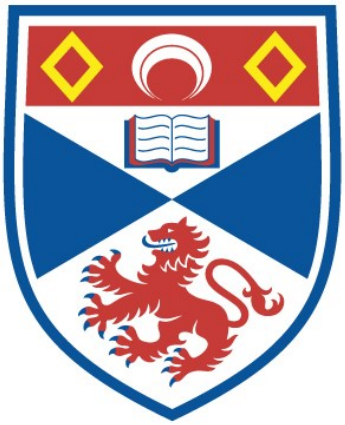
PERFORMANT ASTRONOMICAL IMAGE PROCESSING WITH PYTHON

James Hitchcock

A Thesis Submitted for the Degree of PhD
at the
University of St Andrews



2023

Performant Astronomical Image Processing with Python

James Hitchcock

University of
St Andrews

This thesis is submitted in partial fulfilment for the degree of

Doctor of Philosophy (PhD)

at the University of St Andrews

January 2023

Declaration

Candidate's declaration

I, James Hitchcock, do hereby certify that this thesis, submitted for the degree of PhD, which is approximately 30,000 words in length, has been written by me, and that it is the record of work carried out by me, or principally by myself in collaboration with others as acknowledged, and that it has not been submitted in any previous application for any degree. I confirm that any appendices included in my thesis contain only material permitted by the 'Assessment of Postgraduate Research Students' policy.

I was admitted as a research student at the University of St Andrews in January 2019.

I received funding from an organisation or institution and have acknowledged the funder(s) in the full text of my thesis.

Date 16/5/2023 Signature of candidate

Supervisor's declaration

I hereby certify that the candidate has fulfilled the conditions of the Resolution and Regulations appropriate for the degree of PhD in the University of St Andrews and that the candidate is qualified to submit this thesis in application for that degree. I confirm that any appendices included in the thesis contain only material permitted by the 'Assessment of Postgraduate Research Students' policy.

Date 16/5/2023 Signature of supervisor

Permission for publication

In submitting this thesis to the University of St Andrews we understand that we are giving permission for it to be made available for use in accordance with the regulations of the University Library for the time being in force, subject to any copyright vested in the work not being affected thereby. We also understand, unless exempt by an award of an embargo as requested below, that the title and the abstract will be published, and that a copy of the work may be made and supplied to any bona fide library or research worker, that this thesis will be electronically accessible for personal or research use and that the library has the right to migrate this thesis into new electronic forms as required to ensure continued access to the thesis.

I, James Hitchcock, confirm that my thesis does not contain any third-party material that requires copyright clearance.

The following is an agreed request by candidate and supervisor regarding the publication of this thesis:

Printed copy

No embargo on print copy.

Electronic copy

No embargo on electronic copy.

Date 16/5/2023 Signature of candidate

Date 16/5/2023 Signature of supervisor

Underpinning Research Data or Digital Outputs

Candidate's declaration

I, James Hitchcock, understand that by declaring that I have original research data or digital outputs, I should make every effort in meeting the University's and research funders' requirements on the deposit and sharing of research data or research digital outputs.

Date 16/5/2023 Signature of candidate

Permission for publication of underpinning research data or digital outputs

We understand that for any original research data or digital outputs which are deposited, we are giving permission for them to be made available for use in accordance with the requirements of the University and research funders, for the time being in force.

We also understand that the title and the description will be published, and that the underpinning research data or digital outputs will be electronically accessible for use in accordance with the license specified at the point of deposit, unless exempt by award of an embargo as requested below.

The following is an agreed request by candidate and supervisor regarding the publication of underpinning research data or digital outputs: No embargo on underpinning research data or digital outputs.

Date 16/5/2023 Signature of candidate

Date 16/5/2023 Signature of supervisor

Abstract

Image processing is fundamental to observational astronomy workflows. Astronomers acquire imaging data, and process the imagery to extract useful information. This thesis introduces two new image processing algorithms. The first, *PyTorchDIA*[1] is a GPU-accelerated approach to Difference Image Analysis (DIA). The approach is fast, without sacrificing modelling flexibility. It makes use of the Pythonic, PyTorch machine learning framework to accelerate convolution computations on the GPU, and compute gradients of user-specified objective functions with automatic differentiation methods to fit DIA models quickly and accurately. The second algorithm, *The Thresher*[2], was designed as a new tool to extracting information from Lucky Imaging (LI) data sets. We adopt a modelling approach which optimises a justifiable, physically motivated likelihood function to return the best estimate of the observed astronomical scene. It does this using all available data, and the more data the model is fit to, the better the signal-to-noise and resolution of the scene estimate. This fundamentally differs from conventional shift-and-add procedures, which typically reject the vast majority of the acquired LI data, as in these approaches, the signal-to-noise of the final coadd is inversely related to its resolution. With an eye to accessibility, integration into workflows and open science, the code for these two algorithms has been open sourced. Lastly, we show how Python image processing applications can be used to realise time-critical, demanding computational challenges in a chapter outlining the results of a novel pilot study to detect the occultations of background stars by small, outer solar system objects with high frame-rate sCMOS cameras.

[1] https://github.com/jah1994/PyTorchDIA
[2] https://github.com/jah1994/TheThresher

Acknowledgements

Little research is undertaken in isolation, and the work done during my PhD is no exception. Thank you to my supervisor, Martin Dominik, who gave me the freedom to pursue what I found interesting and engaging. Thanks to Aleks Scholz and Christiane Helling for their essential support in guiding me through the preparation of my first paper to be published during my PhD. Although that work doesn't appear in this thesis, the experience and confidence gained stood me in good stead for the following years. And thank you to Keith Horne, who's sage advice and comments have never been misplaced.

A very special thanks to Markus Hundertmark, who has been a consistent source of good ideas and fruitful discussions. And thank you to Etienne Bachelet, Rachel Street and Yiannis Tsapras for supporting the development of my research, and helping it reach a wider audience. And thank you to all members of the MiNDSTEp community, and Jesper Skottfelt in particular.

I couldn't have asked for a better reviewer, and then colleague, in Daniel Bramich. Thank you to Dan Foreman-Mackey, who's insights helped to spark so much of this work. And thank you to David Hogg for his support both directly via email and indirectly by his excellent research blog!

Very many thanks to Richard Gomer for the opportunities he's opened up for me, his infectious enthusiasm and much welcomed company. And thank you to the staff at McDonald Observatory, and especially John Kuehne for his invaluable assistance.

General acknowledgements

Funding

This work was supported by the Science and Technology Facilities Council of the United Kingdom.

Research Data/Digital Outputs access statement

Research data underpinning this thesis are available by request.

Digital outputs underpinning this thesis are available at [DOI]

- *PyTorchDIA* code repository `https://github.com/jah1994/PyTorchDIA`.
- *PyTorchDIA* paper `https://doi.org/10.1093/mnras/stab1114`
- *The Thresher* code repository `https://github.com/jah1994/TheThresher`
- *The Thresher* paper `https://doi.org/10.1093/mnras/stac427`

Contents

Declaration i

Abstract v

Acknowledgements vii

1 Introduction 1

2 *PyTorchDIA*: A flexible, GPU-accelerated approach to Difference Image Analysis 7

2.1 Declaration . . . 7

2.2 Introduction . . . 7

2.3 Problem Formulation . . . 10

2.3.1 Difference Image Analysis . . . 10

2.3.2 Astronomical Image Processing with PyTorch . . . 12

2.3.3 Difference Image Analysis as an Optimisation . . . 13

2.3.4 Robust loss function . . . 15

2.3.5 Uncertainty estimation - Observed Fisher Information . . . 17

2.3.6 Extension to a Spatially Varying Background . . . 18

2.3.7 Regularising the kernel pixels . . . 19

2.3.8 General purpose computing on GPUs . . . 20

2.3.9 Optimisation as an engineering problem . . . 21

2.3.10 The Algorithm . . . 22

2.4 Simulated Image tests . . . 23

2.4.1 Generating Artifical Images . . . 23

2.4.2 Performance Metrics . . . 25

2.4.3 Simulated Image Test results . . . 27

2.4.4 No 'algorithmic' bias . . . 31

2.5 Real Image Tests . . . 34

2.5.1 Data and reductions . . . 34
2.5.2 Model performance metrics for real data . . . 36
2.5.3 Real Image Test results . . . 37
2.6 Speed tests . . . 41
2.6.1 Real EMCCD images . . . 44
2.6.2 Synthetic CCD images . . . 48
2.6.3 cuDNN: Accelerating convolution computations on NVIDIA GPUs . . . 49
2.7 Conclusions . . . 50

3 *The Thresher* : Lucky Imaging without the Waste 53
3.1 Declaration . . . 53
3.2 Introduction . . . 53
3.3 Problem Formulation . . . 55
3.3.1 Online Multi-frame Blind Deconvolution . . . 55
3.3.2 A Poisson-Gamma-Normal Noise Model for EMCCD Data . . . 60
3.3.3 Noise Model Validation and Calibration . . . 61
3.3.4 Constraints and Regularisation . . . 65
3.3.5 Algorithm and Implementation details . . . 67
3.4 Simulated Image Tests . . . 69
3.4.1 Tests on Noiseless Images . . . 71
3.4.2 Tests on Noisy Images . . . 73
3.5 Real Image Tests . . . 75
3.6 Current limitations of *The Thresher* and the scope for future work . . . 76
3.6.1 Flux Non-linearity . . . 76
3.6.2 Computational expense . . . 78
3.7 Conclusions . . . 79

4 A data system for the serendipitous detection of small outer solar system objects 81
4.1 Declaration . . . 81
4.2 Introduction . . . 81
4.3 Occultation Light Curves . . . 84
4.4 The Data System . . . 87
4.4.1 Real Time Data Processing . . . 87

4.5 Observations at McDonald Observatory 94
4.5.1 Correcting systematic noise 97
4.5.2 Occultation Event Detection 105
4.6 Results and Discussion 106
4.6.1 No detection of real occultations 109
4.6.2 Computing our Detection Efficiency with Simulated Events 110
4.7 Conclusions 117

5 Summary and Future Work **121**

Bibliography **126**

List of Figures

2.1 A comparison of squared error loss against Huber loss for different values of c. . 16

2.2 Example images and fit quality metrics from the simulated image experiments. (*Top row*) An example reference (left) and target (right) image pair generated in the simulation tests. (*Middle row*) The subsequent difference images and fit metrics recovered by the B08 approach (left) and our PyTorch implementation (right). (*Bottom row*) The corresponding convolution kernels and fit parameters for the B08 (left) and PyTorch (right) solutions. The median pixel value is subtracted from the reference image before fitting the kernel and differential background term, and so B_0 will not be equal to zero. The B08 and PyTorchDIA solutions are very similar. 28

2.3 Fit quality and photometric accuracy metrics from the 71569 simulated image tests. Results for the B08 algorithm are in blue (left column), and the PyTorch-DIA results are in red (right column). The signal-to-noise regime of the target image is shown on the x-axis, increasing from left to right. Metrics in each SNR regime are offset from each other for clarity. We use circular markers to denote the sampling regime of the reference image, and crosses to indicate the sampling regime of the kernel. A big circle or cross corresponds to an over-sampled reference image or kernel respectively, and a small circle or cross corresponds to an under-sampled reference image or kernel; there are therefore 4 possible combinations of marker for each SNR regime. The green dashed lines for each sub-plot pair represent the correct, 'ideal' value for each metric. 32

2.4 Example reference-target image pair from the real image performance tests, in the same style as Figure 2.2. Note the unusual PSFs in the reference and target images, and the correspondingly irregular kernels. The target star with which the photometric accuracy was assessed is at the centre of the reference-target image pair. 38

2.5 Fit quality and photometric accuracy metrics from the 6989 real image tests. Results for the B08 algorithm are in blue (left column), and the PyTorchDIA results are in red (right column). The signal-to-noise regime of the target image is shown on the x-axis, increasing from left to right. We use circular markers to denote the sampling regime of the reference image, and crosses to indicate the sampling regime of the kernel. As the reference image is oversampled, all circular markers are large. A large cross corresponds to an oversampled kernel, and a small cross corresponds to an under-sampled kernel; there are therefore 2 possible combinations of marker for each SNR regime.No reference-target image pairs used to compute these metrics fell into the lowest SNR regime, and so that is left blank. The green dashed lines for each sub-plot pair represent the correct, 'ideal' value for each metric. 42

2.6 The time taken to solve for square kernels of different sizes against image single axis length (as a proxy for image size) for our PyTorchDIA implementation on the GPU and the B08 approach for images in a typical DK154 microlensing data set. The PyTorchDIA kernel pixels were initialised with a (top plot) 'flat', box-car kernel and (bottom plot) a symmetric Gaussian with width estimated as $\phi_K = \sqrt{\phi_I^2 - \phi_R^2}$. 46

2.7 Pair plot showing the optimisation solution times for a 19×19 kernel – initialised as either a flat box-car or Gaussian – on a 482 × 482 large image, against the ϕ_I and SNR of the target image. The histograms of the distributions are shown on the diagonal. 47

2.8 The time taken to solve for (square) kernels of different sizes against image single axis length (as a proxy for image size) for our PyTorchDIA implementation on the GPU and the B08 approach for a pair of square synthetic images, cropped to different sizes. 49

2.9 A comparison of the times taken to infer square kernels of different sizes with varying cuDNN settings for a given pair of 1000 × 1000 pixel synthetic images. Note the log-log scale. 51

3.1 Normalised histogram of pixel counts from 500, 256 × 256 pixel (i.e. $\sim 3.3 \times 10^7$ data points in total), dark DK154 EMCCD images, overlain with the fitted Poisson-Gamma-Normal (PGN) likelihood. The positive fat tail is a result of amplified spurious charges. The relative difference between the model and data – computed as (data - model) / model – for each bin is plotted in the lower panel. The MLE for the detector parameters are also shown (see Table 3.2 for units). . 64

3.2 The model image returned by our algorithm after (left to right) 0, 50, 250 and 500 SGD updates from the test on noiseless short exposure PSF images. 72

3.3 128×128 pixel cutouts of the (from left to right) theoretical instrument diffraction pattern, the shift-and-added sharpest 1% of images and the model returned by our algorithm after 12000 updates from the test on the 3000 noiseless short exposure PSF images. All cutouts are normalised relative to the highest intensity pixel, and are plotted on a logarithmic scale. The squared error loss used by our algorithm for these noiseless data does not require non-negativity of the model, and so we do not enforce this constraint. Consequently, some regions close to the centre went negative, and for the purposes of plotting on the log-scale we added a small offset. 72

3.4 Radial profiles of the PSF images in Figure 3.3. Each (normalised) intensity value is the mean of all pixels at the given distance from the peak. The pixel scale is ~ 0.013 arcseconds per pixel. 72

3.5 A comparison of the TLI sharpest 1% and sharpest 50% of images, and the image model returned by *The Thresher* after a single pass over a stack of 3000 simulated short exposures. For comparison with the model, the TLI coadds have had their respective median pixel values subtracted (as estimates of their sky levels) and non-negativity enforced. All images are on a log-scale. 74

3.6 Radial profiles of the bright, central star in Figure 3.5 in the sharpest 1% TLI coadd and the image model returned by *The Thresher*. The theoretical diffraction limited PSF is plotted for comparison. Each (normalised) intensity value is the mean of all pixels at the given distance from the peak. The pixel scale is ~ 0.05 arcseconds per pixel. 74

3.7 A comparison of the TLI sharpest 1% and 50% of images, and the image model returned by *The Thresher* after a single pass over a stack of 4800 real short exposures. For comparison with the model, the TLI coadds have had their respective sky background's subtracted and non-negativity enforced. All images are on a log-scale. 76

3.8 Radial profiles of a bright, fairly isolated, centrally located star in Figure 3.7 in the sharpest 1% TLI coadd and the image model returned by *The Thresher*. The theoretical diffraction limited PSF is plotted for comparison. Each (normalised) intensity value is the mean of all pixels at the given distance from the peak. The pixel scale is ~ 0.09 arcseconds per pixel. 77

3.9 A plot of pixel values of the (kernel convolved) reconstructed image vs. the respective counts in a TLI 100% selection coadd from the tests in (left) Section 3.4.2 and (right) Section 3.5. The straight line – with gradient 1 and intercept 0 – represents photometric consistency between the model and data as a function of flux. The residuals in the bottom panel are expressed as a percentage of the model value. 79

4.1 Noise-free occultation light curves that would be observed with the Kinetix with no filter for circular objects of radii 0.5, 2.5 and 5 km at distances of 40 and 1000 AU, passing in front of an A0V background star, viewed at opposition. The impact parameter and background star angular size are set to $b = 0$ km and $\theta_* = 0.02$ mas respectively. Each row corresponds to the fiducial sampling frequencies of $f = 20$, 80 and 300 Hz for the TAOS II mission, and the Sensitivy and Speed modes of the Kinetix camera used in this work. 88

4.2 Noise-free occultation light curves that would be observed with the Kinetix with no filter for a 2.5 km radius circular object at distances of 40 and 1000 AU, passing in front of an A0V background star, viewed at opposition. From left to right, the impact parameter is set to $b = 0, 1$ and 4 km respectively. The background star angular size is set to $\theta_* = 0.02$ mas. Each row corresponds to the fiducial sampling frequencies of $f = 20$, 80 and 300 Hz for the TAOS II mission, and the Sensitivy and Speed modes of the Kinetix camera used in this work. 89

4.3 Noise-free occultation light curves that would be observed with the Kinetix with no filter for a circular object of radius 2.5 km at distances of 40 and 1000 AU, passing in front of an A0V background star, viewed at opposition. From left to right, the background star angular size is set to $\theta_* = 0.01, 0.02$ and 0.20 mas respectively. The impact parameter is set to $b = 0$ km. Each row corresponds to the fiducial sampling frequencies of $f = 20$, 80 and 300 Hz for the TAOS II mission, and the Sensitivy and Speed modes of the Kinetix camera used in this work. 90

4.4 Noise-free occultation light curves that would be observed with the Kinetix with no filter for circular objects of radii 0.05, 0.1 and 0.25 km at distances of 40 and 1000 AU, passing in front of an A0V background star, viewed at opposition. The impact parameter and background star angular size are set to $b = 0$ km and $\theta_* = 0.02$ mas respectively. Each row corresponds to the fiducial sampling frequencies of $f = 20$, 80 and 300 Hz for the TAOS II mission, and the Sensitivy and Speed modes of the Kinetix camera used in this work. 91

4.5 Noise-free occultation light curves that would be observed with the Kinetix with no filter for a circular object of radius 5 km at distances of 5000 AU and 10,000 AU, passing in front of an A0V background star, viewed at opposition. From left to right, the background star angular size is set to $\theta_* = 0.004, 0.02$ and 0.20 mas respectively. The impact parameter is set to $b = 0$ km. Each row corresponds to the fiducial sampling frequencies of $f = 20$, 80 and 300 Hz for the TAOS II mission, and the Sensitivy and Speed modes of the Kinetix camera used in this work. 92

4.6 Example of a reference image annotated with the (black) source aperture stamps and (red) regions in which to compute sky background levels. The image is of M2, taken in high-speed mode with the Kinetix camera on the McDonald 2.1 meter in September 2021. 95

4.7 The real time plots consist of up to four panels; during the continuous live acquisition, this is updated once every $\sim$ 5 seconds. The first two panels are bright, well-separated aperture source stamps from the most recently acquired image; axes indicate pixels. We also include the option to time-average over many aperture source stamps if the targets are faint. Data are from sources 0 and 4 from Figure 4.6. The third panel shows the normalized light curves from these two stamps for the last 100 images. The final panel shows the light curve root-mean-square deviation (RMSD) as a function of source flux for all detected sources over the last 100 images. Collectively, this information helps us with guiding the telescope and assessing the choice of aperture size. 96

4.8 The raw aperture flux measurements in ADU of a few sources of different brightnesses in an example Sensitivity mode data set. Note the log scale of the y-axis. 100

4.9 The data in Figure 4.8 where each star light curve has been normalised w.r.t its median flux measurement. 101

4.10 A comparison of the raw and corrected relative brightnesses of some star light curves in a data set. The lighter coloured raw light curves are vertically offset for clarity. 102

4.11 Stellar r-band magnitude vs SNR for stars in M11 in typical Sensitivity (circles) and Speed (crosses) mode datasets from our observations on the 2.1 m at McDonald Observatory. We plot the predicted SNR of stars across this brightness regime for our observations on the 2.1 m, the upcoming TAOS II mission and our planned observations on the 11.25 m Maunakea Spectroscopic Explorer (MSE). The noise model predictions include appropriate estimates of the readout noise, dark current, photon shot noise and scintillation. Scintillation noise clearly limits the achievable SNR at the bright end for these short exposure time series, and horizontal lines are marked on the plot showing where this limit should exist for the Sensitivity (undashed) and Speed (dashed) observations. 103

4.12 Occultation light curves for 3 small objects of $r \leq 1$ km at distances of 40 AU (blue) or 1000 AU (orange) passing in front of a r-band magnitude 14 star of spectral type A0V, with an angular size of 0.02 mas. The top row shows how these events would appear in TAOS II data, and the middle and bottom rows show how the same occultation signals would appear in Kinetix observations on the 11.25 m Maunakea Spectroscopic Explorer (MSE). Gaussian white noise was added to the simulated events based on the predicted SNR estimates in Figure 4.11 for each situation. 104

4.13 [Top] Simulated occultation event for a 2500 m radius object at 1000 AU injected into the real time series of a star in an example Sensitivity mode data set. Other occultation parameters are $b = 0$ m, and $\theta_* = 0.02$ mas, and the star is assumed to be of spectral type A0V. The data has been corrected using the approach outlined in Section 4.5.1. [Bottom] The matched filter signal-to-noise ratio time series of the (left) true occultation signal and (right) best-matching U-shaped template. 107

4.14 [Top] Simulated occultation event for a 1250 m radius object at 1000 AU injected into the real time series of a star in an example Sensitivity mode data set. Other occultation parameters are $b = 0$ m, and $\theta_* = 0.02$ mas, and the star is assumed to be of spectral type A0V. The data has been corrected using the approach outlined in Section 4.5.1. [Bottom] The matched filter signal-to-noise ratio time series of the (left) true occultation signal and (right) best-matching U-shaped template. 107

4.15 [Top] Simulated occultation event for a 500 m radius object at 40 AU injected into the real time series of a star in an example Sensitivity mode data set. Other occultation parameters are $b = 0$ m, and $\theta_* = 0.02$ mas, and the star is assumed to be of spectral type A0V. The data has been corrected using the approach outlined in Section 4.5.1. [Bottom] The matched filter signal-to-noise ratio time series of the (left) true occultation signal and (right) best-matching U-shaped template. 108

4.16 [Top] Simulated occultation event for a 250 m radius object at 40 AU injected into the real time series of a star in an example Sensitivity mode data set. Other occultation parameters are $b = 0$ m, and $\theta_* = 0.02$ mas, and the star is assumed to be of spectral type A0V. The data has been corrected using the approach outlined in Section 4.5.1. [Bottom] The matched filter signal-to-noise ratio time series of the (left) true occultation signal and (right) best-matching U-shaped template. 108

4.17 Simulated occultation events detected in 83 Hz data acquired in Sensitivty mode. Plots are annotated with the event parameters. 112

4.18 Simulated occultation events detected in 300 Hz data acquired in Speed mode. Plots are annotated with the event parameters. 113

4.19 The detection efficiency as a function of simulated occultation event parameters for the Sensitivity and Speed mode data sets. 114

4.20 The detection efficiency as a function of occultation event parameters for the different object sizes used in the simulations. 116

4.21 The (normalised) number of recovered and missed events for objects of different sizes as a function of data SNR. 118

List of Tables

2.1 The model accuracy, fit quality, and photometric performance metrics used to assess the DIA implementations in this work. 27

2.2 Fit quality and photometric accuracy metrics for the 71569 simulated image tests for both an implementation of the B08 algorithm and PyTorchDIA. We divide the results of these tests by the SNR regime of the target image and the sampling regimes of the reference image and convolution kernel. The number of simulations used for the computation of the metrics in each of the 12 possible SNR and sampling regime categories is shown in the bottom section. 33

2.3 Fit quality and photometric accuracy metrics for the 6989 real image tests for both an implementation of the B08 algorithm and PyTorchDIA, separated into the SNR and sampling regimes. The number of image stamps used to compute the metrics in each of the SNR and sampling regime categories is shown in the bottom section of the table. As outlying photometric residuals have been removed from some categories prior to computing the MPB and MPV metrics, we write the number of N_{Stamps} values used in their computation as a fraction of the total number of stamp samples. 43

3.1 A table of notation used in this paper. 56

3.2 EMCCD detector parameters. We adopt the same notation as Harpsøe et al. (2012), who differentiate between electrons generated before and after the EM amplification as e^-_{Photon} and e^-_{EM} respectively. 61

4.1 Table of occultation event parameters and their definitions 85

4.2 The open clusters targeted for these observations. In addition to hosting many bright, main sequence stars, these targets were chosen as they were fairly close to the line tangent to the Earth's orbit at the time of observing. 97

4.3 Observation inventory 16th - 22nd September. 3 and 12 ms exposures were recorded in the Kinetix's Speed and Sensitivity modes respectively. 98

4.4 99

1

Introduction

This thesis documents my work on a few astronomical image processing applications for ground-based observations. These applications have been implemented in the Python programming language. Python has consistently been ranked as one of the most popular languages, and in recent years has seen a marked rise in usage in astronomy. This trend doesn't seem to be changing any time soon, and in the last year alone, Python has had the largest increase in popularity amongst the 20 most popular languages in the TIOBE index rankings[1]. In this context, for the sake of both accessibility and integration into workflows in the wider astronomy community, we have released open source Python implementations of the algorithms developed here.

