ones for completion. Also more common methods will be chosen as they are more likely to have modules accessible in python

|  |  |
| --- | --- |
| Method | Time (ms) |
| GRA3 Threshold Absolute gradient | 5.60 |
| GRA4 Squared gradient | 5.90 |
| DCT3 modified DCT | 6.30 |
| MIS2 Brenner’s measure | 7.10 |
| LAP1 Energy of Laplacian | 7.10 |
| LAP2 Modified Laplacian | 7.20 |
| GRA2 Gradient Energy | 7.30 |
| GRA6 Tenegrad | 9.70 |

ROI





There is a trade-off between the window size being large enough to focus the whole image and the spatial resolution. This needs to be tested for the new system, but for now we can use the median of the old system (680x512).

Noise sensitivity

For algorithms like the Laplacian which is good with low noise suffers when noises is introduced. Gradient based algorithms are much better at handling noise. A good work around can be to use a noise filter before passing the contrast algorithm, but this can lose a lot of good information resulting in a bad focus regardless.

PRISMS also has noise below its signal mostly so this should not be a big issue, but a pass could be done for hot pixels for example.

**Existing Python autofocusing modules**

https://doc.qt.io/qtforpython-5/PySide2/QtMultimedia/QCameraFocus.html

<https://image-filtering-suite.readthedocs.io/en/latest/focus_measure.html>

<https://pyimagesearch.com/2015/09/07/blur-detection-with-opencv/>

**Next steps for developing algorithm**

1. Write a focusing algorithm where the contrast metric can be slotted in and changed

1. Using the existing PRISMS algorithm come up with a robust routine to change the focus and find the focus etc.
2. For experimentation write this routine without any connected buttons.

2. Chose a contrast algorithum

1. Take images using PRISMS set-up to various steps for a set distance with known focus
2. Apply the focus metric to these images and see which when fitted with a Gaussian curve comes closest to the actual distance
3. Try for a few set distances in the PRISMS imaging range.
4. record the time to calculate for each one.

3. implement the algorithumn into the routine