Unlike other popular high-level, general purpose languages such as C++ or JAVA, Python is dynamically typed, and so applications can be slow compared to their statically typed counterparts. However, this need not be the case for our applications, and we make grateful use of the rich suite of Python libraries to write performant image processing tools. These libraries

[1] https://www.tiobe.com/tiobe-index/

include NumPy for array programming (Harris et al., 2020), which when combined with the just-in-time compiler Numba (Lam et al., 2015), can straightforwardly transform Python code into fast machine code that can be executed in parallel across multiple CPU cores. We also make use of the PyTorch machine learning framework (Paszke et al., 2019) to run expensive, yet embarassingly parallel image operations on graphics processing units (GPUs), and use its automatic differentiation tools to reliably fit image models to data.

My use of the term "image processing" in the title of this thesis is deliberately vague. The work in Chapters 2 and 3 would probably be better described as "image modelling". Indeed, Chapter 3 in particular argues for the benefits of modelling approaches over the more typical, "operations-based" approaches to dealing with imaging data sets that astronomers are familiar with (e.g. procedures to align images, coadd them etc.). There is absolutely no image modelling in Chapter 4 however, and so in the pursuit of coming up with a title that ties this thesis together, I've settled with the usefully broad term "processing" to describe the act of applying algorithms to images.

With one admission out of the way, I'll move on to my next. The work in most astronomy theses neatly fall within some sub-discipline (Exoplanet science, Star Formation, Galaxies, Cosmology, Instrumentation etc.), and authors will use the introductory chapter to give some context as to how their work sits within the larger landscape. This thesis is mostly concerned with very specific image processing "Methods", which are not associated with any one particular astrophysics goal. Given this, I instead defer outlining the wider context of each of these methods to their respective chapters. I hope you'll agree with me that providing a general review of astronomical image processing approaches would only result in producing an increase in net global boredom, so instead, I use this introductory chapter to summarise my motivations for tackling these specific image processing topics, and what I believe to be the main contributions of each piece of work.

Chapter 2 describes a new approach to Difference Image Analysis (DIA), also known as Image Subtraction. DIA is a popular technique in time domain astronomy, and the aim is to compute the so-called "difference image" of a *reference* image and a *data* image of some astronomical scene. Unfortunately, we can't just subtract the reference from the data, as we must first correct for the inevitable differences in observing conditions between the times each of these images were acquired. For images on the ground in particular, these differences are

largely due to the variable atmosphere, and the two images will almost certainly have different point-spread-functions (PSFs), and different photometric throughputs. We therefore must fit a model to the data image, whose parameters encode the transformations which fit the reference image to it. A problem for many DIA techniques is that *fitting this model to the data can take a long time*. As astronomical data sets get bigger, we need processing methods that scale to this challenge. This is of particular importance for time-critical science cases in modern sky surveys using DIA, such as the Legacy Survey of Space and Time (LSST) (Ivezić et al., 2019). The faster images can be processed, the faster alerts for follow-up observations can be issued to the community. The work in Chapter 2 was motivated by wanting to address this concern without sacrificing model complexity. In grateful collaboration with the co-authors, I introduce a DIA approach which can fit models quickly, and is uniquely flexible with regards to the choice of *noise model*. Unlike any other DIA method we are aware of, our approach can make use of *robust* statistical techniques that allow it to straightforwardly accomodate outlying pixels in the data i.e. those affected by saturation, cosmic rays, instrumental defects etc. This does away with the need of delicate masking and/or (the albeit unreasonably effective) "sigma clipping" procedures which are commonly used in astronomy. I was kindly invited to give a short talk by the LSST Transients and Variable Stars collaboration in October 2021 on this work [2]. If you're short on time, or looking for a high level overview of the key ideas, I'd recommend watching the recording (link in footnote) in place of reading Chapter 2.

The work in Chapter 3 was initially motivated by a dissatisfaction with image coaddition methods. Stacking images together to increase the effective signal-to-noise ratio (SNR) of the observed scene is a very common and useful procedure. However, like in the DIA problem, for images taken through the atmosphere, due to differences in observing conditions between the times the images are acquired, each image will have a different PSF. Consequently, if you simply align and add (or average over) all your images, the effective *seeing* of this coadd will be limited by the worst seeing image you have. For this reason, it is usual to just select a small number of good quality images to contribute to the coadd. The Lucky Imaging (LI) technique is an extreme example of this (Law et al., 2006). At the sub-second timescales over which LI images are acquired, the image-to-image seeing variations in the complex PSFs can be large, as there's limited time-averaging over the PSF speckle patterns. LI is a brute force solution to this problem, which takes many 1000s of these sub-second exposures, and usually selects

[2] `https://www.youtube.com/watch?v=jAEFsrVrlSs&t=1006s`

just the sharpest 1% of images for the coadd, and discards the vast majority of the data. But, it cannot possibly be that when *treated correctly*, this ignored data is useless for improving our knowledge of the observed scene. LI is primarily concerned with taking high-resolution imagery from the ground, and the reason that rejecting data helps in LI is because *the SNR of the final coadd is inversely related to its resolution*. With *The Thresher*, I and co-authors attempted to address this shortcoming of coaddition, and LI shift-and-add procedures in particular. The main contribution is an online, multi-frame blind deconvolution algorithm that is robust to the very low SNR of short exposure LI data. Unlike shift-and-add processing procedures for LI data, our algorithm optimises a physically motivated, justifiable likelihood function for the observed scene. And unlike conventional coaddition procedures, as more and more data is shown to the algorithm, *both the SNR and resolution of the scene estimate improve*.

Chapter 4 is also concerned with the processing of high frame-rate imaging data, but with the specific goal of detecting the occultations of background stars by small objects in the outer solar system. Unlike the two previous chapters, this work has not been published (nor yet prepared for publication), and largely consists of the analysis of data taken in the autumn of 2022, so may be comparatively a little rough around the edges. Nonetheless, I think the science is interesting, and as it shares a common imaging processing thread with the rest of this work, I think it's worth putting it in here. Myself and my colleague, Prof. Richard Gomer (Texas A&M University), are interested in trying to detect and characterise objects in the solar system beyond the Kuiper belt, in the hypothetical Oort Cloud. I stress the word hypothetical, as although solar system formation models predict the existence of this vast reservoir of mass, no observations of its constituent objects have yet been made. As the objects are so small and distant, even the most powerful telescopes cannot detect their reflected light. Therefore, the best way to learn something about them is through serendipity surveys, where just by chance, such an object may pass in front of some distant star, causing a momentary, but possibly measurable dip in its brightness. This has some similarities to the transit method to detecting and characterising the properties of exoplanets, although unlike transit light curves, the light curves associated with occultations by relatively nearby objects can show features associated with Fresnel diffraction (Nihei et al., 2007). Additionally, and the feature most important to image processing work, is that these occultation events occur on sub-second timescales. One therefore needs a very fast camera to detect these events, and the better sampled the light curve is, the better the diffraction pattern can be resolved, and the better the properties of

the occulting object can be characterised. We have developed a data system that can process 3200×3200 pixel scientific Complementary Metal-Oxide-Semiconductor (sCMOS) images at frame rates as high as ~ 500 Hz in real time. Real time processing is essential, since at these frame rates just a second of observing equates to an unwieldy ~ 5 Gb of imaging data. For serendipity surveys, the storage requirements are completely impractical. The challenge is therefore to extract the useful information from each image on a millisecond timescale before throwing it away, *never to be seen again*. The work in this chapter describes the first steps in a long-term project that ultimately aims to mount these fast cameras in the otherwise unused areas around the edges of the focal planes of modern telescopes. The attractive advantage of this is that we can conduct a serendipity survey without taking up any dedicated telescope time; the main science observations proceed as normal, and our data system just sits in the background. We present an analysis of observations produced from trialling our data system on the 2.1m telescope at McDonald Observatory, Texas, acquired between the 16th - 22nd of September 2022.

2

PyTorchDIA: A flexible, GPU-accelerated approach to Difference Image Analysis

2.1 Declaration

This chapter includes material adapted from: Hitchcock, J.A., Hundertmark, M., Foreman-Mackey, D., Bachelet, E., Dominik, M., Street, R. and Tsapras, Y., 2021. PyTorchDIA: a flexible, GPU-accelerated numerical approach to Difference Image Analysis. Monthly Notices of the Royal Astronomical Society, 504(3), pp.3561-3579.

2.2 Introduction

Difference Image Analysis (DIA) describes several astronomical image processing algorithms with the shared goal of delivering precise photometric measurements of variable astronomical sources. Given two images of the same scene acquired at different times, in a DIA framework, one aims to subtract one image from the other and recover a *difference image* from which dif-

ferential fluxes can be directly measured. This makes DIA a particularly effective approach to measuring the photometric variability of objects in crowded stellar fields – such as microlensing campaigns e.g. (Wozniak, 2000; Bond et al., 2001) and studies of globular clusters e.g. (Bramich et al., 2011; Kains et al., 2012; Figuera Jaimes et al., 2013) – where the blending of light from neighbouring sources cannot otherwise be easily disentangled.

Even for two perfectly aligned images from the same instrument, in order to produce a clean subtraction, the DIA algorithm of choice must model the inevitable changes in 1) the Point spread function (PSF) due to variations in seeing, telescope focus or tracking errors, 2) the Photometric scaling from differences in atmospheric transparency, exposure time or instrument throughput and 3) the sky background caused by e.g. changes in the position and phase of the moon.[1]

The differential change in PSF and photometric scaling can be found by inferring the *convolution kernel*, which 'blurs' the sharper of the two images to match the PSF in the other. Alard & Lupton (1998) (hereafter AL98) demonstrated that the kernel can be found using linear least-squares by decomposing it as a set of user-specified basis functions. Specifically, AL98 opted for Gaussian basis functions modified by low-order polynomials. A decade later, the linear least-squares solution was advanced by Bramich (2008) (hereafter B08) who modelled the kernel as a highly flexible (albeit computationally more expensive) discrete pixel array, which is analogous to the AL98 algorithm, but with a choice of delta basis functions for the kernel model. Later advances included extending the DIA solution to a spatially varying kernel and differential background to model position-dependent variations in seeing, transparency and airmass across the field-of-view (FoV) in wide-field imaging data (Alard, 2000; Bramich et al., 2013).

In this analytical linear least-squares framework, a normal matrix for the linear system of equations is required. It is the construction of the normal matrix for these approaches which results in a computational bottleneck. For example, for a (square) pair of images each of size n, and a (square) kernel of size m the construction of the normal matrix for the B08 approach – which implements the same model for the kernel as our algorithm – scales as $\mathcal{O}(n^2m^4)$ (i.e. the run-time increases with the square of the input image size and with the kernel size to the power of four). These problems are exacerbated by the need to iteratively

[1]We note that there are other differences between images that cannot be captured by current DIA models, such as differential refraction or extinction, and we do not address these in this work.

fit for the model parameters in the linear least-squares sense (as the problem is in fact non-linear, see Section 2.3.1) and, typically, the normal matrix will need to be constructed about 3-4 times until convergence is reached. Attempts have been made to exploit symmetries in the problem (see Section 5.2 of Bramich et al. (2013)) or bin pixels around the kernel edges to speed up this construction (Albrow et al., 2009). In the era of wide FoV sky surveys such as the upcoming Legacy Survey of Space and Time (LSST) (Ivezić et al., 2019), or the ongoing Zwicky Transient Facility (ZTF) (Bellm et al., 2018) etc., which aim to deliver prompt alerts of transient astronomical phenomena detected through image subtraction, many current popular approaches make real time event discovery impractical. This has driven some recent advances in the DIA literature, most notably the ZOGY algorithm (Zackay et al., 2016), and even a machine-learning approach (Sedaghat & Mahabal, 2018).

Image processing tasks – where some common operation is performed on very many pixels – are inherently massively parallel. Mild to substantial computational speed-up by parallelising the construction of the normal matrix in classical DIA algorithms with GPUs has been demonstrated in the literature (for example, Hartung et al., 2012; Li et al., 2013; Zhao et al., 2013). Adoption by the larger astronomical community however has been slow.

The main barrier to adoption for most astronomers is likely the lack of a working knowledge of CUDA, NVIDIA's parallel computing platform, which is required to develop GPU-accelerated applications on supported devices. In astronomy, the Python programming language has firmly established itself as a favourite of the majority of the community, and most importantly, will be the primary user-interface language for the next generation of astronomical data management, processing and analysis systems (Perkel, 2018). The appropriateness of DIA image models can be highly data set dependent, and tuning by individual researchers to meet their science goals within a Pythonic framework is to be expected. The work presented here is one of a few recent attempts[2] to address the paired issues of performance and accessibility.

In this chapter, we present a novel Pythonic implementation of an alternative route to the B08 solution, without the need for constructing the computationally expensive normal matrix. Conceptually, our approach is unique in that we model the kernel as if it were the convolutional filter of a very simple convolutional neural network (CNN), which can be solved for efficiently with GPU-accelerated optimisation tools originally developed for deep learning applications.

[2]Including approaches attempting to parallise the construction of the B08 normal matrix (Albrow, 2017)

As we will describe, the machine-learning framework which we use to approach this problem also equips us with powerful modelling tools, and frees us from a number of restrictive assumptions inherent to classical approaches. Our implementation is also unique amongst GPU-accelerated DIA algorithms for being written *entirely* with standard Python packages.

We begin Section 2.3 with an introduction of the basic image model used in our DIA implementation. We then outline how PyTorch could be well suited to addressing existing astronomical image modelling and data processing challenges in general, before describing the details of our own DIA implementation, 'PyTorchDIA'. We quantify the model fit quality and photometric accuracy of PyTorchDIA with tests on both synthetic and real images in Sections 2.4 and 2.5, comparing it directly to the performance of its classical DIA analogue, the B08 algorithm. In Section 2.6, we compare the speed of our GPU-accelerated numerical solution against a fast Cython implementation of the B08 algorithm[3] used in the ROME/REA project (Tsapras et al., 2019), and explore how our algorithm scales with image and kernel size. We summarise our conclusions in Section 2.7.

2.3 Problem Formulation

In this section we outline the DIA problem, and provide a motivating overview for using PyTorch as a tool to address image modelling challenges – of which DIA is just one example – and describe the details of our DIA implementation.

2.3.1 Difference Image Analysis

Given a reference image with pixels R_{ij}, ideally of excellent spatial resolution and high signal-to-noise, and a target image with pixels I_{ij}, of the same scene taken at some other epoch and aligned on the same pixel grid, the model image which represents I_{ij} is given by

$$M_{ij} = [R \otimes K]_{ij} + B_{ij} \,, \tag{2.1}$$

and so the challenge is to find the fit for an accurate kernel, K, and differential background B_{ij}.

Following B08, by representing the kernel as a discrete array K_{lm} containing a total of N_K

[3]The kernel solution method heavily borrows from the relevant section of the pyDANDIA microlensing reduction pipeline, `https://github.com/pyDANDIA`.

pixels, and including an additive scalar differential background, B_0, we can re-write equation 2.1 as

$$M_{ij} = \sum_{lm} K_{lm} R_{(i+l)(j+m)} + B_0 \,. \tag{2.2}$$

The photometric scale factor – which encodes any differences in atmospheric transparency and/or exposure time between the images – is simply the sum of the kernel pixels,

$$P = \sum_{lm} K_{lm} \,. \tag{2.3}$$

Assuming that the pixel values of the target image, I_{ij}, are independently drawn from normal distributions $\mathcal{N}(M_{ij}, \sigma_{ij}^2)$ – where M_{ij} is paramaterised by the vector $\boldsymbol{\theta} = [K_{lm}, B_0]$ (see Section 2.3.3) – the negative log-likelihood function for the target image takes the form

$$-\ln \mathrm{p}(I_{ij}|\boldsymbol{\theta}) = \frac{1}{2}\chi^2 \; + \sum_{ij} \ln \; \sigma_{ij} + \frac{N_{\mathrm{data}}}{2} \ln(2\pi) \,, \tag{2.4}$$

where σ_{ij} are the pixel uncertainties, N_{data} is the number of valid pixels in the target image, and the χ^2 is equal to

$$\chi^2 = \sum_{ij} \left(\frac{I_{ij} - M_{ij}}{\sigma_{ij}} \right)^2 \,. \tag{2.5}$$

It is important to note that N_{data} is not equal to the *total* number of pixels in the target image, as the convolution operation is undefined for pixels within half a kernel's width from the target image edges i.e. for a kernel of size $(2n+1) \times (2n+1)$ and (square) reference and target images of size $N \times N$, $N_{\mathrm{data}} = (N - 2n)^2$.

The σ_{ij} pixel uncertainties are dependent on the image model M_{ij}, and so fitting this model to the target image I_{ij} is therefore a non-linear optimisation task. Linear least-squares approaches like AL98 and B08 must then approach this iteratively, by minimising the χ^2 with fixed estimates for σ_{ij}. After the first estimate of M_{ij} is acquired, the updated σ_{ij} are computed, and then plugged into the χ^2 for the next iteration. This process continues for at least 3 iterations, until some convergence condition is met.

In this work, we use both simulated CCD images and real Electron Multiplying CCD (EMCCD) images, acquired on the Danish 1.54m (DK154) Lucky Imaging camera (Skottfelt et al., 2015) to verify our implementation, each of which requires a different noise model. In what follows, we ignore the noise contributions from the reference image, since these are negligible

in the experiments performed in this work. [4].

For CCD images, we adopt a noise model for the σ_{ij} pixel uncertainties of M_{ij} as

$$\sigma_{ij}^2 = \frac{\sigma_0^2}{F_{ij}^2} + \frac{M_{ij}}{G\,F_{ij}} \tag{2.6}$$

where σ_0 is the read noise (ADU), G is the detector gain (e^-/ADU), and F_{ij} is the master flat field.

The EMCCD images are constructed from typically thousands of shift-and-added sub-second exposures. The electron multiplying gain reduces the read out noise to negligible levels, but the cascade amplification process effectively doubles the variance of the photon noise. For the real EMCCD images used in this work, we adopt a noise model of the form

$$\sigma_{ij}^2 = E\frac{M_{ij}}{G_{\rm Total}\,F_{ij}}, \tag{2.7}$$

where $G_{\rm Total}$ represents the combined gain ($e^-_{\rm phot}$/ADU) of the DK154 Lucky Imaging camera, which is calculated as the ratio of the CCD gain of 25.8 ($e^-_{\rm EM}$/ADU) over the electron-multiplying (EM) gain of 300 ($e^-_{\rm EM}/e^-_{\rm phot}$). Following Harpsøe et al. (2012), we differentiate between electrons before and after the cascade amplification with the notation $e^-_{\rm phot}$ and $e^-_{\rm EM}$ respectively. E represents the 'excess noise factor', and accounts for the probabilistic nature of the cascade amplification as the EMCCD is read out, and is set to be $E = 2$. EMCCD images are flat corrected in the same way as conventional CCD images, and as in Equation 2.6, F_{ij} is the master flat field.

2.3.2 Astronomical Image Processing with PyTorch

The model image of Equation 2.1 is a convolution with some added scalar constant. In the computer science literature, this is *exactly* analogous to an extremely simple convolution neural network (CNN), with a single input and output related by a single convolutional filter, with some additional scalar bias added to the output. Efficient solutions for the 'weights' (the kernel) and 'bias' (background term) to this convolution operation can be implemented within the Python package, PyTorch (Paszke et al., 2019).

[4]Our interest in testing the performance of our DIA algorithm on DK154 EMCCD images relates to our ongoing microlensing follow-up campaign, which is the main science project of the MiNDSTEp consortium, `http://www.mindstep-science.org/`

PyTorch is a popular open source machine learning framework. Its constant development is motivated by the impressive advances in deep learning (for an overview see e.g. (LeCun et al., 2015; Goodfellow et al., 2016)), in which CNNs have played an important role in processing images. Specifically, this package supports CUDA-enabled GPU acceleration and automatic differentiation to perform efficient optimisations. One key motivation behind these developments is the *training* of complex CNNs, consisting of thousands, to hundreds of thousands of parameters. The 2-dimensional convolution (as in Equation 2.1) is *the* core processing operation in these networks, and is straightforward to implement in PyTorch's modelling architecture. Indeed, many useful image models in astronomy can be written as a convolution, and the tools outlined in this work are therefore broadly applicable to many astronomical image processing problems.

We stress that although PyTorch's powerful tools were designed for machine learning, they can be turned to generic data analysis and modelling problems, as shown in this work, of which image models are just a subset. In particular, efficient computation of gradients via automatic differentiation frees the user from having to manually recompute gradients if the parameterisation of the model is changed. This flexibility allows models written in PyTorch to be easily tuned to meet the variety of science goals arising from a diversity of data sets. In astronomical image processing in particular, this flexibility combined with GPU acceleration could make PyTorch a valuable tool for addressing challenges associated with both model complexity and data volume. As PyTorch is Pythonic, these implementations could be easily integrated into existing Python stacks.

2.3.3 Difference Image Analysis as an Optimisation

The crucial advance in AL98 was to formulate DIA as a linear least-squares problem, and several efficient algorithms for solving these problems exist (Golub & Van Loan, 1996). The computational bottleneck associated with this approach is the construction of the normal matrix (see Section 2.2). Models in PyTorch however are generally fit with a numerical optimisation procedure – making use of automatic differentiation – which we outline here.

For a given target image with Gaussian noise contributions, I_{ij}, and current estimates for the kernel K_{lm} and differential background B_0 which transform the reference image, R_{ij}, we define our vector of weights as $\boldsymbol{\theta} = [K_{lm}, B_0]$. Ignoring the irrelevant normalisation constant, the maximum-likelihood-estimate (MLE) for $\boldsymbol{\theta}$ can be found by minimising the overall *loss* (or

negative log-likelihood Cf. Equation 2.4),

$$\begin{aligned}&\underset{\boldsymbol{\theta}=[K_{lm},B_0]}{\arg\min}\ \mathscr{L}_0(\boldsymbol{\theta}) = \\ &= \underset{\boldsymbol{\theta}}{\arg\min}\ \left[\frac{1}{2}\sum_{ij}\left(\frac{I_{ij}-M_{ij}(\boldsymbol{\theta})}{\sigma_{ij}(\boldsymbol{\theta})}\right)^2 + \sum_{ij}\ln\ \sigma_{ij}(\boldsymbol{\theta})\right]\ .\end{aligned} \tag{2.8}$$

We note here that both the image model and noise model are both functions of $\boldsymbol{\theta}$. We drop the notation explicitly showing the dependence of M_{ij} and σ_{ij} on $\boldsymbol{\theta}$ from the rest of this chapter.

Equation 2.8 can be solved by a process of (steepest) gradient descent. At each iteration t in the optimisation, we use the update rule

$$\boldsymbol{\theta}^{(t+1)} \longleftarrow \boldsymbol{\theta}^{(t)} - \alpha^{(t)}\nabla_{\boldsymbol{\theta}^{(t)}}\mathscr{L}_0(\boldsymbol{\theta}^{(t)})\,, \tag{2.9}$$

where α^t is the *learning rate* (or step-size) at each iteration. PyTorch includes implementations of several more sophisticated gradient descent algorithms, which can compute *adaptive* learning rates for *each* parameter (see Ruder 2016 for a good overview). In our implementation, we use the Adaptive Moment Estimation (Adam) algorithm , which computes learning rates for the kernel pixels and the background parameters from estimates of the first and second moments of their gradients at each step (Kingma & Ba, 2014). Adam has been empirically demonstrated to work well on non-convex optimisation problems, is computationally efficient, and the hyper-parameters are both intuitive and require little tuning from the astronomer (see Section 2.3.9 for an overview of the 'engineering' aspects of the optimisation).

With judicious choices for both the learning rates and the parameter initialisations, solutions via steepest descent when accelerated on the GPU are lightning fast (see Section 2.6), but without this hardware acceleration, this approach in general can be slow. Solving this problem on the CPU – and therefore forgoing the massive inherent parallelism otherwise exploited in Equation 2.9 – would result in a substantial performance hit. This problem is particularly severe when close to the minimum of the loss function, where the gradients become increasingly shallow and the gradient steps become correspondingly smaller. An effective solution to this problem is to make use of quasi-Newton optimisation methods which approximate the curvature of the loss surface.

These approaches converge extremely quickly where the loss surface can be modeled quadratically, and use an approximation of the inverse of the Hessian at each t iteration, $B(\boldsymbol{\theta})^{(t)}$, to condition the search direction, giving the update rule

$$\boldsymbol{\theta}^{(t+1)} \longleftarrow \boldsymbol{\theta}^{(t)} - \alpha^{(t)} B^{-1}(\boldsymbol{\theta})^{(t)} \nabla_{\boldsymbol{\theta}^{(t)}} \mathscr{L}_0(\boldsymbol{\theta}^t)\,. \tag{2.10}$$

For this convex optimisation, this is a very good assumption when close to the minimum. Newton methods in general are sensitive to bad parameter initialisation (i.e. when we're initially far from the minimum), and so we advocate for optimising via steepest descent steps – as in Equation 2.9 – to first get close to the minimum, before converging with the more memory intensive quasi-Newton procedure once the relative change in the loss between any two optimisation steps falls below a user-specified threshold. Specifically, we use a L-BFGS method (see Chapter 7.2 of Nocedal & Wright 2006) which is available as an in-built algorithm in PyTorch[5].

2.3.4 Robust loss function

In general, the assumption that all the pixel values in the target image are drawn from $\mathcal{N}(M_{ij}, \sigma_{ij}^2)$ will be violated for real images. Instrumental defects, cosmic ray hits, saturated pixels, and variable or transient sources etc. will all arise as outlying pixel values. Our Gaussian loss function (Equation 2.4) will be badly affected by these outliers, as it minimises the squared difference between the model and the data. The standard approach to outlier rejection in astronomy is to iteratively remove these 'bad' pixels by *sigma-clipping*, and B08 use this approach to mitigate the impact of outliers to the least-squares solution. In addition to being very sensitive to the accuracy of the adopted noise model, this *procedure* does not explicitly penalise the rejection of data, nor is it necessarily clear how many iterations should be performed. In the context of classical DIA algorithms, these iterations incur an extra computational expense, as a new normal matrix must be constructed each time.

As we are not restricted to standard least-squares minimisation with our optimisation procedure, we have the freedom to choose *robust* alternatives to Equation 2.8 which are less sensitive to outlying values. Huber (1992) identified a family of wide-tailed univariate distributions. The 'ϵ-contaminated' Gaussian model in particular corresponds to a unimodal sym-

[5] `https://pytorch.org/docs/stable/optim.html`

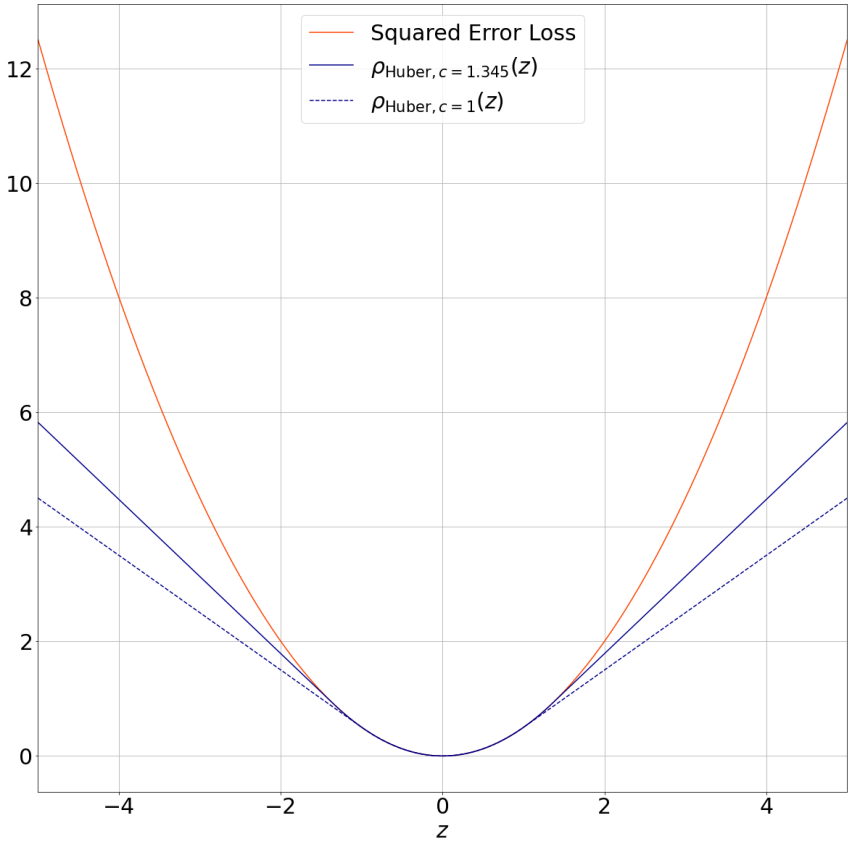


Figure 2.1: A comparison of squared error loss against Huber loss for different values of c.

metric distribution which resembles a Gaussian for central values and a Laplacian in the tails. The probability density function (up to a normalising constant) with location and scale parameters μ and σ has the form

$$f(x) \propto \frac{1}{\sigma} \exp\left[-\rho_{\mathrm{Huber},c}\left(\frac{x-\mu}{\sigma}\right)\right], \tag{2.11}$$

Substituting $z = (x-\mu)/\sigma$, $\rho_{\mathrm{Huber},c}(z)$ is the loss function, which is defined as

$$\rho_{\mathrm{Huber},c}(z) = \begin{cases} \frac{1}{2}z^2, & \text{for } |z| \leq c \\ c(|z| - \frac{1}{2}c), & \text{for } |z| > c\,. \end{cases} \tag{2.12}$$

We see that $\rho_{\mathrm{Huber},c}(z)$ computes the squared error between the model and data for small residuals, and the absolute deviation for outliers above some user-specified threshold, c. This threshold defines the switch between quadratic and linear treatment of the error (see Figure 2.1), and Huber & Ronchetti (2009) recommend $c = 1.345$ as a suitable value for enforcing robustness while retaining reasonable efficiency for normally distributed data.

Using the same notation as in Section 2.3.1, if we assume the pixel values in the target image I_{ij} are drawn from the distribution in Equation 2.11, the negative log-likelihood function takes the general form

$$-\ln p(I_{ij}|\boldsymbol{\theta}) = \sum_{ij} \rho_{\text{Huber},c}\left(\frac{I_{ij}-M_{ij}}{\sigma_{ij}}\right) + \sum_{ij} \ln \sigma_{ij} + Q\,, \tag{2.13}$$

where Q is the normalising constant,

$$Q = N \ln\left(\sqrt{2}\,\text{erf}\left(\frac{c}{\sqrt{2}}\right) + \frac{2\exp\left[\frac{-c^2}{2}\right]}{c}\right). \tag{2.14}$$

Note that when c becomes large, $Q = (N/2) \times \ln 2\pi$, which is equivalent to the normalisation constant of a Gaussian log-likelihood (Cf. Equation 2.4).

As our implementation makes use of automatic differentiation, it is straightforward for the user to experiment with loss functions, as they are freed from having to manually recompute gradients. We highlight the Huber loss in particular as this enforces robustness without sacrificing performance (see Section 2.6.1), but any number of wide-tailed distributions (e.g. Student's t distribution) could be used as drop-in replacements.

2.3.5 Uncertainty estimation - Observed Fisher Information

The central limit theorem states that than any well-behaved likelihood function approaches a Gaussian near its maximum. The *Fisher Information Matrix* (FIM) is a measure of the curvature of the likelihood function with respect to the model parameters – intuitively, this can be thought of as a 'sensitivity' – and its inverse provides a lower bound on the asymptotic variance of the Maximum Likelihood Estimate (MLE).

The observed FIM, $\boldsymbol{F}(\boldsymbol{\theta})$ for our N_K+1 parameters is the $(N_K+1,\ N_K+1)$ matrix containing the entries

$$F_{pq} = \frac{\partial^2}{\partial\theta_p \partial\theta_q} \ln \text{p}(I_{ij}|\boldsymbol{\theta})\,,\ 1 \le p,\ q \le N_K+1\,, \tag{2.15}$$

where $\ln \text{p}(I_{ij}|\boldsymbol{\theta})$ is the log-likelihood (Equation 2.4).

The inverse of $\boldsymbol{F}(\boldsymbol{\theta})$ evaluated at the MLE for $\boldsymbol{\theta}$ (i.e. $\hat{\boldsymbol{\theta}}_{\text{MLE}}$) can then be used as an estimate

for the covariance matrix

$$\hat{\boldsymbol{\Sigma}} = \mathrm{Cov}(\hat{\boldsymbol{\theta}}_{\mathrm{MLE}}) = [\boldsymbol{F}(\hat{\boldsymbol{\theta}}_{\mathrm{MLE}})]^{-1} \,, \tag{2.16}$$

which provides an estimate of the uncertainties on the model parameters by taking the square roots of the diagonal elements of $\mathrm{Cov}(\hat{\boldsymbol{\theta}}_{\mathrm{MLE}})$. Fisher information provides us with the limiting precision with which model parameters can be estimated for any given data set i.e. the subsequent error bars cannot be smaller (Heavens, 2009). Formally, the *Cramér-Rao* inequality states that the uncertainty on some parameter θ_p is given by

$$\Delta\theta_p \geq (F^{-1})_{pp}^{1/2} \,. \tag{2.17}$$

This method is subject to two assumptions 1) the likelihood function correctly describes the data generating process (i.e. the error distribution of the measurements is correctly described by the likelihood), and 2) the likelihood really is approximately Gaussian at the MLE. In practice, both of these assumptions will likely be violated, and in these situations, the uncertainty estimates by this approach can be severely underestimated. Given this, Andrae (2010) strongly recommend to test this assumption by checking the validity of $\hat{\boldsymbol{\Sigma}}$. In general, a valid covariance matrix $\hat{\boldsymbol{\Sigma}}$ must be positive definite (i.e. for any non-zero vector $\boldsymbol{x}$, $\boldsymbol{x}^T.\hat{\boldsymbol{\Sigma}}.\boldsymbol{x} > 0$). If either the determinant of $\hat{\boldsymbol{\Sigma}}$ is negative, or (after diagonalising the matrix) any eigenvalue is found to be negative or zero, then $\hat{\boldsymbol{\Sigma}}$ is not valid. We include both these tests in our code release, and a warning flag is raised if any is failed.

2.3.6 Extension to a Spatially Varying Background

Within the PyTorch architecture, a spatially varying background can be easily modelled by replacing B_{ij} in Equations 2.1 and 2.8 with a linear combination of functions of x and y. We adopt a polynomial model of some user-specified degree d,

$$B(x,y) = \sum_{m=0}^{d} \sum_{n=0}^{d-m} b_{mn}\, \eta(x)^m \xi(y)^n \tag{2.18}$$

where b_{mn} are the polynomial coefficients to be inferred, and $\eta(x)$ and $\xi(y)$ are the nor-

malised spatial coordinates,

$$\begin{aligned} \eta(x) &= (x - x_c)/N_x \ , \\ \xi(y) &= (y - y_c)/N_y \ , \end{aligned} \tag{2.19}$$

which result from a Taylor expansion of coordinates (x, y) about the image centre (x_c, y_c), for an image of $N_x \times N_y$ pixels. This choice of coordinates is recommended by Bramich et al. (2013), and has the effect of (1) improving the orthogonality of the spatial polynomial terms, and (2) preventing the coefficients of the higher order polynomial terms from being pushed towards zero.

2.3.7 Regularising the kernel pixels

As noted by Becker et al. (2012), while kernels modeled as a discrete array of pixels are very flexible, the consequent fidelity with the data can result in significant overfitting. To guard against excessively noisy kernels, we provide the option to regularise the loss with the addition of a penalty term. Following the notation in Bramich et al. (2016) (hereafter B16) – where the strength of the regularisation is controlled by the parameter λ, which must be tuned empirically – the scalar objective function to minimise now takes the form,

$$\arg\min_{\boldsymbol{\theta}} \mathscr{L}(\boldsymbol{\theta}) = \arg\min_{\boldsymbol{\theta}} \left[\mathscr{L}_0(\boldsymbol{\theta}) + \lambda N_{\mathrm{data}} \boldsymbol{\theta}^{\mathrm{T}} \boldsymbol{L}^{\mathrm{T}} \boldsymbol{L} \boldsymbol{\theta} \right] , \tag{2.20}$$

where N_{data} is the number of target image pixels, $\boldsymbol{\theta}$ is the vector of parameters to optimise, $\mathscr{L}_0(\boldsymbol{\theta})$ is the loss function, and $\boldsymbol{L}$ is the symmetric and positive-semidefinite $(N_K+1)\times(N_K+1)$ *Laplacian matrix*, which represents the connectivity graph of the set of kernel pixels, and has elements

$$L_{uv} = \begin{cases} N_{\mathrm{adj},u}, & \text{for } v = u \leq N_K, \text{ and } N_{\mathrm{adj},u} \text{ is the number of} \\ & \text{kernel pixels adjacent to the kernel pixel} \\ & \text{corresponding to } u \\ -1, & \text{for } u \leq N_K, v \leq N_K, v \neq u, \text{ and } u \text{ and } v \text{ are} \\ & \text{adjacent kernel pixels.} \\ 0, & \text{otherwise.} \end{cases} \tag{2.21}$$

This penalty term is derived from an approximation of the second derivative of the kernel

surface (Becker et al., 2012). Intuitively, it favours compact kernels, where adjacent kernel pixels should not vary too sharply. The optimal value of λ should, ideally, be tuned for each image, and B16 found optimal values could be anywhere in the range $\lambda = 0.1 - 100$.

By regularising the kernel weights, we are in effect introducing a Bayesian prior, which would then transform our solution into a *maximum a posteriori* (MAP) optimisation. In our experiments in the following sections, we restrict ourselves to the MLE solution by simply minimising either Equation 2.8 or 2.13.

2.3.8 General purpose computing on GPUs

For this work, we ran our PyTorchDIA implementation on two different NVIDIA GPUs. The computations associated with the results presented in Section 2.4 and 2.5 were performed on a GeForce GTX 1050 on a laptop, and the speed tests in Section 2.6 were run on a Tesla K80 on Google's Colab[6] service, a free-to-use cloud-based computation environment. There are a couple of important details worth describing on using these devices for scientific computing.

Firstly, while modern CPUs are able to handle computations on 64-bit floating point numbers efficiently, such operations are either not supported on GPUs, or are associated with a significant reduction in performance (Göddeke et al., 2005). It is for this reason that 32-bit precision – or increasingly commonly, 16-bit precision – is used in deep learning. For this problem, we have found computations typically take at least $2 - 5$ times longer for fiducial image sizes using 64-bit precision with a Tesla K80 [7]. For memory intensive optimisers such as L-BFGS, the hit to performance will be even worse. For this reason, for the results in this work we use 32-bit floating point precision for our PyTorchDIA implementation. This speed-up comes at the expense of some accuracy, but we show this difference to be small in Sections 2.4 and 2.5.

Convolution computations on NVIDIA GPUs can be accelerated by highly-tuned algorithms available in specialised libraries. One such NVIDIA library, cuDNN, specialises in efficient computations with small kernels. These algorithms are selected by cuDNN heuristically, and not all are deterministic. As such, for an identical pair of images, the number of computations

[6] `https://colab.research.google.com`

[7] Tesla cards are designed to perform fast computations at higher precision than most GPU models, and a slowdown of $\sim 2 - 5$ at 64-bit would still outperform some state-of-the-art classical DIA methods (see Section 2.6). Most NVIDIA models however (including GeForce cards), excel at 32-bit and lower precisions, but would suffer a more severe slowdown at 64-bit.

used to infer the convolution kernel may be different on different runs. Consequently, the optimiser may converge at a slightly different point in the parameter space. We explore the speed-up associated with enabling cuDNN tools in Section 2.6.

Finally, while not explored in this work, we highlight *mixed precision* 'training' as a technique which our approach could benefit from. PyTorch now supports Automatic Mixed Precision (AMP) training, in which 32-bit data can be automatically cast to half precision for some types of computationally expensive operations on the GPU[8]. By appropriately scaling the loss during the training/optimisation, AMP prevents *underflow* that would otherwise cause gradients to drop to 0 at half precision, and can achieve the same accuracy as training only with 32-bit precision with significant speed-up, depending on the GPU architecture and model design (Micikevicius et al., 2017). Some recent NVIDIA GPUs now include *Tensor cores*, which are specifically designed to perform highly optimised 16-bit matrix multiplications. Consequently, cuDNN convolution algorithms such as 'GEMM' are particularly well suited to benefit from this technique (Jordà et al., 2019).

2.3.9 Optimisation as an engineering problem

As we fit for the convolution kernel via an optimisation, hardware alone will not determine the solution time. There is an 'engineering' aspect to optimisation that the user should be aware of. Here, we summarise three key choices that the user must make: 1) Parameter initialisation; 2) The learning rate (or step size) and 3) The convergence condition. We also provide an overview of (and justification for) the specific choices made when generating the results presented in this manuscript.

Throughout this work, we *always* background subtract the reference image, and so we initially set the differential background parameter, B_0, to be equal to the median pixel value of the given target image. For the fit quality and photometric accuracy tests in Sections 2.4 and 2.5, we make no assumptions about the shape of the kernel at initialisation. We set all kernel pixels to have the same value, with the only condition being that they sum to 1 (i.e. each pixel is initialised as $1/N_K$). In Section 2.6, we experiment with initialising the kernel as a symmetric Gaussian, with a shape parameter judiciously set with knowledge of either the known or measured differences in the PSFs of the reference and target image pair.

[8] `https://pytorch.org/docs/stable/notes/amp_examples.html`

The Adam algorithm used at the start of the optimisation – with which we perform steepest gradient descent – allows us the freedom to set individual learning rates for our model parameters, which will be adaptively tuned (see Section 2.3.3). For all tests in this work, we set the learning rate of the Adam optimizer for the pixels in the kernel at 0.001, as recommended in Kingma & Ba (2014). We have found that it is advantageous to use a fairly high learning rate for B_0, and we set this to either 1, 10 or even 100, dependent on the data set. There is a strong anticorrelation between P and B_0, and quickly finding the approximate photometric scaling between the two images allows us to disentangle this offset from the inference of the spatial differences in the images (associated with the different PSFs), which is encoded in the shape (and not the 'scale') of the convolution kernel. The learning rate of the L-BFGS optimiser is always set to 1.

We use the same convergence condition for all tests in this work. This decision was made on the basis of the model performance and photometric accuracy metrics in Sections 2.4 and 2.5 , and should balance a satisfactory model accuracy with the time taken to converge. If between any two steepest descent updates the relative change in the loss is less than 10^{-6} (i.e. we are getting close to the minimum) a L-BFGS optimiser takes over. For each subsequent quasi-Newton update, the L-BFGS function evaluation routine terminates when either the change in loss between evaluations reaches the limit of our numerical precision – which corresponds to a relative change in loss in the range $10^{-8} - 10^{-7}$ between the last two optimisations steps – or a first order optimality condition is met, such that the gradient of the scalar objective function to be minimised with respect to each model parameter is less than 10^{-7}.

2.3.10 The Algorithm

Our DIA algorithm is as follows. Decisions to be made by the user are in italics. 'tol' is the user-specified relative change in the loss between successive steepest descent (SD) updates, at which point the optimisation routine switches from SD to L-BFGS.

- *Choose the square kernel dimensions (must be odd), and whether to use a scalar, or polynomial fit of degree d, for the differential background*
- *Set Adam's per-parameter learning rates and 'tol'*
- *Set the convergence condition*

- *Initialise* $\boldsymbol{\theta}$
- Begin minimising the chosen scalar objective function (Equations 2.4 or 2.13, with or without the optional penalty term (Equation 2.20)) with steepest gradient descent (Equation2.9)
- At each iteration **if** the relative change in the loss is less than 'tol', switch to a quasi-Newton update (Equation 2.10) and **continue** until convergence condition satisfied.

2.4 Simulated Image tests

Artificially generated images can be used to assess the accuracy of our algorithm. In this section, we compare the fit quality and photometric accuracy of our numerical, GPU-accelerated algorithm against an implementation of the analytic B08 algorithm with a data set of 71569 synthetic images.

2.4.1 Generating Artifical Images

We base our image generation procedure on Section 5.1 of B16. The results from the simulated image tests in that work were shown to closely agree with similar tests using real CCD data. Specifically, we generate images similar to those in the 'S10' set of that work, wherein the reference image of each image pair has 10 times less variance than the target.

We first generate a noiseless reference, $R_{\text{noiseless},ij}$, of size 142×142 pixels as follows.

(i) We generate a normalised two-dimensional symmetric Gaussian PSF for this image, parameterised by a standard deviation, ϕ_R, drawn from the continuous uniform distribution $U \sim (0.5, 2.5)$.

(ii) We populate the 142×142 pixel array template of $R_{\text{noiseless},ij}$ with N_{stars}, whose lg density $\lg \rho_{\text{stars}}$ per 100×100 pixel region, is drawn from the uniform distribution $U \sim (0, 3)$. The fractional pixel coordinates for each star are uniformly drawn between the image axis lengths.

(iii) For each of the N_{stars}, we draw a value of $\mathscr{F}^{-3/2}$ from the uniform distribution $U \sim (10^{-9}, 10^{-9/2})$, where $\mathscr{F}$ is a given star's flux (ADU). This flux distribution is a good approximation when imaging to a fixed depth for certain regions in the Galaxy. For reasons of performing PSF photometry at the position of the brightest star in the difference images, we move the pixel

coordinates of the star associated with the largest $\mathscr{F}$ value to the centre of the image.

(iv) For each star, the normalised Gaussian profile generated in (i) is placed at the appropriate pixel coordinates, and scaled by the star flux from (iii).

(v) Finally, a constant background level S_R is drawn from the continuous distribution, $U \sim (10, 1000)$, is added to the image.

(vi) It is common practice to create a high signal-to-noise reference frame by either stacking images or increasing the exposure time. To simulate this, we generate a 'noise map', $\sigma_{R,ij}$ to apply to $R_{\text{noiseless},ij}$ with 10 times less pixel variance than is applied to the target image. This can be achieved by scaling the uncertainties by a factor of $10^{-1/2} \sim 0.316$. Adopting the usual CCD noise model (Equation 2.6) with $\sigma_0 = 5$ (ADU), $G = 1$ (e^-/ADU) and $F_{ij} = 1$, we compute the reference frame pixel uncertainties as

$$\sigma_{R,ij} = 10^{-1/2} \sqrt{\sigma_0^2 + R_{\text{noiseless},ij}} \, . \tag{2.22}$$

(vii) A 142×142 pixel image, W_{ij}, with values drawn from a standard normal distribution, $\mathcal{N}(0, 1)$, was also generated, and the noisy reference, $R_{\text{noisy},ij}$, is formed as

$$R_{\text{noisy},ij} = R_{\text{noiseless},ij} + W_{ij} \sigma_{R,ij} \, . \tag{2.23}$$

For each $R_{\text{noisy},ij}$ we then generate a corresponding target image.

(viii) As with the reference images, we choose a symmetrical Gaussian PSF for the target images, paramaterised by ϕ_I. As the convolution of a Gaussian with a Gaussian is another Gaussian, the kernel is itself a Gaussian, and we draw the corresponding kernel width ϕ_K from $U \sim (0.5, 2.5)$. We can then compute the width of the PSF in the target image,

$$\phi_I^2 = \phi_R^2 + \phi_K^2 \, , \tag{2.24}$$

and repeat steps (iv) - (v), using ϕ_I in place of ϕ_R and S_I in place of S_R to generate $I_{\text{noiseless},ij}$. Additionally, we also apply sub-pixel shifts to the stellar fractional pixel positions along both the x- and y-axis, with the Δx and Δy shifts each drawn separately for each simulation from $U \sim (-0.5, 0.5)$.

(ix) Similar to (vi), we then generate the noise map

$$\sigma_{I,ij} = \sqrt{\sigma_0^2 + I_{\text{noiseless},ij}} \,, \tag{2.25}$$

(x) and then generate a 142×142 pixel image, W_{ij}, with values drawn from $\mathcal{N}(0,1)$, and create $I_{\text{noisy},ij}$,

$$I_{\text{noisy},ij} = I_{\text{noiseless},ij} + W_{ij}\sigma_{I,ij} \,. \tag{2.26}$$

The signal-to-noise ratio of the target image is computed as

$$\text{SNR}_I = \frac{\sum_{ij}(I_{\text{noiseless},ij} - S_I)}{\sqrt{\sum_{ij}\sigma_{I,ij}^2}} \,. \tag{2.27}$$

2.4.2 Performance Metrics

We assess the fit quality and photometric accuracy of the kernel and background solutions with the following performance metrics. The derivations of these metrics are based on Section 5.2 of B16. For easy reference, the definitions and equation numbers for all metrics are listed in Table 2.1.

(i) Model error. The mean squared error (MSE) assesses how well the inferred model image, M_{ij}, matches the true target image, $I_{ij,\text{noiseless}}$,

$$\text{MSE} = \frac{1}{N_{\text{data}}}\sum_{ij}(M_{ij} - I_{\text{noiseless},ij})^2 \,. \tag{2.28}$$

The smallest values of MSE indicate the best fit in terms of model error.

As noted in Bramich et al. (2015), systematic errors in the photometric scale factor, P, can badly influence photometric accuracy. Given this, we also assess the inferred P and B_0 parameters, whose true values are equal to 1 and 0 respectively in these simulations.

(ii) Fit quality. The mean fit bias (MFB) and mean fit variance (MFV) are measures of the

bias and excess variance in $M_i j$, and are given as

$$\mathrm{MFB} = \frac{1}{N_{\mathrm{data}}} \sum_{ij} \frac{I_{\mathrm{noisy},ij} - M_{ij}}{\sigma_{I,ij}} \,, \tag{2.29}$$

$$\mathrm{MFV} = \frac{1}{N_{\mathrm{data}} - 1} \sum_{ij} \left(\frac{I_{\mathrm{noisy},ij} - M_{ij}}{\sigma_{I,ij}} - \mathrm{MFB} \right)^2 \,. \tag{2.30}$$

Kernel and background solutions with a MFB close to 0, and a MFV close to unity are measured to have a good fit quality.

(iii) Photometric accuracy. To assess the photometric accuracy of the solution, we perform PSF fitting photometry at the position of the brightest star in the difference image. We generate a normalised PSF object parameterised by ϕ_R, centered at the position of the brightest star, and convolve this with the kernel solution. This is renormalised, giving us a normalised PSF object to fit to the difference image. The true target image PSF width is known by Equation 2.24, and we set the size of the renormalised PSF object 'stamp' to be $9\phi_I \times 9\phi_I$ pixels large. We then fit this PSF stamp to an equally sized cutout from the difference image (centered at the position of brightest star) with a scaling factor $\mathscr{F}_{\mathrm{diff}}$ and an additive constant, weighted with the known pixel variances in the target image, $\sigma^2_{I,ij}$. The fitted $\mathscr{F}_{\mathrm{diff}}$ is then scaled to the photometric scale of the reference image, giving $\mathscr{F}_{\mathrm{measured}} = \mathscr{F}_{\mathrm{diff}}/P$.

The theoretical minimum variance of $\mathscr{F}_{\mathrm{measured}}$ for a PSF-fitting procedure which scales a PSF model to a stamp with pixel indices rs is

$$\sigma^2_{\mathrm{min}} = \frac{1}{P^2_{\mathrm{true}}} \left(\sum_{rs} \frac{\mathscr{P}^2_{I,rs}}{\sigma^2_{I,rs}} \right)^{-1} \,, \tag{2.31}$$

where P_{true} is the true photometric scale factor (equal to 1 in our simulations) and $\mathscr{P}_{I,rs}$ is the true PSF of the brightest star in the target image i.e. a normalised Gaussian with standard deviation ϕ_I.

As there are no variable sources, over N_{sim} simulations of accurate kernel and background solutions we should expect a distribution of $\mathscr{F}_{\mathrm{measured}}/\sigma_{\mathrm{min}}$ values with mean 0 and a variance of unity. The mean photometric bias (MPB) and mean photometric variance (MPV) over N_{sim} simulations, each indexed by k, would be equal to the statistics

Metric	Definition	Equation
P	Photometric Scale Factor	2.3
MSE	Mean Squared Error	2.28
MFB	Mean Fit Bias	2.29
MFV	Mean Fit Variance	2.30
MPB	Mean Photometric Bias	2.32
MPV	Mean Photometric Variance	2.33

Table 2.1: The model accuracy, fit quality, and photometric performance metrics used to assess the DIA implementations in this work.

$$\mathrm{MPB} = \frac{1}{N_{\mathrm{sim}}} \sum_k \frac{\mathscr{F}_{\mathrm{measured},k}}{\sigma_{\mathrm{min},k}} \tag{2.32}$$

$$\mathrm{MPV} = \frac{1}{N_{\mathrm{sim}} - 1} \sum_k \left(\frac{\mathscr{F}_{\mathrm{measured},k}}{\sigma_{\mathrm{min},k}} - \mathrm{MPB} \right)^2 . \tag{2.33}$$

As noted in B16, although the MPV should be close to unity, it may have values less than this when the target image is overfitted, and/or when the model PSF fitted to the difference image (i.e. formed from the convolution of the reference image PSF with the inferred kernel) is different than the true PSF of the target image.

Figure 2.2 shows an example 142×142 [pixels] reference and target image pair generated for these tests. The difference images and kernels corresponding to the B08 and PyTorchDIA solutions, annotated with the fit quality and photometric accuracy metrics, are shown below.

2.4.3 Simulated Image Test results

In these tests, both B08 and PyTorchDIA attempt to fit the image model in Equation 2.1, with an unregularised (square) 19×19 pixel kernel. The generated (square) reference and target images are each 142×142 pixels large, and so the number of target image pixels which enter the fit is $N_{\mathrm{data}} = (142 - 2 \times 9)^2 = 15376$. B16 used 141×141 large reference images and included 10201 target image pixels in their solution, and so the results presented here can be meaningfully compared against that prior work.

Following B16, we first divide the results of our 71569 simulation tests into 3 regimes by the signal-to-noise ratio (SNR) of the target image, I: (1) $8 < \mathrm{SNR}_I < 40$; (2) $40 < \mathrm{SNR}_I < 200$ and (3) $200 < \mathrm{SNR}_I < 1000$. Each of these 3 categories is then divided into a 4 further

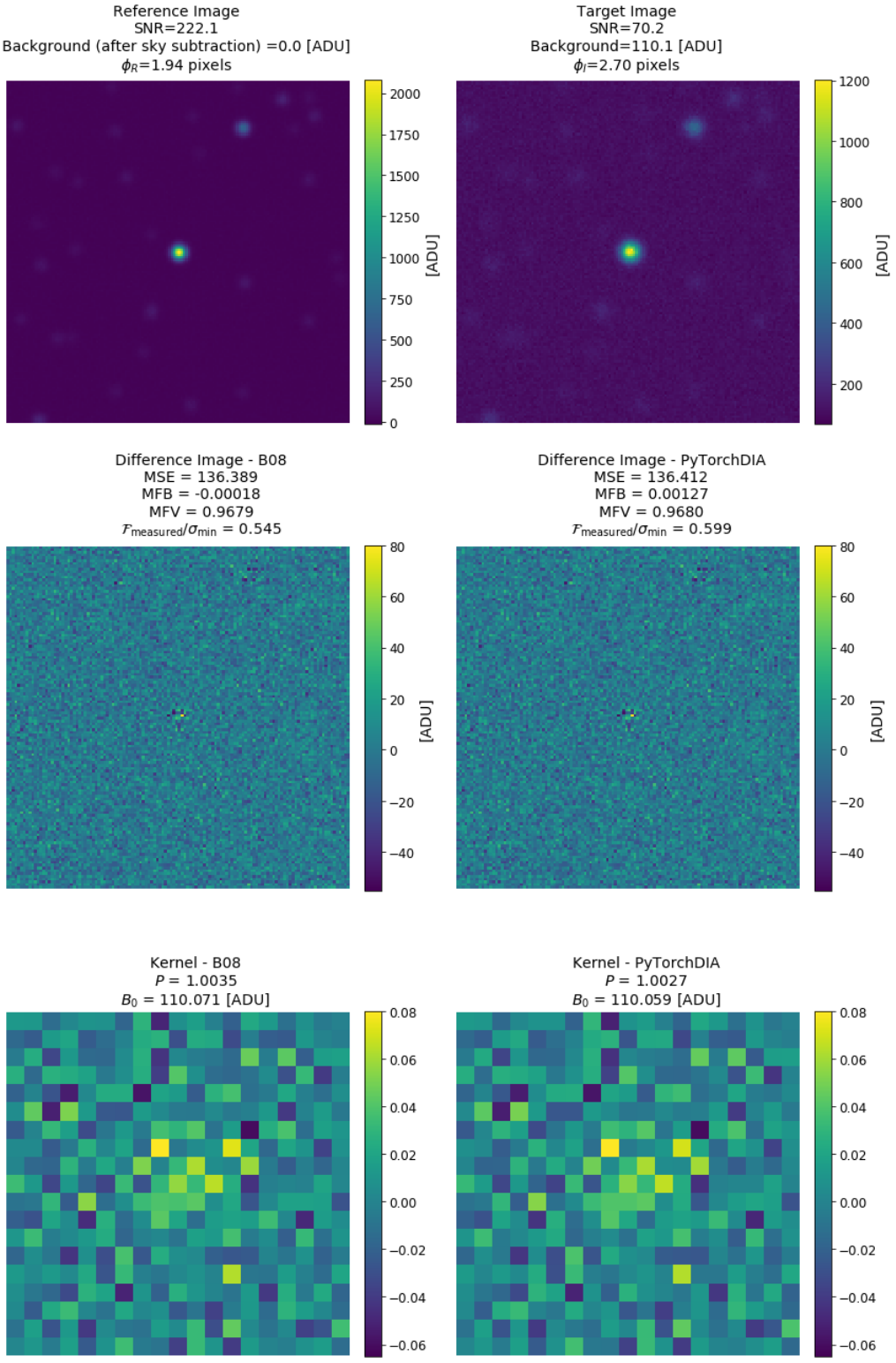


Figure 2.2: Example images and fit quality metrics from the simulated image experiments. (*Top row*) An example reference (left) and target (right) image pair generated in the simulation tests. (*Middle row*) The subsequent difference images and fit metrics recovered by the B08 approach (left) and our PyTorch implementation (right). (*Bottom row*) The corresponding convolution kernels and fit parameters for the B08 (left) and PyTorch (right) solutions. The median pixel value is subtracted from the reference image before fitting the kernel and differential background term, and so B_0 will not be equal to zero. The B08 and PyTorchDIA solutions are very similar.

categories by the sampling regime of the images: (i) $\phi_R > 1$ and $\phi_K > 1$, (ii) $\phi_R > 1$ and $\phi_K < 1$, (iii) $\phi_R < 1$ and $\phi_K > 1$, (iv) $\phi_R < 1$ and $\phi_K < 1$.

The distributions of the fit metrics are in general skewed, and so we report the median values as a robust estimate of the central value in Figure 2.3 for both the B08 and PyTorchDIA implementations. Each sub-plot pair in Figure 2.3 pertains to a single metric, with the B08 results in blue in the left column, and PyTorchDIA results in red in the right column. On the x-axes, we plot the SNR regime of the target image, categorised as above. There are 4 points in each SNR regime in each sub-plot, each of which corresponds to a different sampling regime, and we offset these from each other for clarity. We use circular markers to denote the sampling regime of the reference image, and crosses to indicate the sampling regime of the kernel. A big circle or cross corresponds to an over-sampled reference image or kernel respectively, and a small circle or cross corresponds to an under-sampled reference image or kernel. The green dashed lines for each sub-plot pair represent the 'best' value for each metric. We also tabulate these results in Table 2.2, and include the 16th and 84th percentiles about the median of the distributions. In the bottom section of this table, we note the number of simulations which fall into each SNR and sampling regime category.

Across all the metrics, one of the biggest differences between the B08 and PyTorchDIA solutions is seen in the values for the photometric scale factor. B08 is more accurate in general, but differences between the median values in each SNR and sampling regime are typically small (~ 0.001), and become negligible when the SNR is high. B16 showed that the accuracy of P is strongly determined by the SNR of the target image, and in Table 2.2, we also see that the distribution of P values about the median becomes tightly concentrated about unity as the SNR increases. Interestingly, for results in any given sampling regime, there is no clear similarity in trends between the two approaches with the target image SNR.

There are no large differences between B08 and PyTorchDIA in terms of MSE, with B08 better only at the level of the first decimal place. Looking at Table 2.2, B08 does however begin to slightly outperform PyTorchDIA in the highest SNR regimes when both the kernel and the reference frame are under-sampled.

As observed in B16, both approaches show overall gradually decreasing MFB as the SNR increases, although B08 shows a negative bias while PyTorchDIA is typically positively bias. PyTorchDIA again only appears to do noticeably worse than B08 in the highest SNR category

when both the reference image and the kernel are undersampled.

The MFV values returned by B08 and PyTorchDIA are also very similar. The B08 MFV values are usually lower than those for PyTorchDIA, although all median values are less than unity. These results too are consistent with B16, who also found the unregularised kernel was prone to overfitting when the SNR of the target image is low.

The differences between B08 and PyTorchDIA in terms of MPB are small, with both showing the same trends with SNR and sampling regimes. As found by B16, with increasing SNR, the MPB greatly improves. That this is typically best when the kernel is oversampled, and worst when both the kernel and reference image are undersampled, is likely in part due to sampling issues associated with the PSF model which is scaled to the flux in the difference image. Encouragingly, with reference to the values in Table 2.2, we can see that in general, PyTorchDIA performs even better than B08. That the MPB metrics are similar despite the differences in MFB is likely due to our choice to simultaneously fit for a constant scalar offset in our PSF fitting procedure, which would correct for any net over-/underestimation in the background parameter of the image model.

The same trends in the MPV values with both SNR and sampling regimes can again be seen by the two approaches. With an increasing SNR, the MPV increases. This behaviour was also seen by B16, and it can be explained as reduced overfitting of the bright central stars from which this metric is computed. For both approaches, apart from some cases when both the kernel and reference image are undersampled, the MPV is always less than unity, and similar results were obtained by B16 (see Figure 6 therein). We experimented by performing an additional 21271 tests in which the photometry of the faintest star in the image was used to compute the MPV. In this instance, no MPV values were less than 1.1.

While being overall very similar, the PyTorchDIA MPV values are slightly higher than their B08 counterparts. Because B08 overfits more strongly than PyTorchDIA (see MFV values), we should expect the noise in the B08 difference images to be slightly more suppressed. Consequently, the photometry will tyically have a lower variance. The difference in MPV is greatest when both the kernel and reference image are undersampled. This may be associated with the worse MFB values for PyTorchDIA in this category, as while the additive constant in our PSF fitting procedure will be able to correct for inaccuracies in the differential background, it will do so at the cost of slightly increased variance in the photometry.

In conclusion, PyTorchDIA performs very similarly to the B08 algorithm in these tests on synthetic images, and it is encouraging to see that the major conclusions drawn from these tests are consistent with those in B16. These images have Gaussian noise contributions, and have no outliers, and it is therefore not surprising that the B08 algorithm does slightly better overall, since the linear least-squares formulation (within an iterative scheme) correctly describes the observation model, and this approach operates at twice the floating-point precision of PyTorchDIA.

2.4.4 No 'algorithmic' bias

As PyTorchDIA is a numerical algorithm, it is essential to know to what precision it can recover the correct solution if random noise has been removed from the data. By default, PyTorchDIA operates at F32, giving us 6-8 decimal digits of precision.

Following Section 3.1 of B08, we perform the following tests (i) - (iv) where known, noiseless transformations are applied to a reference image to generate a target image. The 142×142 pixel reference image used in what follows is generated randomly, as outlined in Section 2.4.1. We compute the fractional error, ϵ, of P and B_0, as $\epsilon_P = |(P - P_{\rm True})/P_{\rm True}|$ and $\epsilon_{B_0} = |(B_0 - B_{0,\rm True})/B_{0,\rm True}|$[9].

(i) The simplest possible test we can perform is to difference an image against a copy of itself. The kernel should be exactly 1 at the centre and 0 everywhere else. Encouragingly, PyTorchDIA is able to return the correct answer to within the theoretical convergence tolerance, with fractional errors of $\epsilon_P = 4 \times 10^{-7}$ and $\epsilon_{B_0} = 1 \times 10^{-8}$. The central kernel pixel was correct to a precision of 7 decimal digits.

(ii) Next, we convolve the reference with a Gaussian kernel of $\phi_K = 1.5$ pixels. Once again, PyTorchDIA returns this Gaussian kernel to within its convergence tolerance, with $\epsilon_P = 3 \times 10^{-7}$ and $\epsilon_{B_0} = 1 \times 10^{-8}$.

(iii) The target image is created by shifting the reference image by one pixel in each of the positive x and y spatial directions, without resampling. The corresponding kernel should be the identity kernel (central pixel value of 1 and 0 elsewhere) shifted by one pixel in each of the negative l and m kernel coordinates. PyTorchDIA returns a kernel with a peak pixel value of unity precise to F32 machine precision, and $\epsilon_P = 4 \times 10^{-7}$ and $\epsilon_{B_0} = 1 \times 10^{-8}$.

[9] As always, we background subtract the reference frame, so $B_0 > 0$.

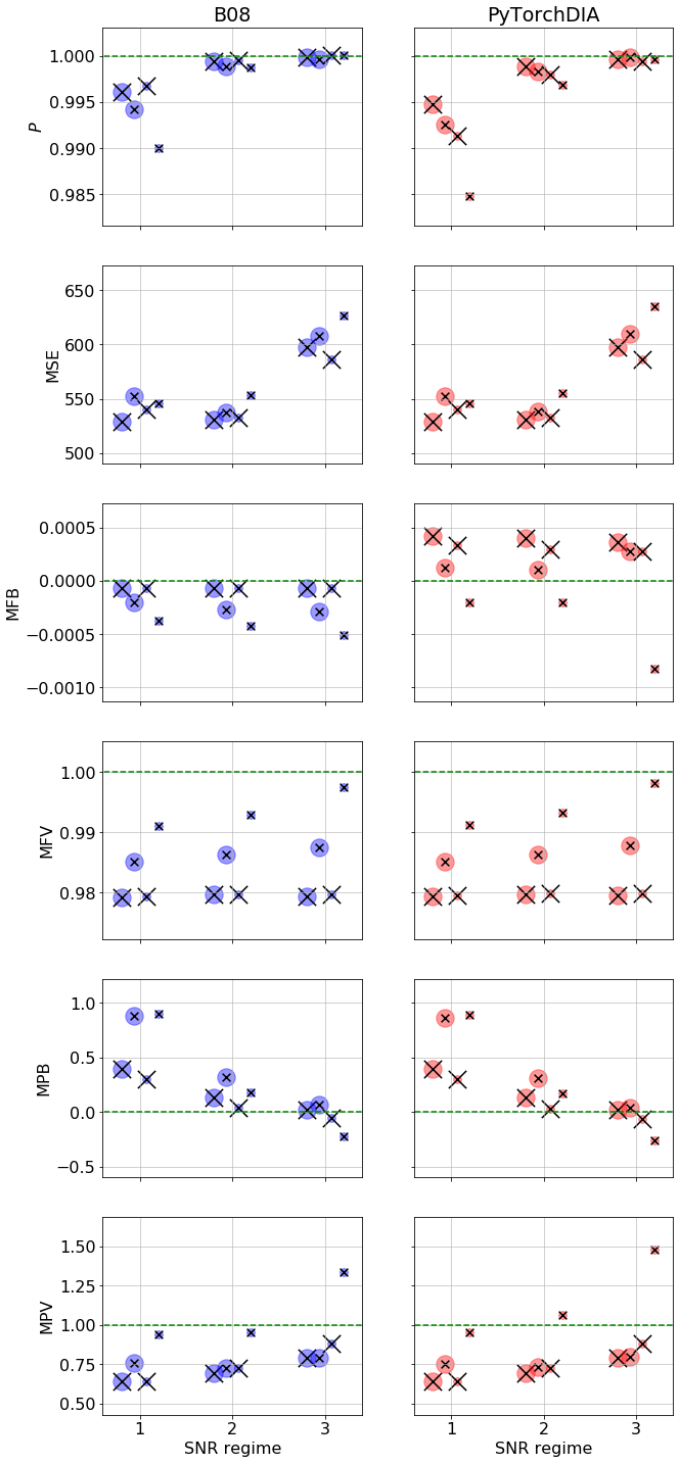


Figure 2.3: Fit quality and photometric accuracy metrics from the 71569 simulated image tests. Results for the B08 algorithm are in blue (left column), and the PyTorchDIA results are in red (right column). The signal-to-noise regime of the target image is shown on the x-axis, increasing from left to right. Metrics in each SNR regime are offset from each other for clarity. We use circular markers to denote the sampling regime of the reference image, and crosses to indicate the sampling regime of the kernel. A big circle or cross corresponds to an over-sampled reference image or kernel respectively, and a small circle or cross corresponds to an under-sampled reference image or kernel; there are therefore 4 possible combinations of marker for each SNR regime. The green dashed lines for each sub-plot pair represent the correct, 'ideal' value for each metric.

Metric	SNR regime	$\phi_R > 1, \phi_K > 1$	$\phi_R > 1, \phi_K < 1$	$\phi_R < 1, \phi_K > 1$	$\phi_R < 1, \phi_K < 1$
P (B08)	1	$0.9961^{+0.0335}_{-0.0358}$	$0.9942^{+0.0316}_{-0.0380}$	$0.9968^{+0.0273}_{-0.0323}$	$0.9900^{+0.0279}_{-0.0342}$
P (PyTorchDIA)	1	$0.9948^{+0.0335}_{-0.0360}$	$0.9926^{+0.0320}_{-0.0376}$	$0.9913^{+0.0277}_{-0.0328}$	$0.9848^{+0.0285}_{-0.0368}$
P (B08)	2	$0.9994^{+0.0125}_{-0.0125}$	$0.9989^{+0.0124}_{-0.0134}$	$0.9996^{+0.0101}_{-0.0108}$	$0.9987^{+0.0103}_{-0.0111}$
P (PyTorchDIA)	2	$0.9989^{+0.0124}_{-0.0126}$	$0.9983^{+0.0128}_{-0.0135}$	$0.9980^{+0.0098}_{-0.0112}$	$0.9969^{+0.0114}_{-0.0135}$
P (B08)	3	$0.9998^{+0.0045}_{-0.0046}$	$0.9997^{+0.0045}_{-0.0046}$	$1.0001^{+0.0037}_{-0.0040}$	$1.0001^{+0.0039}_{-0.0040}$
P (PyTorchDIA)	3	$0.9996^{+0.0045}_{-0.0045}$	$0.9999^{+0.0047}_{-0.0050}$	$0.9995^{+0.0038}_{-0.0040}$	$0.9996^{+0.0042}_{-0.0049}$
MSE (B08)	1	$528.53^{+328.79}_{-329.61}$	$552.62^{+310.51}_{-349.03}$	$539.85^{+318.94}_{-342.21}$	$545.34^{+313.22}_{-340.47}$
MSE (PyTorchDIA)	1	$528.54^{+328.80}_{-329.60}$	$552.63^{+310.51}_{-349.03}$	$539.86^{+318.95}_{-342.21}$	$545.42^{+314.59}_{-340.54}$
MSE (B08)	2	$530.88^{+343.43}_{-330.54}$	$537.47^{+337.46}_{-328.53}$	$532.74^{+342.01}_{-337.18}$	$553.47^{+336.57}_{-347.22}$
MSE (PyTorchDIA)	2	$530.90^{+343.43}_{-330.55}$	$537.92^{+337.16}_{-328.88}$	$532.82^{+341.94}_{-337.24}$	$555.51^{+337.07}_{-346.81}$
MSE (B08)	3	$597.17^{+352.67}_{-362.18}$	$608.09^{+361.21}_{-361.96}$	$585.85^{+365.45}_{-356.65}$	$626.40^{+415.64}_{-366.67}$
MSE (PyTorchDIA)	3	$597.33^{+352.64}_{-362.28}$	$610.09^{+361.05}_{-363.08}$	$585.93^{+365.41}_{-356.49}$	$634.81^{+415.59}_{-371.31}$
MFB (B08)	1	$-0.0001^{+0.0000}_{-0.0001}$	$-0.0002^{+0.0001}_{-0.0002}$	$-0.0001^{+0.0000}_{-0.0001}$	$-0.0004^{+0.0002}_{-0.0003}$
MFB (PyTorchDIA)	1	$0.0004^{+0.0004}_{-0.0003}$	$0.0001^{+0.0003}_{-0.0003}$	$0.0003^{+0.0004}_{-0.0004}$	$-0.0002^{+0.0005}_{-0.0006}$
MFB (B08)	2	$-0.0001^{+0.0000}_{-0.0001}$	$-0.0003^{+0.0001}_{-0.0002}$	$-0.0001^{+0.0000}_{-0.0001}$	$-0.0004^{+0.0002}_{-0.0004}$
MFB (PyTorchDIA)	2	$0.0004^{+0.0005}_{-0.0004}$	$0.0001^{+0.0009}_{-0.0008}$	$0.0003^{+0.0004}_{-0.0003}$	$-0.0002^{+0.0044}_{-0.0060}$
MFB (B08)	3	$-0.0001^{+0.0001}_{-0.0001}$	$-0.0003^{+0.0001}_{-0.0003}$	$-0.0001^{+0.0001}_{-0.0002}$	$-0.0005^{+0.0003}_{-0.0008}$
MFB (PyTorchDIA)	3	$0.0004^{+0.0007}_{-0.0007}$	$0.0003^{+0.0018}_{-0.0013}$	$0.0003^{+0.0005}_{-0.0004}$	$-0.0008^{+0.0025}_{-0.0044}$
MFV (B08)	1	$0.9792^{+0.0116}_{-0.0112}$	$0.9851^{+0.0116}_{-0.0115}$	$0.9794^{+0.0114}_{-0.0115}$	$0.9911^{+0.0115}_{-0.0127}$
MFV (PyTorchDIA)	1	$0.9793^{+0.0116}_{-0.0112}$	$0.9851^{+0.0117}_{-0.0115}$	$0.9794^{+0.0114}_{-0.0115}$	$0.9912^{+0.0116}_{-0.0127}$
MFV (B08)	2	$0.9796^{+0.0113}_{-0.0116}$	$0.9862^{+0.0121}_{-0.0123}$	$0.9797^{+0.0113}_{-0.0117}$	$0.9930^{+0.0143}_{-0.0133}$
MFV (PyTorchDIA)	2	$0.9797^{+0.0113}_{-0.0116}$	$0.9863^{+0.0123}_{-0.0123}$	$0.9798^{+0.0113}_{-0.0117}$	$0.9933^{+0.0147}_{-0.0134}$
MFV (B08)	3	$0.9794^{+0.0114}_{-0.0112}$	$0.9875^{+0.0122}_{-0.0117}$	$0.9797^{+0.0118}_{-0.0113}$	$0.9974^{+0.0304}_{-0.0156}$
MFV (PyTorchDIA)	3	$0.9796^{+0.0114}_{-0.0112}$	$0.9878^{+0.0128}_{-0.0118}$	$0.9799^{+0.0118}_{-0.0114}$	$0.9982^{+0.0384}_{-0.0161}$
MPB (B08)	1	0.401	0.880	0.300	0.900
MPB (PyTorchDIA)	1	0.400	0.862	0.300	0.887
MPB (B08)	2	0.138	0.324	0.038	0.184
MPB (PyTorchDIA)	2	0.136	0.313	0.036	0.176
MPB (B08)	3	0.027	0.074	−0.055	−0.219
MPB (PyTorchDIA)	3	0.023	0.046	−0.059	−0.257
MPV (B08)	1	0.640	0.760	0.640	0.937
MPV (PyTorchDIA)	1	0.639	0.754	0.640	0.955
MPV (B08)	2	0.692	0.728	0.725	0.951
MPV (PyTorchDIA)	2	0.693	0.728	0.726	1.066
MPV (B08)	3	0.789	0.788	0.880	1.339
MPV (PyTorchDIA)	3	0.789	0.794	0.883	1.478
	1	12332	4151	4078	1349
$N_{\text{Simulations}}$	**2**	13224	4447	4387	1526
	3	14673	4900	4829	1673

Table 2.2: Fit quality and photometric accuracy metrics for the 71569 simulated image tests for both an implementation of the B08 algorithm and PyTorchDIA. We divide the results of these tests by the SNR regime of the target image and the sampling regimes of the reference image and convolution kernel. The number of simulations used for the computation of the metrics in each of the 12 possible SNR and sampling regime categories is shown in the bottom section.

(iv) The reference image is shifted by half a pixel in each of the positive x and y spatial directions to create the target image, an operation that requires the resampling of the reference image. We use a bicubic spline resampling method[10]. PyTorchDIA performs very well, successfully returning the complicated kernel, with fractional errors on P and B_0 of $\epsilon_P = 4 \times 10^{-7}$ and $\epsilon_{B_0} = 1 \times 10^{-8}$.

As PyTorchDIA is able to recover the true fit parameters correct to the data type's decimal digit precision, we conclude that there is no bias associated with our numerical algorithm.

2.5 Real Image Tests

DIA is a particularly effective tool for measuring the flux and positions of variable sources in crowded fields (see the start of Section 2.2 for some examples). The MiNDSTEp consortium performs follow-up observations of microlensed targets towards the galactic bulge with the high frame-rate EMCCD cameras on the Danish 1.54m telescope (DK154), at ESO La Silla (Skottfelt et al., 2015). Short 0.1 second exposures are shift-and-stacked to eliminate tip tilt distortions, and recover scenes with $\sim 2-3$ times better resolution than conventional long integrations. The combined effects of the mount (the DK154 mirror rests on three points) and the shift-and-add procedure produces irregular, triangular PSFs, with a sharp peak and diffuse halo. With a scale of 0.09 arcseconds per pixel, this high resolution EMCCD instrument explores a subset of the sampling regimes covered in the CCD simulations in Section 2.4.

2.5.1 Data and reductions

We use a sample of 251 512 × 512 pixel images, each consisting of up to 3000 shift-and-stacked 0.1 s short exposures[11] (i.e. each stack is equivalent to at most a 5 minute exposure), obtained with the 'red' camera (approximately equivalent to a broad SDSS 'z' filter Fukugita et al. 1996) on the Danish 1.54m Lucky Imaging instrument between 2019-07-17 and 2019-07-30. The observations are of the centre of the OGLE-III-BLG101 microlensing field, $(l, b) = (0.1331°, -1.9643°)$. With a pixel scale of 0.09 arcseconds per pixel, the camera covers a field of view of 45 × 45 arcseconds^2. All images are bias subtracted and flat corrected, and the master flats associated with each night of observing are used in the noise model for the following DIA performance tests (Equation 2.7).

[10] Specifically, we use the function scipy.ndimage.interpolation.shift, with no pre-filtering.

[11] During the shift-and-stack procedure, if a shift is above a certain threshold the frame is rejected, and so some stack consist of slightly less than 3000 images.

To create a high signal-to-noise reference, 13 sharp images acquired on 2019-07-20 were registered to the same pixel grid using bicubic spline resampling. The stacked reference frame was constructed by then summing the registered images, and dividing by 13. To measure the PSF of this stacked image, a (symmetric) Gaussian PSF model was independently fit to 5 bright, isolated stars, from which we found a median width of $\phi_R = 2.89$ pixels. The median ϕ_I of the 238 remaining target images was 3.83 pixels, with a maximum of 6.67 pixels.

In preparation for assessing the photometric accuracy of the algorithms, we measured the fluxes and positions of the stars in the reference image. A total of 236 candidate sources were detected in the image. We avoided all stars near the reference image edges and those with a peak pixel flux less than 3×10^4 ADU. This gave us a reduced sample size of the 37 brightest stars. Due to processing time constraints (see below), this sample was further reduced to 30, approximately uniformly spaced stars. The light curves of these 30 stars can be used to compute the MPB and MPV metrics (see Section 2.5.2).

For both the reference and each of the 238 target images, a 142×142 pixel region was cutout around the positions of each of the 30 stars. This approach avoids resampling the target images to align them with the reference, and prevents introducing correlated noise into the target images by interpolating between pixels.[12]

Cursory image model fits to these target image stamps identified kernels with a size of $7\phi_I \times 7\phi_I$ to give good results. For the tests in Section 2.4, a $9\phi_I \times 9\phi_I$ kernel corresponded to a 19×19 pixel array. For these tests on real images, even though the kernel is set to be just $7\phi_I \times 7\phi_I$ large, due to the fine sampling of the LI images, the median size of the kernel in pixel-space for this data set is 27×27 pixels, with a maximum size of 47×47 pixels. For the B08 algorithm in particular – which scales with the square of the number of N_K kernel pixels – processing times are much more expensive than in Section 2.4, and this is why we further reduced our sample of bright stars down from 37 to 30 to keep this investigation tractable.

This should give us a total of 7140 reference-target image pairs (i.e. $\sim 10\%$ the size of the simulated image data set), but in some instances, for stars close to the borders of the target image, large shifts between this image and the reference meant that part of the 142×142 pixel region fell outside the image edge. Additionally, the measured signal-to-noise ratio of the target image in some instances exceeded 1000, and so these were rejected to provide a

[12] We note that the noise in all shift-and-stacked images will be correlated to some degree due to the shift procedure.

more consistent comparison with the results from Section 2.4. After these cuts, we had a final sample of 6989 reference-target image pair stamps.

2.5.2 Model performance metrics for real data

For this test, we compare the PyTorchDIA code implementing our robust loss function with $c = 1.345$ (Equation 2.13), against the B08 solution making use of sigma-clipping. We run the B08 approach for a total of 3 iterations, and clip $|\epsilon_{ij}| > 5$ pixels on the third iteration only.

Although we are both limited to a smaller sample size and do not know the true noise properties or model parameters of the actual images and kernel solutions respectively, we can still use almost all of the metrics outlined in Section 2.4.2 to assess the accuracy of the implementations. As the true model image is unknown, we are unable to use the MSE metric to measure the model error. The photometric scale factor, P, while also unknown, can however be compared on a relative scale, which requires an estimate for P_{true}.

It was found that the inferred P was correlated with the distance from the image centre along the x-axis. This is due to a change in the sky level along this axis associated with instrumental effects, which influences P due to the anti-correlation between the photometric scale factor and B_0. Consequently, P_{true} for the entire image is not well represented by a single number (e.g. the median value from all 30 stamps). Given this, we divided the x-axis into 4 equally spaced regions, and computed 4 separate P_{true} values using the subset of stamps in each of these region. The measured P from any stamp was then normalised by the P_{true} corresponding to the region it belonged to along the chip's x-axis. As B08 outperformed PyTorchDIA in terms of P in Section 2.4, we use the P values inferred by this approach to determine P_{true}.

The remaining metrics require an accurate noise model to be meaningful. For these EMCCD images, we use Equation 2.7, and substitute the M_{ij} obtained on the third and final iteration of the B08 solution of each reference-target image pair to estimate the pixel uncertainties. These pixel uncertainties can then be used to calculate the MFB and MFV metrics, in place of $\sigma_{I,ij}$. Similarly, by substituting the target image for $I_{\text{noiseless},ij}$ in Equation 2.27 we can use these pixel uncertainties to calculate the signal-to-noise ratio of the target image for each reference-target image pair. For the third iteration of the B08 approach, we also sigma-clip $|\epsilon_{ij}| > 5$ pixels to guard against variable sources or spurious pixels affecting the kernel solution. We use this associated bad pixel mask to omit these pixels from the calculation of the corresponding MFB

and MFV metrics for both the B08 and PyTorchDIA solutions.

We perform PSF fitting photometry at the centre of the difference images obtained from each reference-target image pair (i.e. at the position of one of the 30 possible stars) to assess the photometric accuracy of the approach. We infer an empirical model PSF for the target image in the following way. For each reference-target image pair, we identify the peak pixel values of bright sources in the reference, and set all other pixel values to 0. We then use our PyTorchDIA implementation to infer the 'PSF' which convolves this scene of (approximate) delta-functions to the target image. That is, we are inferring an estimate of the PSF of the target image with the 'kernel' solution of the PyTorchDIA code. When inferring this empirical PSF, we weight all pixels equally, which gives the pixel values of the very brightest stars more weight relative to the other pixels in the image.

As in Section 2.4.2, we use a small square stamp for the PSF fitting. We set the radius of this stamp (i.e. the half-width of the stamp) to 1.5 FWHMs of the target image (i.e. $\sim 1.5 \times 2.355 \times \phi_I$ pixels), rounded up to the nearest odd integer pixel. The PSF object is normalised, and scaled to the flux of the difference image. To guard against the influence of (possibly variable) neighbouring sources or spurious pixels in the stamp not captured by the PSF model, we scale the model to a star's differential flux under our robust loss function (Equation 2.12), with fixed pixel uncertainties and $c = 1.345$.

As with the MFB and MFV metrics, we adopt Equation 2.7 as the noise model, substituting the M_{ij} values obtained on the final iteration of the B08 solution. In order to calculate σ^2_{min} (Equation 2.31), we substitute the normalised empirical PSF inferred by the PyTorchDIA code for $\mathscr{P}_{I,rs}$, and substitute the region-dependent P_{true} inferred by the B08 implementation. An example reference-target image pair and associated fit quality metrics for both the B08 and PyTorchDIA implementations are shown in Figure 2.4.

2.5.3 Real Image Test results

We again start our analysis by splitting our results into three signal-to-noise ratio (SNR) regimes by the signal-to-noise of the target image, I: (1) $8 < \text{SNR}_I < 40$; (2) $40 < \text{SNR}_I < 200$ and (3) $200 < \text{SNR}_I < 1000$. We found that no reference-target image pairs correspond to the very lowest SNR regime, and so we are forced to restrict our analysis to regimes (2) and (3). Also, unlike the simulated image tests, all 251 real images used here are well sampled. A majority

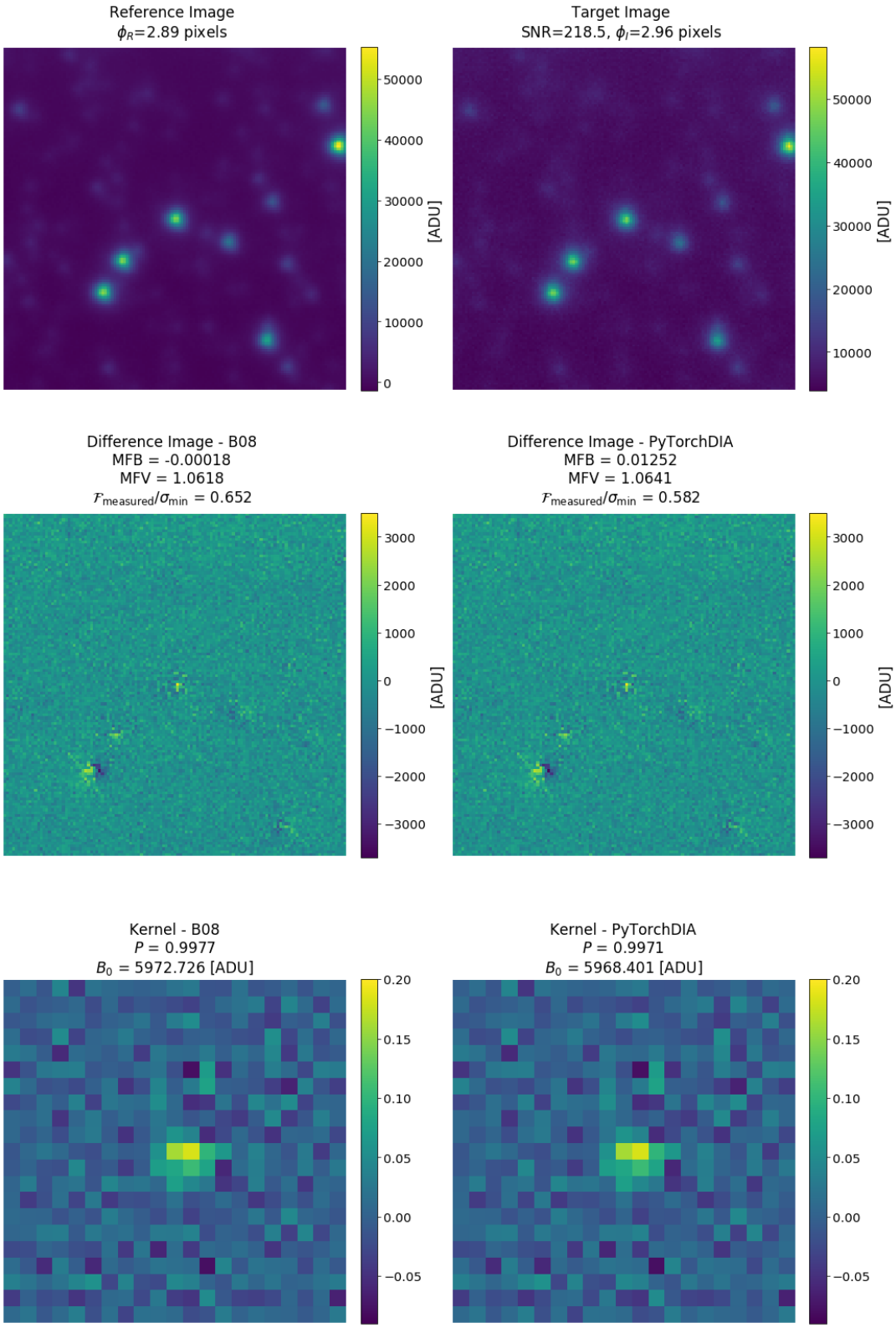


Figure 2.4: Example reference-target image pair from the real image performance tests, in the same style as Figure 2.2. Note the unusual PSFs in the reference and target images, and the correspondingly irregular kernels. The target star with which the photometric accuracy was assessed is at the centre of the reference-target image pair.

195 of the target images correspond to the case where the kernel is oversampled (i.e. $\phi_K > 1$), and for the other images the kernel is undersampled (i.e. $\phi_K < 1$). We therefore further divide each signal-to-noise regime into two sampling regimes, dependent on whether the kernel is approximately over or undersampled. We plot the median metric values and the MPB and MPV for both B08 (blue) and PyTorchDIA (red), each split into the 4 possible sub-categories in Figure 2.5. As all images are oversampled, the circular markers are all large. The small and large crosses correspond to under- and oversampled kernels. The theoretical best value for each metric is plotted as a dashed green line. The median metrics and the 16th and 84th percentiles of their distributions are tabulated in Table 2.3.

It was noted that some star differential light curves could have a noticeable non-zero offset from their reference frame flux level. In order to correct for this before grouping the normalised photometric residuals from across all light curves into each of the SNR and sampling regime categories, each star light curve was shifted such that its median photometric residual value was 0. Additionally, there was a small subset of outlying photometric residuals that may badly affect the MPB and MPV metrics. To remove the influence of these bad outliers in each category, we first calculate the median absolute deviation of the $\mathscr{F}_{\text{measured},k}/\sigma_{\text{min},k}$ values scaled to a standard deviation σ_{MAD} as a robust estimate of the spread of the underlying distribution. This is defined as

$$\sigma_{\text{MAD}} = \kappa \times \text{median} \left| \frac{\mathscr{F}_{\text{measured},k}}{\sigma_{\text{min},k}} - \text{median}\left(\frac{\mathscr{F}_{\text{measured}}}{\sigma_{\text{min}}} \right) \right| , \tag{2.34}$$

where $\kappa = 1.4826$ for approximately normally distributed data. Before calculating the MPB and MPV metrics shown in Figure 2.5 and Table 2.3, we have removed all photometric residuals with absolute values more than $4.5\sigma_{\text{MAD}}$ from median$(\mathscr{F}_{\text{measured}}/\sigma_{\text{min}})$ for each SNR and sampling regime category. Note that the number of outlying points may differ for the B08 and PyTorchDIA solutions.

We tabulate the number of reference-target image pairs in each category in Table 2.3. Due to the sigma-clipping used before computing the MPB and MPV, we write the number of image pairs in each category used to compute these particular metrics as a fraction of the total number of image pairs. All image pairs in each category were used to compute all the other metrics, as these are robust to outliers.

We can again see that the PyTorchDIA and B08 approaches are broadly consistent, with

both DIA implementations returning very similar metric values. Across all SNR and sampling regimes, only the median MFB values show consistently large differences. PyTorchDIA exhibited notably larger MFB values in the tests on simulated images, and here too, the B08 approach performs better with respect to fit bias. The P values for B08 appear to be superior, but recall that these are normalised to the median P values obtained from each x-axis subregion by the B08 algorithm, so we should expect the B08 P values to be closer to unity. For both implementations, as seen in the tests in Section 2.4 and similar experiments in B16, the accuracy with which P can be determined clearly improves with increasing SNR.

Similar trends with both MFV and MPB are seen from both approaches. There is a relative paucity of results corresponding to an undersampled kernel – particularly for SNR category 2 – but for the richly populated categories corresponding to oversampled kernels, both of these metrics improve with increasing SNR, as also observed in the simulated tests.

The MPV metrics for both approaches are noticeably larger here than in the simulations. However, we again see that the photometric variance goes up with the SNR of the target image, and that both approaches are overall very similar. Unlike the simulations, there is no guarantee that the 30 selected stars are the brightest in their respective stamps, and each stamp is well populated with stars in this crowded field (see top panel of Figure 2.4). We should not then be overfitting the target stars, and so MPVs greater than unity are expected, since the fractional contribution of flux of the target to the entire stamp will, in general, be much less than unity (see e.g. Figure 10 of B16). Also, of the 43 images corresponding to an undersampled kernel, 23 of these images have a PSF sharper than the stacked reference. In these instances, the model image will inevitably fail to match the PSFs – as it would in fact have to de-convolve, rather than convolve the reference – which will result in bad artefacts at the position of sources in the difference image. Indeed, in each SNR regime, the MPV corresponding to the undersampled kernel is much worse than its oversampled counterpart. However, these effects are common to both approaches, and here, we stress that we simply want to highlight the similarity of the MPV metrics obtained by the B08 and PyTorchDIA implementations. Indeed, the PyTorchDIA metrics can only be compared in a relative sense to the B08 results, as the B08 solutions for P and σ_{ij} are used to compute these metrics for both approaches. As we do not know the ground-truth for these values, we cannot say whether B08 or PyTorchDIA is closest to the correct answer, but only how similar they are to each other.

In addition to the differences in floating-point precision that would have influenced the small differences in results in Section 2.4, the two approaches now assume two different noise distributions for the target image pixels. For this real, outlier populated data, PyTorchDIA optimises a robust scalar objective function, while the B08 approach uses an iterative sigma-clipping procedure, so we should expect this to contribute to differences between the metric values also. And of course, the sample sizes in each category are smaller than their simulated data counterparts (particularly so for the instance of $\phi_K < 1$ in SNR category 2) and therefore noisier, which could partly explain some of the small differences between the metric values for the two approaches.

In summary, similar trends in metrics with SNR and sampling regime categories are shown by both the B08 and PyTorchDIA implementations, and these trends resemble those found in our experiments on simulated images. This both validates that our simulated images are reasonable approximations of real data, and that our robust PyTorchDIA solution provides competitive photometric performance with the B08 algorithm when applied to real astronomical data sets.

2.6 Speed tests

In the speed tests in this Section, we compare the performance of our GPU-accelerated numerical implementation against a fast, analytical least-squares fit solution using compiled Cython code. The PyTorchDIA code is run on Google's Colab[13] service, a free-to-use cloud-based computation environment. The instance used for these experiments was equipped with a NVIDIA Tesla K80 GPU, with 2496 CUDA cores (with 'compute capability' 3.7), with up to 16280 Mb of GPU RAM free. The Cythonised B08 solution is run on an Intel(R) Xeon(R) CPU @ 2.30GHz, with $\sim$ 12.6 Gb RAM available.

Having established the accuracy of our algorithm on both synthetic and real images in Sections 2.4 and 2.5, we first provide a motivating test exploring the computational speed-up over the state of the art classical approach on a set of real DK154 microlensing observations in Section 2.6.1. We then use a pair of large synthetic images to formally explore the scaling of our algorithm with image and kernel size in Sections 2.6.2 and 2.6.3.

[13] `https://colab.research.google.com`

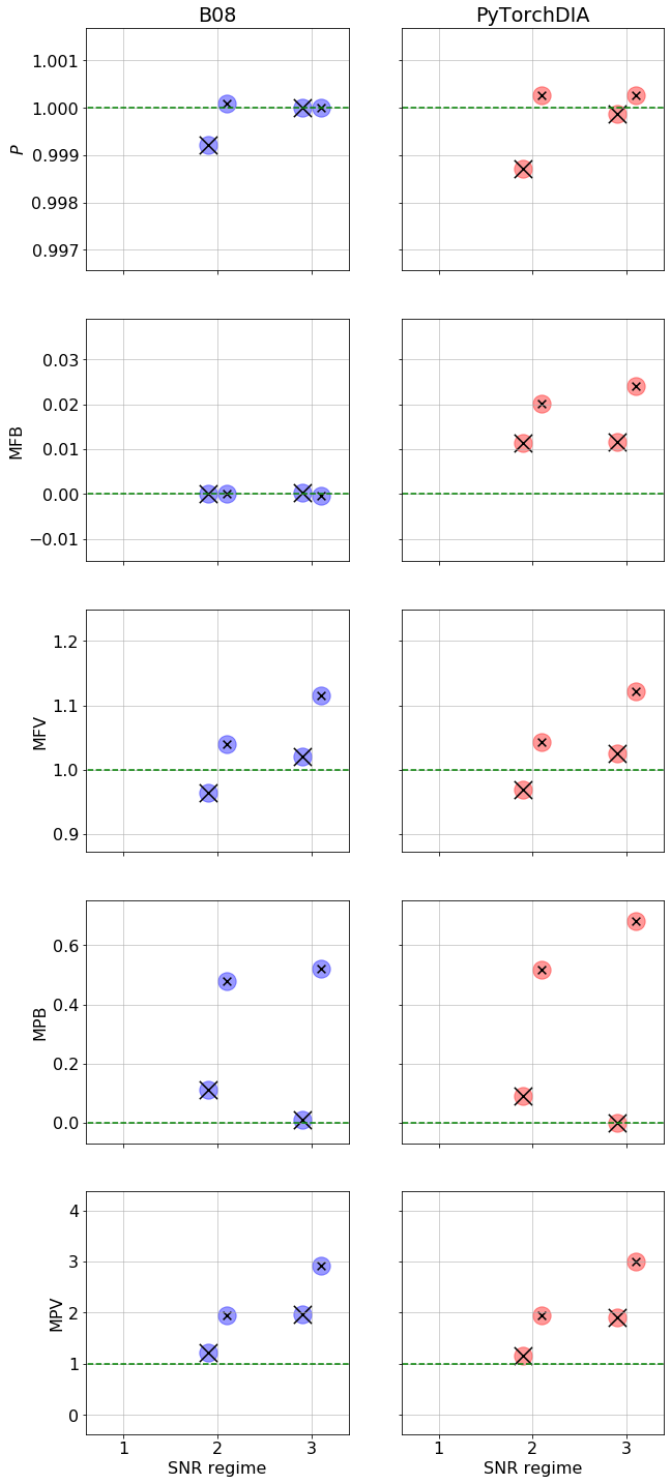


Figure 2.5: Fit quality and photometric accuracy metrics from the 6989 real image tests. Results for the B08 algorithm are in blue (left column), and the PyTorchDIA results are in red (right column). The signal-to-noise regime of the target image is shown on the x-axis, increasing from left to right. We use circular markers to denote the sampling regime of the reference image, and crosses to indicate the sampling regime of the kernel. As the reference image is oversampled, all circular markers are large. A large cross corresponds to an oversampled kernel, and a small cross corresponds to an under-sampled kernel; there are therefore 2 possible combinations of marker for each SNR regime.No reference-target image pairs used to compute these metrics fell into the lowest SNR regime, and so that is left blank. The green dashed lines for each sub-plot pair represent the correct, 'ideal' value for each metric.

Metric	**SNR regime**	$\phi_R > 1, \phi_K > 1$	$\phi_R > 1, \phi_K < 1$
P (B08)	1	-	-
P (PyTorchDIA)	1	-	-
P (B08)	2	$0.9992^{+0.0136}_{-0.0122}$	$1.0000^{+0.0070}_{-0.0036}$
P (PyTorchDIA)	2	$0.9987^{+0.0134}_{-0.0121}$	$1.0004^{+0.0065}_{-0.0048}$
P (B08)	3	$1.0000^{+0.0086}_{-0.0063}$	$1.0000^{+0.0050}_{-0.0048}$
P (PyTorchDIA)	3	$0.9999^{+0.0083}_{-0.0066}$	$1.0003^{+0.0049}_{-0.0052}$
MFB (B08)	1	-	-
MFB (PyTorchDIA)	1	-	-
MFB (B08)	2	$0.0001^{+0.0002}_{-0.0001}$	$0.0000^{+0.0002}_{-0.0002}$
MFB (PyTorchDIA)	2	$0.0114^{+0.0038}_{-0.0036}$	$0.0198^{+0.0082}_{-0.0054}$
MFB (B08)	3	$0.0003^{+0.0005}_{-0.0004}$	$-0.0003^{+0.0007}_{-0.0009}$
MFB (PyTorchDIA)	3	$0.0116^{+0.0045}_{-0.0044}$	$0.0244^{+0.0098}_{-0.0067}$
MFV (B08)	1	-	-
MFV (PyTorchDIA)	1	-	-
MFV (B08)	2	$0.9649^{+0.0480}_{-0.0431}$	$1.0392^{+0.0328}_{-0.0204}$
MFV (PyTorchDIA)	2	$0.9699^{+0.0477}_{-0.0402}$	$1.0416^{+0.0331}_{-0.0209}$
MFV (B08)	3	$1.0200^{+0.0809}_{-0.0475}$	$1.1168^{+0.0901}_{-0.0561}$
MFV (PyTorchDIA)	3	$1.0253^{+0.0827}_{-0.0473}$	$1.1246^{+0.1212}_{-0.0602}$
MPB (B08)	1	-	-
MPB (PyTorchDIA)	1	-	-
MPB (B08)	2	0.112	0.415
MPB (PyTorchDIA)	2	0.090	0.456
MPB (B08)	3	0.008	0.530
MPB (PyTorchDIA)	3	−0.001	0.703
MPV (B08)	1	-	-
MPV (PyTorchDIA)	1	-	-
MPV (B08)	2	1.220	1.902
MPV (PyTorchDIA)	2	1.165	1.892
MPV (B08)	3	1.979	2.851
MPV (PyTorchDIA)	3	1.929	2.945
N_{Stamps} (B08)	**1**	-	-
N_{Stamps} (PyTorchDIA)	**1**	-	-
N_{Stamps} (B08)	**2**	1981/1992	57/57
N_{Stamps} (PyTorchDIA)	**2**	1983/1992	57/57
N_{Stamps} (B08)	**3**	4060/4097	843/843
N_{Stamps} (PyTorchDIA)	**3**	4056/4097	842/843

Table 2.3: Fit quality and photometric accuracy metrics for the 6989 real image tests for both an implementation of the B08 algorithm and PyTorchDIA, separated into the SNR and sampling regimes. The number of image stamps used to compute the metrics in each of the SNR and sampling regime categories is shown in the bottom section of the table. As outlying photometric residuals have been removed from some categories prior to computing the MPB and MPV metrics, we write the number of N_{Stamps} values used in their computation as a fraction of the total number of stamp samples.

2.6.1 Real EMCCD images

For these tests, we use a typical microlensing data set obtained with the DK154's 'red' camera during the 2019 MiNDSTEp observing season. The data set consists of 159 shift-and-added exposures, each consisting of as many as 1200 short 0.1 s exposures (i.e. ~ 2 minute integrations).

The sharpest image was chosen as the reference, with a $\phi_R = 1.82$ pixels. The median target image width was $\phi_I = 3.18$ pixels. All images were bias subtracted and flat corrected. The target images were then re-sampled using bicubic spline interpolation to align them with the reference.

We use the flat field used for the data reduction in our noise model (Equation 2.7). For the B08 implementation, we perform 3 model iterations and employ sigma-clipping, masking pixels associated with $|\epsilon_{ij}| > 5$, to guard against variable sources (e.g. the gravitationally lensed target). For our PyTorchDIA implementation, we minimise the negative log-likelihood corresponding to our robust loss function (Equation 2.13).

We test how the speed of our implementation scales with both image and kernel size by symmetrically cropping the borders of the images to different extents, and solving for both a 19×19 and 25×25 kernel. We measure the solutions times for each of the 158-large set of target images for our GPU-accelerated numerical algorithm, and plot the median solution time for a single image against the single axis length of the cropped square images (as a proxy for image size) in Figure 2.6, for two different choices of parameter initialisation. The 'error' bars on the median image solution times are the 16th and 84th percentiles of the distribution of the 158 target image solution times. For comparison, we plot the time taken for 3 model iterations of the B08 analytical least-squares approach. We do not plot uncertainties for the B08 solution times. The total time of this analytic approach is dominated by the normal matrix construction, which is dependent only on the size of the kernel and images, and therefore effectively consistent across all 158 target images for each image and kernel size combination.

Unsurprisingly, the massive parallelisation inherent to the convolution computations in our implementation means it performs very well for even quite large image and kernel size combinations (see Section 2.6.2 for a formal discussion of how the algorithm scales.). The median solution times for both the 19×19 and 25×25 kernels are at least an order of magnitude faster

than for the B08 approach. When scaled to data sets consisting of hundreds or thousands of images, this clearly provides substantial savings in processing times.

The two plots in Figure 2.6 only differ in the choice of initialisation of the kernel pixels for PyTorchDIA. The plot on the top shows the solution times for a kernel initialised as a 'flat', box-car which sums to 1. The bottom plot shows the results for kernels initialised as symmetric Gaussians, normalised to sum to 1, with shape parameter estimated as $\phi_K = \sqrt{\phi_I^2 - \phi_R^2}$. For these stacks of short EMCCD exposures, neither ϕ_I nor ϕ_R are particularly well approximated with a Gaussian, and the median solution times are at most only ~ 1.1 times faster. However, the spread of the distribution of solution times is clearly tighter when a Gaussian in used for the initialisation compared to the flat box-car.

To gain some insight into the cause of the differences between the spread of values for the two different kernel initialisations (and therefore inform us of how to get the best performance out of PyTorchDIA) we plotted the solution times against the PSF width, ϕ_I, and signal-to-noise ratio (SNR) of the target image for the set of 482×482 images fit with a 19×19 kernel as a pairplot in Figure 2.7. We quantified the correlation between solution times and these two image properties with the Spearman's rank correlation coefficient, ρ[14]. Both kernel initialisations result in solution times which are anticorrelated with ϕ_I ($\rho_{\phi_I,\text{Box-car}} = -0.69$ and $\rho_{\phi_I,\text{Gaussian}} = -0.29$) and correlated with the signal-to-noise ratio ($\rho_{\text{SNR,Box-car}} = 0.63$ and $\rho_{\text{SNR,Gaussian}} = 0.26$). We expect however that the cause of these trends is due only to ϕ_I and specifically, the *difference* between the width of the PSFs in the target image and reference image. That the solution time is correlated with the signal-to-noise of the target image is due to the fact that the signal-to-noise is strongly anticorrelated with ϕ_I.

The very strong anti-correlation between the solution time for the box-car kernel and ϕ_I is very likely due to this 'flat' kernel surface being a much better initialisation for wide kernels. When the PSFs between the reference and the target images are very similar, the kernel is sharply peaked at the centre, and a flat kernel is far from the solution. Initialising with a Gaussian, although imperfect for the irregular PSFs associated with these images, appears to mitigate this problem to some degree, although the solution time still blows-up when the kernel is approximately under-sampled.

[14]This provides a measure of *any* monotonic relation between two variables. The distributions (vs solution time) are clearly not linearly related, and so Pearson's correlation coefficient – which assumes a linear relation – is not suitable here.

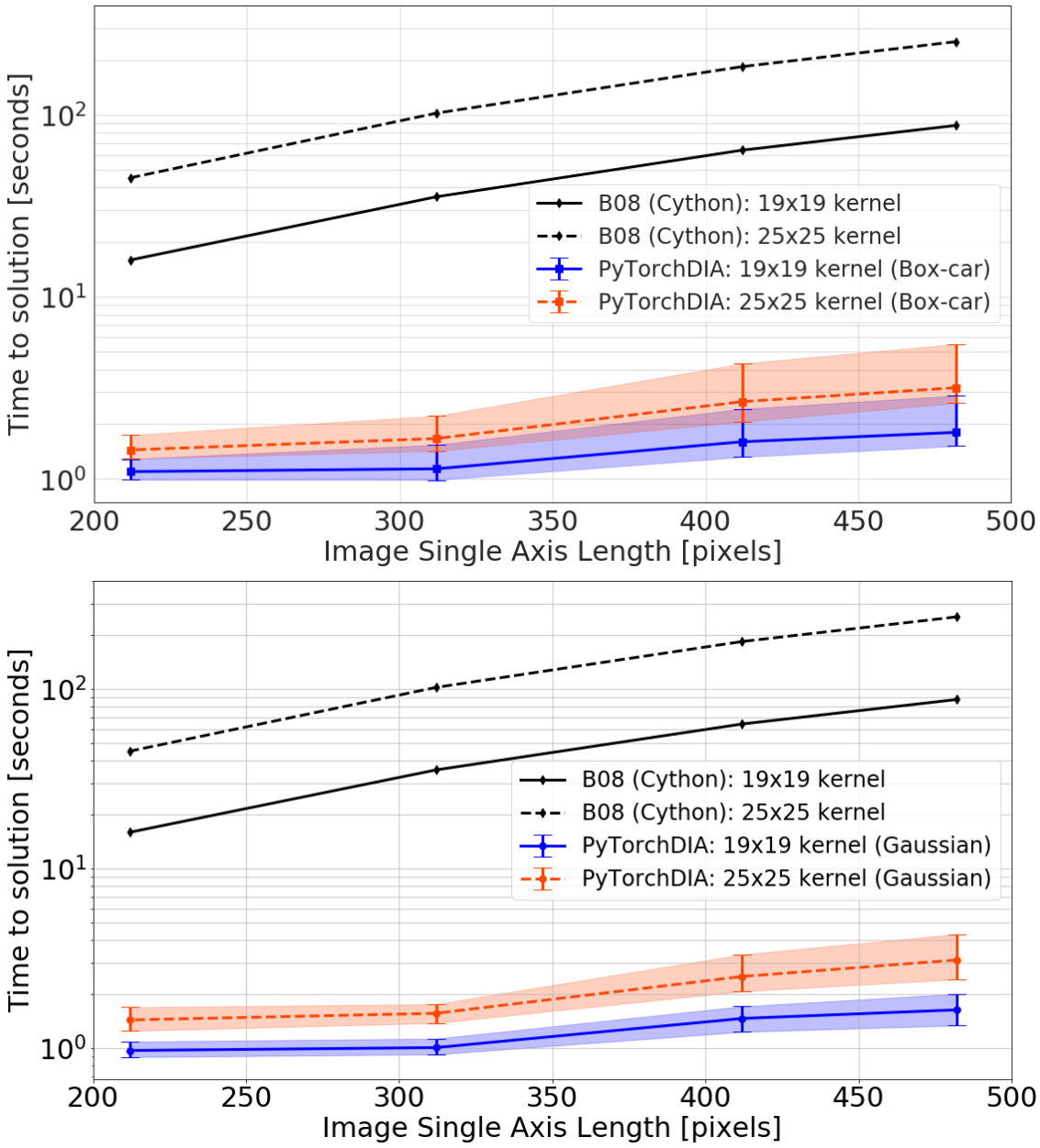


Figure 2.6: The time taken to solve for square kernels of different sizes against image single axis length (as a proxy for image size) for our PyTorchDIA implementation on the GPU and the B08 approach for images in a typical DK154 microlensing data set. The PyTorchDIA kernel pixels were initialised with a (top plot) 'flat', box-car kernel and (bottom plot) a symmetric Gaussian with width estimated as $\phi_K = \sqrt{\phi_I^2 - \phi_R^2}$.

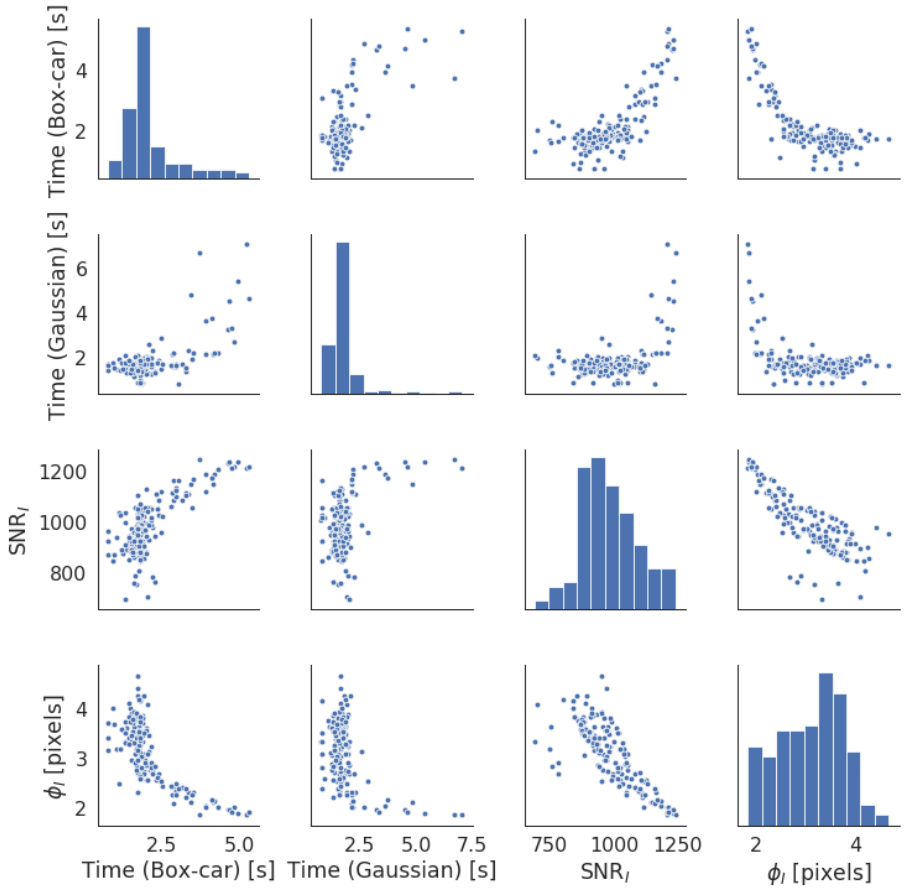


Figure 2.7: Pair plot showing the optimisation solution times for a 19×19 kernel – initialised as either a flat box-car or Gaussian – on a 482×482 large image, against the ϕ_I and SNR of the target image. The histograms of the distributions are shown on the diagonal.

2.6.2 Synthetic CCD images

Here, we test the performance of our implementation on synthetic images. We use our synthetic image generation procedure (see Section 2.4.1) to first generate a large 4000 × 4000 pixel noiseless reference image. We set the logarithm of the stellar density per 100 × 100 pixels to be 1.5, we set $\phi_R = 1$ and the sky level at 100 ADU. Fluxes are assigned randomly as described in Section 2.4.1. We set $\phi_K = 1.5$, and generate a target image with $\phi_I = \sqrt{\phi_R^2 + \phi_K^2}$ in the same manner as above. We then randomly add noise to the images in way described in Section 2.4.1, such that the target image has 10 times more pixel variance than the reference.

As there are no outlying pixels in this simulated pair of images, for the PyTorchDIA implementation, we minimise the Gaussian negative log-likelihood (Equation 2.4). The pair of 4000 × 4000 images are symmetrically cropped to assess how the solution times scale with image size. For each kernel and image size combination (and choice of kernel initialisation), we plot the solution times in Figure 2.8, for kernels initialised with either a flat box-car, or symmetrical Gaussian with width equal to $\phi_K = 1.5$.

As with the speed results in the prior Section, for larger images, the PyTorchDIA 19 × 19 and 25 × 25 kernel solution times are at least an order of magnitude faster than their B08 counterparts. In this case, since the correct convolution kernel really is a Gaussian, initialising the kernel as a Gaussian substantially reduces the number of optimisation steps required before convergence, with these solution times typically being $\sim 4-5$ faster than with a box-car initialisation for the kernel. Formally, the B08 normal matrix construction time scales as $\mathcal{O}(n^2m^4)$, where n and m are the sizes of the (square) images and kernel respectively. The two black curves in both Figures 2.6 and 2.8 follow this scaling (i.e. $\sim (25/19)^4$). Our optimisation procedure computes a direct convolution, which is predicted to scale as $\mathcal{O}(n^2m^2)$. The data points corresponding to the PyTorchDIA solution times in Figure 2.8 approximately obey this rule – in both the separation of the two curves and the separation between data points on each curve – although we should expect some variation due to differences in the number of optimisation steps required before converging and, potentially, cuDNN's choice of algorithm for the convolution computations, which can depend on both kernel and image size (see Section 2.6.3). In general, we found the number of optimisation steps required for convergence to slightly decrease with increasing image size. This makes sense, in that the effective information content of the image is greater the larger it is, and so the gradient steps are more accurate.

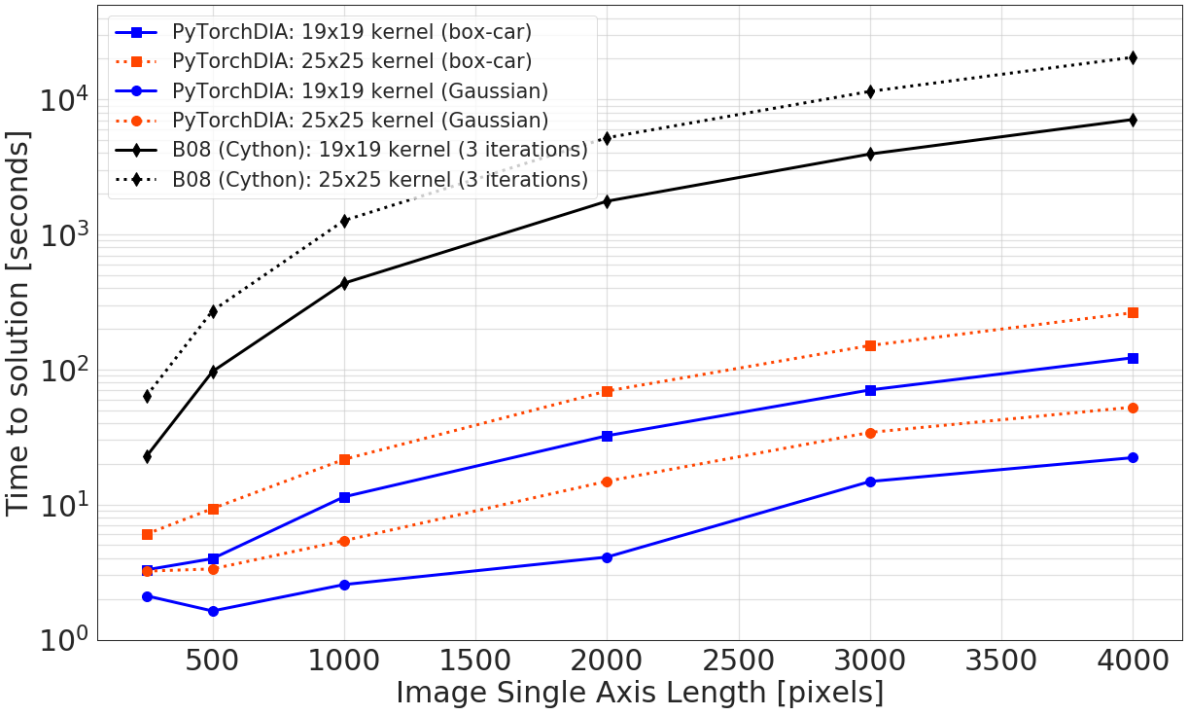


Figure 2.8: The time taken to solve for (square) kernels of different sizes against image single axis length (as a proxy for image size) for our PyTorchDIA implementation on the GPU and the B08 approach for a pair of square synthetic images, cropped to different sizes.

This explains why for the smallest image size in Figure 2.8 the time to solve for some of the kernels takes longer than expected. Our solution time is then, not completely determined by the formal $\mathcal{O}(n^2m^2)$ scaling.

2.6.3 cuDNN: Accelerating convolution computations on NVIDIA GPUs

One major use-case of PyTorch is for deep learning, in which CNNs, consisting of many small kernels, are used for feature detection in image recognition tasks (for example, Lawrence et al., 1997; Krizhevsky et al., 2012). The current research interest in deep learning applications has motivated device manufacturers such as NVIDIA to develop highly tuned GPU-accelerated libraries, specifically designed to improve the performance of common processing operations (e.g. forward and backward convolution) with these small kernels. By default, PyTorch makes use of NVIDIA's cuDNN library to accelerate convolutions on NVIDIA GPUs. This library has access to both deterministic and non-deterministic algorithms to compute convolutions, which are selected heuristically to accelerate computations[15]. As the sizes of useful kernels in DIA

[15]While not explored here, cuDNN also contains tools to benchmark convolution computations for a given kernel and image size combination. When processing many images, it performs tests to assess which cuDNN algorithm performs best on the first example, caches this information, and uses this best performing algorithm on all further images in the data set. Turning this feature towards astronomical data set reduction with PyTorchDIA will be explored in future work.

are themselves fairly small, it is expected that PyTorchDIA will also benefit from these tools. Indeed, we have enabled the use of non-deterministic cuDNN algorithms to accelerate the computation of convolution operations in all the results presented so far in this paper. In this subsection, we explore the benefit of this library in the context of DIA.

For this test, we measure how for an image pair of fixed sizes, the time to infer the kernel scales with kernel size. We use the pair of synthetic 4000×4000 pixel images from the prior section, symmetrically cropped to smaller 1000×1000 images. We infer the associated convolution kernel (initialised with a box-car) by minimising the Gaussian negative log-likelihood for this pair of synthetic images, which contain no outlying pixels. The time to infer the kernel on the GPU for different cuDNN settings in shown in Figure 2.9.

For all small kernels tested here, allowing cuDNN to use non-determinstic algorithms for the convolution computations results in slightly faster inference times. There is a clear change in behaviour for kernels larger than 23×23 pixels, and while the non-deteriminsitic computations again seem to win overall, the scaling with kernel size is now far less predictable. This makes sense, as cuDNN is optimised for working with small kernels. Clearly however, for the larger kernels, small changes in kernel size can lead to sporadic changes in performance. This will most likely be due to changes in the choice of convolution algorithm implemented by the software for any given image-kernel size combination, although a full exploration of the different CUDA convolution algorithms, how they are chosen, and how they scale, is outside the scope of this work.

2.7 Conclusions

We have presented a new algorithm for difference image analysis, where we model the target image as the output of a simple convolutional neural network. The kernel is treated as a discrete pixel convolutional filter, with weights which can be fit for efficiently within the PyTorch architecture. Specifically, we make use of automatic differentiation and GPU support to perform a lightning fast optimisation of the image model. We validate the fit quality and photometric accuracy of our implementation against its closest classical DIA analogue, with both simulated and real images. Our algorithm is simple to understand, and written entirely in standard Python packages, with an emphasis on accessibility to the wider astronomical community.

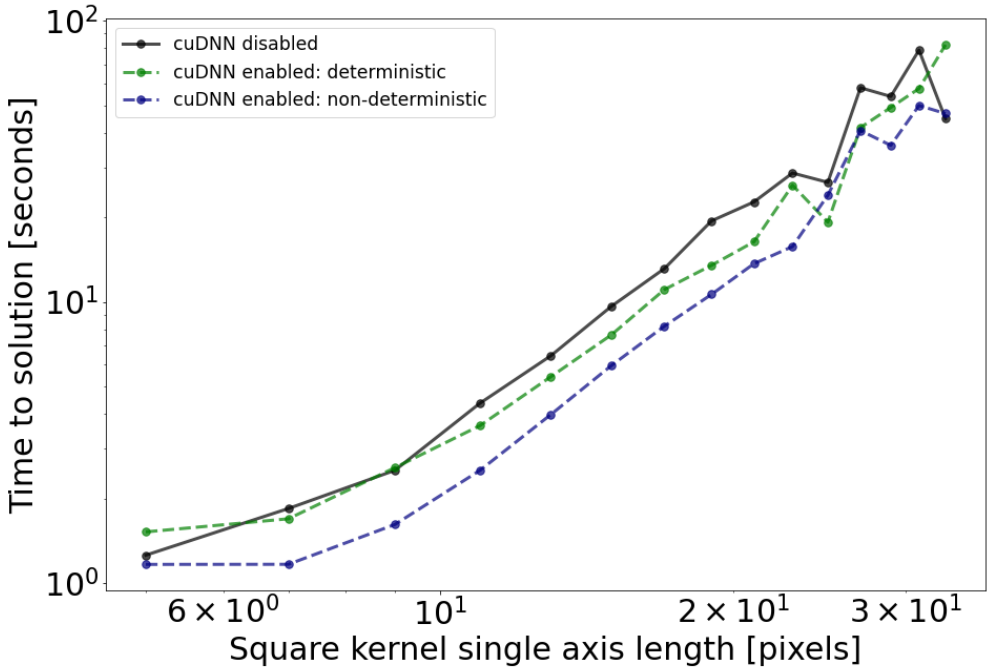


Figure 2.9: A comparison of the times taken to infer square kernels of different sizes with varying cuDNN settings for a given pair of 1000 × 1000 pixel synthetic images. Note the log-log scale.

Its benefits over some traditional approaches can be summarised as follows:

- *Speed*: We exploit the massive parallelism inherent to the convolution operation by making use of highly-tuned GPU-accelerated deep learning libraries to perform efficient convolution computations. With a good choice of learning rate, the optimisation procedure converges rapidly, and our algorithm can perform DIA on astronomical data sets at least an order of magnitude faster than its classical analogue.

- *Scalability*: For a pair of (square) images, each of size n, convolved with a (square) kernel of size m, our implementation approximately scales as $\mathcal{O}(n^2m^2)$, while the classical approach goes as $\mathcal{O}(n^2m^4)$.

- *Flexibility*: In our optimisation framework, we can maximise the (correct) Gaussian likelihood suitable for conventional CCD/EMCCD astronomical exposures (where the Gaussian approximation of Poissonian photon noise is valid), without resorting to an iterative procedure of χ^2 minimisation. This framework also allows us to relax the Gaussian noise assumption, and optimise robust scalar objective functions for images affected by outlying pixels. This provides a justifiable alternative to procedural sigma-clipping ap-

proaches. Further, we make use of automatic differentiation tools that free the user from having to manually recompute gradients if the paramaterisation of the model is changed, making experimentation by users straightforward.

The current main disadvantage to this approach is the use of 32-bit numerical precision to ensure the GPU-accelerated convolution computations are performed rapidly. Also, some engineering (e.g. choice of learning rates) is required by the researcher to get the best performance. And while the $\mathcal{O}(n^2 m^2)$ scaling typically holds for a given image-pair and kernel combination, our approach is a non-linear (convex) optimisation, and so the solution times for our approach are not entirely deterministic. We also note that access to mid- to high-end GPUs can be an issue, and their use in both the gaming market and cryptocurrency mining has caused a surge in prices. Given this, Tensor Processing Units (TPUs) – which now support floating point computations – could be a viable alternative for some use-cases.

We highlight automatic mixed precision training, available since PyTorch 1.6.0, as an area to explore in future work to further accelerate the convolution computations. Given the impressive recent advances in general purpose GPU computing, in large part driven by the deep learning community, we are well positioned to take advantage of improvements to these tools. Lastly, we again stress that the application to DIA is just one possible example of astronomical image processing that can benefit from deep learning tools, as very many useful image models include a convolution operation.

All code associated with the work in this chapter can be found at `https://github.com/jah1994/PyTorchDIA`.

3

The Thresher : Lucky Imaging without the Waste

3.1 Declaration

This chapter includes material adapted from: Hitchcock, J.A., Bramich, D.M., Foreman-Mackey, D., Hogg, D.W. and Hundertmark, M., 2022. The Thresher: Lucky imaging without the waste. Monthly Notices of the Royal Astronomical Society, 511(4), pp.5372-5384.

3.2 Introduction

The atmosphere produces high frequency temporal and spatial variations in the point-spread function (PSF) of ground-based astronomical images, as turbulent convective cells pass over the telescope aperture. The time-averaging effect of fluctuations of the PSF for conventional observations quickly degrades the resolution, and results in blurry, band-limited images. A class of techniques for compensating for this involve obtaining a large number of extremely short exposures close to the timescale of the atmospheric variations. These short exposures individually have low signal-to-noise and very complicated PSFs, but they provide strong con-

straints on the diffraction limited scene when many images are used collectively.

The idea of exploiting the high resolution information in short exposures was first suggested by Labeyrie (1970) and there has been a rich literature on this topic since. In particular, a very popular technique which we will call *traditional* lucky imaging (TLI) (Law et al., 2006) is based on Fried's derivation of the probability of obtaining a frame with unusually "lucky" seeing[1] when many images are taken quickly. That is, occasionally—just by chance—the wavefront distortions from the atmosphere will come close to cancelling the imperfections in your telescope. These best images are identified, shifted, and co-added, with the rest of the data relegated to the dustbin. TLI is very popular because it is simple to understand and implement, it is computationally tractable, and it can be used with very inexpensive equipment. As a technique, however, it is not *really* inexpensive because it results in a substantial amount of wasted data. Typical TLI results are based on only the best (i.e. sharpest) percentage of the acquired images, and progress has focused on improving the way images are co-added or selected to improve the efficiency of the method (Staley et al., 2010; Mackay, 2013, for example).

The work presented in this chapter was motivated by the (correct) feeling that, treated properly, there is no way that the discarded majority of data in a TLI imaging stack can be detrimental to constraining the astronomical scene! In the context of TLI, the fundamental reason why throwing away data *helps* is that the core data analysis step is shift-and-add co-addition of the imaging, which (though widely used in astronomy) is not justifiable when the point-spread function is varying rapidly.

The method we present here—*The Thresher*—is a new flavour of an old idea. It is a special case of a broad class of algorithms called *blind deconvolution* (Ayers & Dainty, 1988; Campisi & Egiazarian, 2017). It is closely based on the online multi-frame blind deconvolution (OMFBD) image analysis method of Hirsch et al. (2011) in particular. However, our algorithm differs from this prior work in several important ways: 1) *The Thresher* implements a physically motivated, justifiable likelihood function for short-exposure images; 2) our algorithm implements a robust stochastic gradient descent procedure based on state-of-the-art optimisation algorithms, and 3) It is entirely general to the choice of image model and likelihood function, as it makes use of automatic differentiation tools. Points 1) and 2) in particular make *The Thresher*

[1] A measure of the blurring of an astronomical image due to atmospheric turbulence.

extremely well suited to dealing with realistically faint and noisy high frame-rate astronomical imaging data.

In this work, we describe the theory behind our algorithm and the details of our particular implementation. Most importantly, we demonstrate the significant improvements over TLI made possible by this technique using both simulated and real data from Electron-Multiplying (EM) CCD, fast imaging cameras. A fundamental limitation of TLI is that the signal-to-noise of the final co-add is *inversely related* to its resolution (i.e. the better the resolution, the poorer the signal-to-noise). We show that this need not be the case; *The Thresher* has the potential of returning an image with the signal-to-noise of the entire data set at the resolution of the very best images.

This chapter is structured as follows. In Section 3.3, we describe the inference problem and the algorithm developed to solve it. We present our results on simulated and real data in Sections 3.4 and 3.5, respectively. In Section 3.6, we probe current limitations of the algorithm and outline the scope for further improvements. Section 3.7 states our conclusions.

Finally, in order to help the reader keep track of the notation introduced in the following sections, we include a table of symbols and their definitions in Table 3.1).

3.3 Problem Formulation

3.3.1 Online Multi-frame Blind Deconvolution

We start by writing down a model for our imaging data. For any short exposure, y_n, in our Lucky Imaging (LI) data set of N images, a reasonable model for the light distribution at any given ij pixel is

$$m_{n,ij} = [k_n \otimes s]_{ij} + b_n \, . \tag{3.1}$$

Equation 3.1 states that each image is generated by the convolution of some high-resolution, 'true' image s, with some unknown blur kernel, k_n, and some additive sky background, b_n. Throughout this work, we assume that s is time invariant, and that both k_n and b_n are spatially invariant. The latter is justified for the small angular area LI data that we use – on the order of the scale of the isoplanatic patch – where the seeing is approximately constant over the field-of-view.

Given some statistical model for the *noise* in our imaging data, we can write down a likeli-

Symbol	**Definition**
Variables	
$\boldsymbol{\theta}$	vector of image model parameters
m	model for an image
k	blur kernel
s	scene model
b	sky background
$\boldsymbol{y}$	vector of images
y	single image
N	total number of images
n_{pix}	total number of pixels in a single image
L	total loss for imaging data
l	loss for a single image
ϕ	hyper-parameter for tuning the L1 regularisation on k
α_0	SGD learning rate
$\hat{\mu}$	exponentially decaying average of the gradients of the loss w.r.t s
$\hat{\nu}$	exponentially decaying average of the squared gradients of the loss w.r.t s
$\boldsymbol{\Omega}$	vector of noise model parameters
f	A/D conversion factor
G	electron-multiplying (EM) gain
σ_0	readout noise
c	spurious charge
q	quantum efficiency
λ	rate parameter of the Poisson distribution
Y	image co-add
M	model for an image (image co-add)
K	blur kernel (image co-add)
B	sky background (image co-add)
σ	pixel uncertainties (image co-add)
Operators and special functions	
$\otimes$	convolution
∇	vector differential operator
$\|\| \cdot \|\|_1$	L1 norm
H	Heaviside step function
I_1	modified Bessel function of the first order
Iterables and indices	
I	number of complete passes over all images
n	single image index
t	SGD update index
i	image row pixel index
j	image column pixel index

Table 3.1: A table of notation used in this paper.

hood function for y_n given our vector of model parameters, $\boldsymbol{\theta} = [k_1, ..., k_N, b_1, ..., b_N, s]$. The notation is kept light for clarity. For each y_n image, k_n is modelled as a square array of pixels of some user-specified size, and b_n is a scalar. s is modelled as an array of pixels with the same dimensions as the data images. Assuming independence between images, and independence between pixels, the likelihood function for our entire data set, $\boldsymbol{y} = [y_1, ..., y_N]$, is just the product of the individual pixel likelihoods,

$$p(\boldsymbol{y}|\boldsymbol{\theta}, \boldsymbol{\Omega}) = \prod_{n=1}^{N} \prod_{ij} p(y_{n,ij}|k_n, b_n, s, \boldsymbol{\Omega}) \,. \tag{3.2}$$

We introduce $\boldsymbol{\Omega}$ to represent the list of other known quantities the likelihood is conditional on e.g. the parameters of the detector noise model.

For numerical convenience, it is helpful to instead work with logarithms, which transforms the above product into a sum. We can then define the total *loss* as the negative log-likelihood of our imaging data given the model parameters

$$\begin{aligned} L(\boldsymbol{y}; \boldsymbol{\theta}, \boldsymbol{\Omega}) &= -\ln p(\boldsymbol{y}|\boldsymbol{\theta}, \boldsymbol{\Omega}) \\ &= -\sum_{n=1}^{N} \sum_{ij} \ln p(y_{n,ij}|k_n, b_n, s, \boldsymbol{\Omega}) \\ &= \sum_{n=1}^{N} l(y_n; k_n, b_n, s, \boldsymbol{\Omega}) \,, \end{aligned} \tag{3.3}$$

with $l(y_n; k_n, b_n, s, \boldsymbol{\Omega})$ being the loss associated with a single image.

Unfortunately, there are a couple of practical complications that prevent us from straightforwardly minimising Equation 3.3 to obtain the maximum-likelihood estimate (MLE) for our model parameters. Firstly, LI data sets consist of thousands of images, and it is not practical to load these all into computer memory simultaneously. Secondly, and most importantly, in this blind deconvolution setting, for each y_n image, k_n and s are perfectly degenerate, as there exists a limitless number of possible combinations of these parameters that could generate the observed data (Equation 3.1).

Hirsch et al. (2011) proposed an iterative *online* algorithm, which needs to access only a single image at any time, and makes the problem tractable, consisting of two steps for each y_n

image in our data set:

$$\text{(i)}\ [\hat{k}_n,\ \hat{b}_n] = \underset{k_n \geq 0,\ b_n}{\arg\min}\, l(y_n; k_n, b_n, s_t, \boldsymbol{\Omega}) \tag{3.4}$$

$$\text{(ii)}\ s_{t+1} \leftarrow s_t - \alpha_t \nabla_{s_t} l(y_n; \hat{k}_n, \hat{b}_n, s_t, \boldsymbol{\Omega})\,. \tag{3.5}$$

In step (i), the current estimate of the scene, s_t, is kept fixed, and the unique k_n and b_n which minimise the loss for the given y_n are inferred. In step (ii), these new estimates for the blur kernel and sky background are used to re-evaluate the loss. The gradient of the loss is computed with respect to s_t, which is then updated with a single steepest descent step with step-size α_t, to give us a new estimate for the high-resolution scene, s_{t+1}. In this way, we can sequentially update our estimate of s by repeating steps (i) and (ii) for every image we can access. If required, multiple I passes can be made over the imaging data (i.e. giving a total of $I \times N$ updates).

As k_n and b_n are unique to each y_n, these should be inferred by minimising the loss for any given image. s however is common to all images, and so only a single update should be performed in step (ii). This update is an example of stochastic gradient descent (SGD; see Bottou et al. 2018 for an excellent overview). For settings in which it is impractical to compute the gradient of the total loss with respect to our model parameters, ∇L, SGD allows us to make progress by computing *stochastic* approximations to the total gradient, ∇l. Despite being noisy approximations to ∇L, optimisation by SGD typically makes rapid initial progress, as when far from the optimum, ∇l will very likely have the same sign as ∇L. This progress will generally be very sensitive to the step-size, particularly once in the vicinity of the optimum, but with some appropriately decaying step-size[2], SGD can be shown to converge in both convex and non-convex settings (Bottou et al., 1998). In practice however, it is useful to terminate the optimisation early, as issues with model representation can occur as the scene is deconvolved.

In order to stabilise and automatically anneal the stochastic updates to s, we propose a practical modification to step (ii) by introducing the following update into our SGD step,

$$s_{t+1} \leftarrow s_t - \frac{\alpha_0}{\sqrt{\hat{\nu}_t} + \epsilon}\hat{\mu}_t\,, \tag{3.6}$$

where $\hat{\mu}_t$ and $\hat{\nu}_t$ are the (unbiased) exponentially decaying running averages of the gradi-

[2]This is hugely problem specific, and typically requires empirical tuning. Despite SGDs popularity and long history, the optimal choice of step-size for a given optimisation problem is an open research problem.

ent and squared gradients respectively, and serve as estimates of the mean and (uncentered) variance of the gradients. This is an example of the popular Adam update, which adaptively tunes the learning rate for the parameters in a way that is sensitive to their gradient history.

We refer the reader to Kingma & Ba (2014) for definitions of how $\hat{\mu}_t$ and $\hat{\nu}_t$ are computed. Therein, we adopt the suggested values for the hyper parameters. This includes the value for ϵ, which is a small number added to the denominator of Equation 3.6 to improve numerical stability.

It is useful to think of the quantity $\hat{\mu}_t / \sqrt{\hat{\nu}_t}$ as something like the signal-to-noise ratio (SNR) of the gradients. The Adam update builds *momentum* for parameters with persistent non-zero gradients, and exerts friction when the variance of past gradients grows. This is useful, since parameters with a low 'SNR' should be characterised by oscillating about some optimum (or they are just insensitive to the data), and so their step-size decreases automatically.

For our problem, s has many free parameters – modelled as pixels on a grid – associated with both the sky background and astronomical sources of interest. Provided we can reliably estimate the sky level with our model (Equation 3.1), the gradient histories of pixels associated with only this smooth background (i.e. regions in s not populated with sources) should typically be centered around 0. Consequently, their effective step-sizes are naturally annealed by the update in Equation 3.6. Conversely, pixels associated with sources should have a greater consistency in gradient sign and lower variances, and so their effective step-size is annealed less aggressively. This has the overall effect of stabilising the deconvolution, as it helps to suppress noise amplification[3] in the updates to s.

Another major advantage of this SGD framework is that the only hyperparameter that needs some empirical tuning is the learning rate, α_0. This can be set independently for every parameter, and for this problem, we strongly recommend that it is! As argued above, the pixels associated with astronomical sources in s should be allowed to vary the most – for this is where the data are most informative – and so these parameters should be assigned larger initial learning rates. Further, α_0 has the added advantage of approximately bounding the size of the per-parameter updates for each SGD step. In this way, it performs a function very similar to the update clipping scheme used by Lee et al. (2017) to stabilise the deconvolution of noisy astronomical images.

[3]Inaccuracies in either step (i) or (ii) will propagate to the next step in a positive feedback loop.

3.3.2 A Poisson-Gamma-Normal Noise Model for EMCCD Data

Having written down a model for our imaging data, and an algorithm for inferring its parameters, we now need to choose a suitable loss function (i.e. a negative log-likelihood) that represents our belief about the observation noise. In this work, we consider observations acquired with Electron-Multiplying CCDs (EMCCDs).

A distinguishing feature of astronomical scenes is their huge dynamic range; some objects are very bright, and many more are typically very faint. Objects of interest in individual LI images will exist at very low signal-to-noise, with many below a reasonable detection threshold in any given exposure. Furthermore, collections of sufficiently short LI exposures are richly informative, containing high frequency (spatial) information otherwise lost in conventional, band-limited, long exposures. These high frequency components in general exist at the lowest intensities, and will therefore be very sensitive to the noise. For all but the very brightest sources, in these photon-starved images, Gaussian approximations to Poissonian counting statistics do not apply, and as will be seen, the electron-multiplication process particular to EMCCDs introduces an additional source of noise. An accurate statistical model which incorporates our *physical* knowledge of the noise generating processes of EMCCDs should allow us to better exploit this information, and take full advantage of the maximum-likelihood method for making robust and accurate inferences from our imaging data.

Korevaar et al. (2011) and Hirsch et al. (2013) independently derived the same physically motivated likelihood function for EMCCD data. This probability density function (PDF) is built from convolutions of 1) Poissonian photon noise and spurious charge events, 2) The cascade amplification of charge in the EM register, which is well approximated by a Gamma distribution, and 3) Normally distributed readout noise. After setting $\mathbf{\Omega} = [f, G, \sigma_0, c, q]$ (refer to Table 3.2 for definitions of these detector parameters), in units of ADU^{-1}, this Poisson-Gamma-Normal (PGN) likelihood takes the form,

$$p_{\mathrm{PGN}}(y_{n,ij}|k_n, b_n, s, \mathbf{\Omega}) = f\, H[y_{n,ij}] \sqrt{\frac{\lambda_{n,ij}}{y_{n,ij}\, G^2}}\, I_1\left[\frac{2f}{G}\sqrt{\lambda_{n,ij}\, y_{n,ij}}\right] \exp\left[-\frac{f}{G}(\lambda_{n,ij} + y_{n,ij})\right] + \frac{f}{\sqrt{2\pi}\,\sigma_0} \exp\left[-\frac{1}{2}\left(\frac{f\, y_{n,ij}}{\sigma_0}\right)^2\right] \exp\left[-\frac{f}{G}\lambda_{n,ij}\right], \quad (3.7)$$

Parameter	Description and units
f	A/D conversion factor (e^-_{EM} / ADU)
G	Electron-multiplying (EM) gain (e^-_{EM} / e^-_{Photon})
σ_0	Readout noise (e^-_{EM})
c	Spurious charge (e^-_{Photon})
q	Quantum efficiency (dimensionless)

Table 3.2: EMCCD detector parameters. We adopt the same notation as Harpsøe et al. (2012), who differentiate between electrons generated before and after the EM amplification as e^-_{Photon} and e^-_{EM} respectively.

where our image model (Equation 3.1), enters through

$$\lambda_{n,ij} = qm_{n,ij} + (G/f)c \,. \tag{3.8}$$

Note that H is the Heaviside step function and I_1 is the modified Bessel function of the first order.

For individual y_n exposures and our image model (Equation 3.1), the loss function under this noise model is simply the sum of the per-pixel negative log-likelihoods,

$$l_{\mathrm{PGN}}(y_n; k_n, b_n, s, \boldsymbol{\Omega}) = -\sum_{ij} \ln p_{\mathrm{PGN}}(y_{n,ij} | k_n, b_n, s, \boldsymbol{\Omega}) \,, \tag{3.9}$$

and the total loss over all images is equal to

$$L_{\mathrm{PGN}}(\boldsymbol{y}; \boldsymbol{\theta}, \boldsymbol{\Omega}) = \sum_{n=1}^{N} l_{\mathrm{PGN}}(y_n; k_n, b_n, s, \boldsymbol{\Omega}) \,. \tag{3.10}$$

3.3.3 Noise Model Validation and Calibration

We will now assess how well the PGN noise model can capture the noise properties of real EMCCD images acquired with the Danish 1.54m (DK154) TCI 'red' camera (Skottfelt et al., 2015). Of course, real world data brings real world problems, and the distribution of DK154 image counts on *raw* images stored for analysis is not well represented by Equation 3.7.

Firstly, due to storage constraints, despite being read out at 16-bit precision, the pixel values in the raw images are quantised to integers. We therefore 'dither' the raw images prior to further calibration by adding to each integer pixel value a float randomly sampled from

the uniform distribution, $\mathscr{U} \sim (-0.5, +0.5)$. This replaces the discretised raw image with a plausible analogue representation as originally observed by the camera. Under the values of f and G that apply to DK154 data, the associated uncertainty introduced by this procedure is approximately an order of magnitude smaller than the readout noise, and can safely be ignored.

Secondly, as with conventional CCD images, systematic bias and inter-pixel quantum efficiency differences should be corrected for. After dithering, we therefore bias subtract and flat-field all EMCCD images. However, one additional step must be taken to calibrate the EMCCD images, as there is a significant bias level drift – caused by on-chip heat-dissipation associated with the fast readout and large current – unique to each and every image, that should be corrected. This drifting bias level can be measured by comparing the values of the pixels in the unilluminated overscan regions of the given image and the master bias frame[4]. As these pixels too are affected by the cascade EM amplification, Harpsøe et al. (2012) proposed using the truncated mean of the distribution of the difference in these pixel counts as an estimate of the bias level drift. Specifically, the truncated mean is taken to be the mean computed after rejecting the top 5% of counts, and this value is subtracted from the relevant science frame.

We note that, in principle, one could introduce another free parameter into our detector model to represent this bias level drift, whose value could then be optimised, which would leverage the power of our physically motivated noise model. This is definitely more accurate than the heuristic approach outlined above, but it is challenging to implement in practice as the likelihood surface with respect to this parameter is discontinuous – since the likelihood is piecewise about a point determined by this parameter – and so non-gradient based optimisation approaches would need to be used.

Calibrating the noise model parameters

We took a sequence of 500 DK154 dark images with the same camera settings that are used throughout this work. From these we can calibrate all but one of our noise model parameters. As these images are not illuminated, any 'signal' can be assumed to be solely due to spurious

[4]Every (i, j) pixel value in the master bias frame is the truncated mean of the corresponding pixel values from 1000 bias images, where the top 5% of counts for the given pixel are rejected. The master bias frame can be considered noiseless, since the mean pixel variation between master bias images is an order of magnitude lower than the readout noise (Section 5.2.1 of Skottfelt et al. (2015))

charge events, which is equivalent to substituting $\lambda_{n,ij} = (G/f)c$ into Equation 3.7. Consequently, we cannot determine q – which is in general very challenging to measure – and so we adopt the detector manufacturer's value of $q = 0.8$ throughout this work.

For a stack of $n = 1, ..., 500$ dark images acquired with an EMCCD camera with detector parameters $\mathbf{\Omega} = [f, G, \sigma_0, c, q]$, we minimise the following objective function,

$$\begin{aligned}\left[\hat{f}, \hat{G}, \hat{\sigma}_0, \hat{c}\right] &= \underset{f,G,\sigma_0,c}{\arg\min}\, L_{\mathrm{PGN}}(\boldsymbol{y}; \mathbf{\Omega}) \\ &= \underset{f,G,\sigma_0,c}{\arg\min} \sum_{n=1}^{N} l_{\mathrm{PGN}}(y_n; \mathbf{\Omega})\,, \\ &= \underset{f,G,\sigma_0,c}{\arg\min} - \sum_{n=1}^{N} \sum_{ij} \ln p_{\mathrm{PGN}}(y_{n,ij} | \mathbf{\Omega})\,,\end{aligned} \tag{3.11}$$

where all free detector parameters are modelled as unknown constants. As there is a single readout register, this is a reasonable approximation for f, G and σ_0. For the purposes of this work, we restrict our analysis to a 256×256 pixel central part of the DK154 camera, over which c is also approximately constant (see Figure 3 in Harpsøe et al. (2012)). Equation 3.11 was optimised with the Adam algorithm (which is outlined in Section 3.3.1, but here we are using the full gradient, ∇L_{PGN}, when updating). The model PDF (Equation 3.7) parameterised with the MLEs is overlain onto a normalised histogram of the data in Figure 3.1, with bin widths determined using the approach in Knuth (2006). The range of data to be binned was restricted to that with a predicted probability density of at least 10^{-6} under the MLE of the noise model. As there are $\sim 3.3 \times 10^7$ data points in total, this guards against large numbers of empty (or near-empty) bins. The relative difference between the model and binned data is plotted in the lower panel. We can see in Figure 3.1 that the PGN noise model captures the asymmetric, and heavy-tailed features of our data distribution arising from the EM amplified spurious charges. We stress that in no way should this model be thought of as the 'truth'; it is only an approximation. But it is a practically effective, physically motivated, and broadly accurate representation of our data.

Now equipped with an accurate, detailed noise model for the detector – with calibrated instrument parameters – we can proceed to tackle the deconvolution problem.

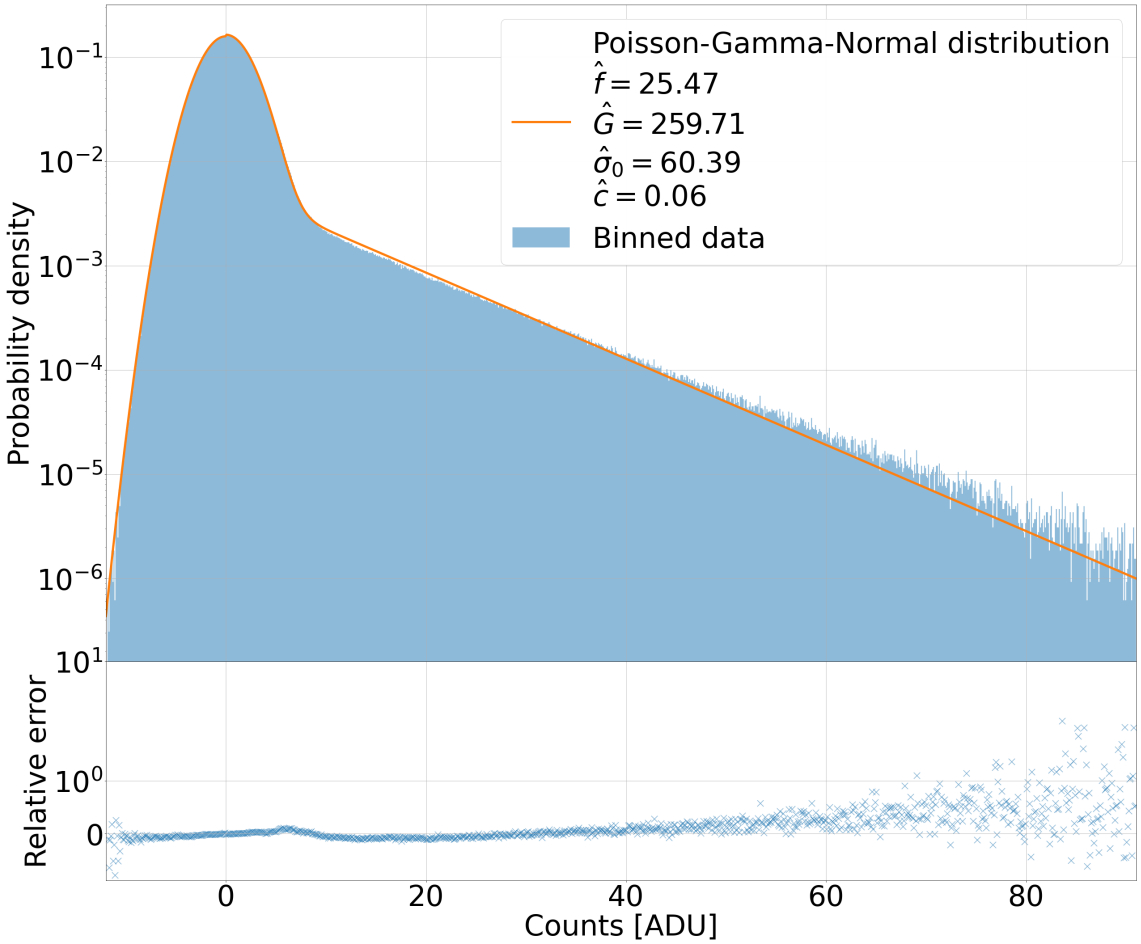


Figure 3.1: Normalised histogram of pixel counts from 500, 256 × 256 pixel (i.e. $\sim 3.3 \times 10^7$ data points in total), dark DK154 EMCCD images, overlain with the fitted Poisson-Gamma-Normal (PGN) likelihood. The positive fat tail is a result of amplified spurious charges. The relative difference between the model and data – computed as (data - model) / model – for each bin is plotted in the lower panel. The MLE for the detector parameters are also shown (see Table 3.2 for units).

3.3.4 Constraints and Regularisation

Due to the ill-posed nature of the blind deconvolution problem, it can be useful to include constraints and regularisation/penalty terms (i.e. Bayesian priors) on the model parameters to restrict the scope of their possible solutions, or push the model parameters to some *a priori* preferred value when the data are uninformative.

Non-negativity

In the deconvolution literature, it is very common to restrict the pixel values in the underlying scene s to non-negative values. For our problem, while it is reasonable to restrict the pixels in the blur kernel to non-negative values, it is less clear whether it is correct to enforce non-negativity on s. In any given LI image, there will almost certainly exist faint sources at or below the detection threshold that blend into the sky background. In the units of the final inferred scene, it could be that these sources have *negative* flux, and will be strongly affected by any non-negativity constraint.

In practice, this decision should, at least partly, be motivated by the adopted noise model. It seems reasonable to force s to be non-negative if using the PGN noise model, as this will guard against possible numerical issues with likelihood evaluations, since under this particular noise model, the image model m_n, should be non-negative everywhere; the PGN noise model does not permit negative photoelectrons[5]. However, if using a simple squared error loss (as we do for tests on simulated, noiseless images in Section 3.4.1), no such 'physical' argument for non-negativity of s can be made. In this case, the crucial point is that the data do not constrain the *absolute* value of any given parameter in s, but only the *relative* values of the parameters. Consequently, in this work, we *do not* enforce non-negativity on s in the tests on noiseless data (Section 3.4.1), but we *do* impose this constraint for the tests on data modelled with the PGN noise model (Sections 3.4.2 and 3.5).

Penalised Maximum-Likelihood-Estimation

Noise amplification is the perennial problem for many deconvolution algorithms, and this is only exacerbated by the extremely low signal-to-noise of our short exposure data. It is then

[5]For this reason, we also restrict $b_n \geq 0$ when using the PGN noise model, especially as we always sky-subtract s_0 to mitigate the anti-correlation between k_n and b_n. The differential background level between this sky-subtracted model and the data will be non-negative, justifying this constraint, which in combination with the non-negativity constraints on k_n and s will ensure m_n is always non-negative.

common to include penalty terms in the loss function to guard against excessively noisy blur kernels and scene estimates.

Given the degenerate nature of the inference problem, the accuracy with which we can determine each of the k_n kernels will strongly determine how we update s at each step. Indeed, extensive empirical experience gained from developing and testing *The Thresher* has shown us that almost all problems in the reconstruction of s can be traced back to poor kernel inference. The delta-function basis used to model the kernel is highly flexible, but this comes at the cost of increased overfitting, resulting in excessively noisy kernels, particularly when the signal-to-noise is low (Becker et al., 2012; Bramich et al., 2016, provide some instructive examples from the highly analogous Difference Image Analysis literature). This is a problem, as noise in the kernel will propagate to successive estimates for the scene. To guard against this, we include the L1 norm on the kernel pixels as a penalty term to our loss function, which pushes kernel pixels towards 0 unless the data are informative. Consequently, we can re-write step (i) (Equation 3.4) as

$$\text{(i)}\ [\hat{k}_n,\ \hat{b}_n] = \underset{k_n \geq 0,\ b_n}{\arg\min}\, l(y_n; k_n, b_n, s_t, \mathbf{\Omega}) + n_{\text{pix}}\, \phi\, ||k_n||_1\ , \tag{3.12}$$

where n_{pix} is the number of pixels in each y_n image, and ϕ is a tuning constant which sets the strength of the regularisation, and must be set empirically. In practice, we find values in the range of $\phi = 0.001 - 0.1$ to be reasonable, with lower signal-to-noise data requiring stronger kernel regularisation.

It is common to also include penalty terms on the scene which reflect our beliefs about the properties of images. These penalty terms typically disfavour high frequencies (e.g. sharp gradients) in the scene, and so have a smoothing effect, and can help suppress noise. Popular choices include the L1 or L2 norm on the image gradients. Unfortunately, these penalty terms actually *favour* blurry scenes (Levin et al., 2011). Additionally, they are prone to introducing correlated artefacts into the image model, and their gradient fields produce complicated PSFs which pose challenges for the purposes of any further measurements on the reconstructed image. These issues pose serious challenges for the astronomer, and distinguishes our use-case from the general image reconstruction setting. In astronomy, the target of the deconvolution must almost always be useful for the purpose of making *measurements*, and the PSF is typically one of the things that we are most interested in measuring! Having a practically useful model

for the PSF is extremely important for a variety of downstream tasks that we are likely to care about (e.g. source detection, astrometry, flux measurements).

Lastly, as discussed in Vio et al. (2005), the penalisation of the high frequencies of the image model could effectively nullify our efforts in implementing an accurate statistical description of the noise which attempts to exploit this low intensity, high frequency information. These penalty terms are designed to ensure that low /mid frequencies dominate the reconstruction. While unimportant for band-limited imaging data, this could badly affect the performance on LI data, as it prevents us from fully exploiting the unique advantages these short-exposures offer us in constraining the high-resolution estimate of the scene.

To summarise, in practice we find that as long as the blur kernel estimation is accurate, these simple penalisation terms for the scene are either unnecessary, or create more problems than they solve.

No sum-to-one constraint

There is a remaining multiplicative degeneracy between the k_n blur kernels and s. We have found that successive updates to s will increase its photometric scale, and the scale of the blur kernels will decrease accordingly in order to fit the data (Equation 3.1). This does not cause any problems for the results presented here, but for applications where preserving the photometric scale of the data in the reconstructed image is important e.g. (uncalibrated) source flux measurements, a sum-to-one constraint could be applied to the blur kernel pixels. Also, while not being a problem we have encountered, it is conceivable that numerical issues could arise as the scale of the kernels gets smaller, further motivating some sort of regularisation or constraint. For these reasons, we will experiment with implementing such a feature in future applications of our algorithm.

3.3.5 Algorithm and Implementation details

We sketch the pseudo-code for *The Thresher* below (Algorithm 1). Inferring the blur kernel and differential background in step (i) for every y_n image is the computational bottleneck of the algorithm. This problem has received renewed attention in astronomy in the context of Difference Image Analysis (DIA) – also known as image subtraction – in the advent of the Legacy Survey of Space and Time (Ivezić et al., 2019), and a variety of popular algorithms exist (Alard & Lupton, 1998; Bramich, 2008; Zackay et al., 2016, for example). However,

these approaches all assume a Gaussian noise model for the imaging data – which is leveraged to solve the problem analytically – and cannot be generalised to handle the PGN likelihood (Equation 3.7) appropriate for our short exposure EMCCD images. Consequently, we base our kernel solution approach on the *PyTorchDIA* algorithm (Hitchcock et al., 2021), which re-casts the DIA problem as an optimisation, making it general to the choice of objective function. As the name suggests, this code is built within the PyTorch machine learning framework (Paszke et al., 2019), and makes use of the powerful tools within – such as automatic differentiation and highly optimised, GPU-accelerated convolution computations – to make it performant.

Like *PyTorchDIA*, *The Thresher* also makes use of PyTorch to automatically compute gradients of the objective function with respect to the image model parameters. We stress again that this does not just bring advantages in computational efficiency. Because of these automatic differentiation capabilities, *The Thresher* is entirely general to the choice of the image model *and* noise model. Although we focus on the case of EMCCD images in this paper, *The Thresher* can straightforwardly generalise to work with other high frame-rate detectors with very different noise properties, such as the next generation of sCMOS devices (Qiu et al., 2013; Steele et al., 2016; Walker, 2020).

We should note that the convolution of s with a (square) kernel is undefined for pixels in s within half a kernel-width from its edges. Therefore, when fitting k_n and b_n in step (i) of the algorithm, these edge pixels in s and y_n do not enter the loss. Then, in order to ensure that the updated s preserves the original dimensions of the y_n images in step (ii), we must pad s when convolving it with $\hat{k}_n$. In this work, we used a naive zero-padding, and this may result in “edge-effects” in the reconstructed image; while the background level of s should be close to zero, it is not exactly zero, and any sources that are situated on the image edges could be strongly affected. This padding is a delicate but necessary operation, since without it s would become smaller and smaller at each iteration every time it was convolved with a new $\hat{k}_n$.

Also, in order to keep the kernel array size reasonable, we align each y_n image with s to the nearest integer pixel at each iteration. This avoids having to interpolate between pixel values, since any sub-pixel precision mis-alignment between y_n and s will be captured by $\hat{k}_n$ (Bramich, 2008). While the integer shifts between images should almost always be less than half a kernel width in size, and so will not affect the fit in step (i), any shift, regardless of size will affect the update in step (ii), as "missing" pixels around the image edges must be replaced

to preserve the dimensions of each y_n image. As with the aforementioned zero-padding, we again choose to substitute these missing pixels with zeros as a crude estimate of the sky level in the short exposure imaging, which will also contribute to any edge-effects in the reconstructed image.

Algorithm 1 *The Thresher*

Input: Stream of images y_n for $N \geq 2$
Output: Reconstructed image s
(*scene*) Initialise s_0 as the sharpest shift-and-added X% of TLI images and (optionally) subtract some estimate of its sky background
(*scene*) Set α_0 so that it is proportional to s_0
(*kernel*) Set size of kernel array and ϕ

for I passes over the data **do**

    while another image y_n available **do**
        Align y_n with s_t to the nearest integer pixel
        (i) $[\hat{k}_n, \hat{b}_n] = \underset{k_n \geq 0, b_n}{\arg\min} l(y_n; k_n, b_n, s_t, \mathbf{\Omega}) + n_{\text{pix}} \phi ||k_n||_1$
        (ii) $s_{t+1} \leftarrow s_t - \frac{\alpha_0}{\sqrt{\hat{v}_t + \epsilon}} \hat{\mu}_t$

end
return last estimate s_t

3.4 Simulated Image Tests

We first test our algorithm on a detailed simulation of a Lucky Imaging data set. This artificial telescope is similar to the DK154, with the same mirror size, but a pixel scale at 0.05 ”/pix, rather than the actual DK154s LI camera’s 0.09 ”/pix sampling. The DK154 ‘red’ camera has a broadband filter with an effective wavelength of $\sim$ 800 nm, and so we adopt this as the wavelength for this monochromatic simulation.

Simulating Lucky Imaging data

We used the HCIPy package (Por et al., 2018) to generate the atmospheric phase screens needed to simulate the complicated LI PSFs. We adopted a single layer model generated by the infinitely long phase screen extrusion method of Assémat et al. (2006), which provides a memory efficient approach to generating realistic, time-evolving phase screens as would be observed over the course of a long LI run. Observing conditions were chosen to be similar to those under ‘good’ La Silla seeing conditions, at 0.6 arcseconds, corresponding to a Fried parameter, $r_0 = 27$ cm at 800 nm. Archival GSM data were used to set the outer scale of the

turbulence to $\mathscr{L}_0 = 24$ m, and the wind speed to $v = 5.8$ m/s (Martin et al., 1998).

Planar light waves – consisting of real and imaginary components – are propagated through this atmospheric model and are incident on a circular telescope aperture with diameter $D = 1.54$ m. The *instantaneous* speckle-pattern observed on the detector at the focal plane of the telescope is then obtained by applying a fourier transform to this wavefront and down-sampling it to the appropriate pixel scale, at 0.05 arcseconds per pixel.

The *coherence time* for speckle-patterns associated with this single layer atmospheric model can be computed as $\tau_e \sim 0.31D/v = 80$ ms (Tubbs, 2003). In practice, the Danish 1.54m usually takes 100 ms exposures, and so some degree of time-averaging will occur. To simulate the PSFs associated with each 100 ms exposure, we therefore time-average 100 speckle patterns sampled at 1 ms intervals.

Equipped with a simulation for the PSF, we can now populate a set of images with point sources in the following way.

(i) We first define a 128×128 pixel grid, and choose this to be populated with 10 point sources.

(ii) Fluxes (in units of $e^-_{\rm Photon}$) are randomly drawn from $U \sim (10, 500)$ for every source.

(iii) The sub-pixel (x, y) positions of these 10 sources are randomly populated across the image, while avoiding the regions close to the edges. The brightest source is moved to the exact centre of the image.

(iv) In each image, the associated 0.1 s time-averaged PSF is scaled to the flux from (ii) at the position of (iii) on the grid defined in (i).

(v) A sky level of 0.1 $e^-_{\rm Photon}$ is then added to each image.

(vi) Finally, for every pixel in each noiseless image, ADU values are randomly sampled from the PGN PDF (Equation 3.7), with $f = 25$ ($e^-_{\rm EM}$/ADU), $G = 300$ ($e^-_{\rm EM}/e^-_{\rm Photon}$), $\sigma_0 = 60$ ($e^-_{\rm EM}$), $c = 0$ ($e^-_{\rm Photon}$) and $q = 1$. This assumes that all contributions from spurious charges are zero, the quantum efficiency is flawless, and that instrumental effects such as per-frame bias level drift, have been perfectly corrected for.

In this way, we generate a stack of 3000 simulated 100 ms exposures.

3.4.1 Tests on Noiseless Images

Before deploying our algorithm on realistically noisy and coarsely sampled astronomical images, we will first assess what it is capable of learning from idealised, noiseless and well-sampled, high frame-rate imaging data. For this test, we use our spool of 3000 noiseless short exposure PSFs, which are generated at a pixel scale of ∼0.013 arcseconds.

The model image was initialised with the sharpest 1% of shift-and-added exposures. Figure 3.2 shows how this estimate of the high-resolution scene improves with successive SGD updates. We ran our algorithm for a total of four passes (equivalent to 12,000 SGD updates) over the noiseless image spool, and in Figure 3.3, we plot the diffraction limited Airy pattern, the sharpest 1% of shift-and-added exposures, and the model returned by our algorithm after 12,000 updates. As the data are noiseless, we adopted a simple squared error for the loss function (i.e. (data - model)^2). Under this loss function, we do not adopt any non-negativity constraint on the scene model. As the data are noiseless, we do not need to enforce any L1 regularisation on the kernel pixels. The SGD step-size was initialised as $\alpha_0 = 0.005 s_0$, and the kernel array was set to be 129×129 pixels large[6] .

Encouragingly, albeit in this highly idealised scenario, the model returned by *The Thresher* is able to recover something like the Airy diffraction pattern of the telescope. In fact, the central lobe of the model PSF is tighter than that of the instrument diffraction pattern. This can be more clearly seen by plotting the radial profiles of these images in Figure 3.4. We can see that the peaks of the rings returned by the model are at approximately the same location as those of the diffraction limited PSF, and the intensities of the peaks decreases at about the same rate with distance from the centre. However, for rings closer to the peak intensity, they tend to be shifted slightly towards the centre, and the central lobe itself is noticeably sharper.

It is reasonable to wonder if it really is possible for *The Thresher* to do better than the optics alone. In theory, the information to achieve this does exist in our *collection* of images. Due to atmospheric turbulence, the short exposure PSFs have wandering centroids, and so we get a slightly different view of the scene in each observation; there is additional spatiotemporal data available in the sequence of images. It is this extra information which allows super-resolution methods to reconstruct images sharper than the diffraction limit (Borman & Stevenson, 1998).

[6] In this case of noiseless data and a simple squared error loss function, we found that fitting the large kernel array in step (i) of our algorithm was more quickly achieved with the L-BFGS method (Liu & Nocedal, 1989), and so we used that optimisation algorithm to minimise Equation 3.4 in this test, and this test only.

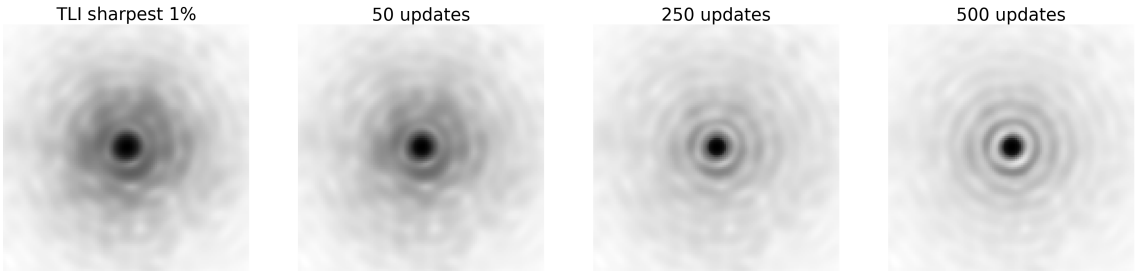


Figure 3.2: The model image returned by our algorithm after (left to right) 0, 50, 250 and 500 SGD updates from the test on noiseless short exposure PSF images.

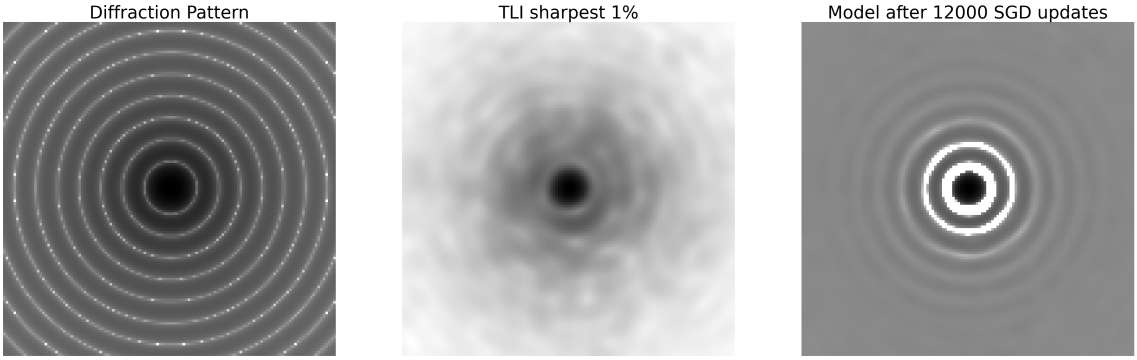


Figure 3.3: 128 × 128 pixel cutouts of the (from left to right) theoretical instrument diffraction pattern, the shift-and-added sharpest 1% of images and the model returned by our algorithm after 12000 updates from the test on the 3000 noiseless short exposure PSF images. All cutouts are normalised relative to the highest intensity pixel, and are plotted on a logarithmic scale. The squared error loss used by our algorithm for these noiseless data does not require non-negativity of the model, and so we do not enforce this constraint. Consequently, some regions close to the centre went negative, and for the purposes of plotting on the log-scale we added a small offset.

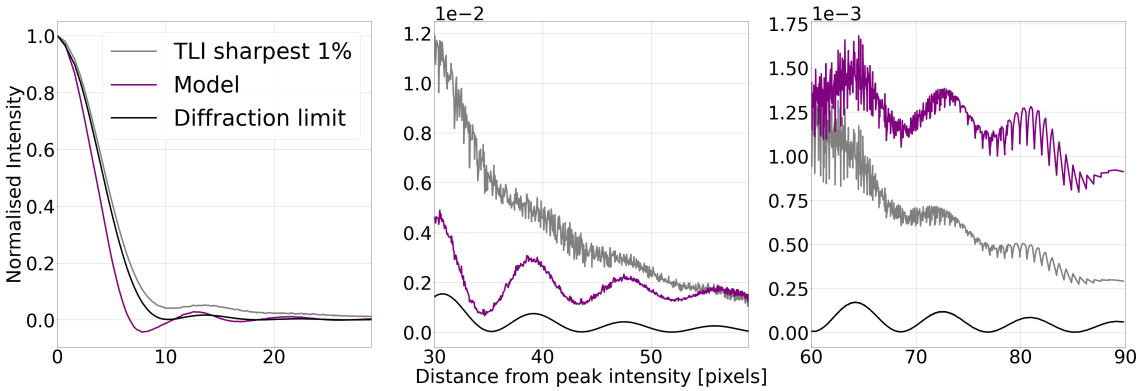


Figure 3.4: Radial profiles of the PSF images in Figure 3.3. Each (normalised) intensity value is the mean of all pixels at the given distance from the peak. The pixel scale is ∼ 0.013 arcseconds per pixel.

3.4.2 Tests on Noisy Images

Next, we run our algorithm on our simulated DK154 EMCCD data (see Section 3.4 for a description of the how the images are generated). The initialisation for the model, s_0, was constructed as the TLI sharpest 50% of images. The median pixel value of s_0 was subtracted as an estimate of the background, and non-negativity was enforced. We set the kernel array size to be 25×25 pixels, $\phi = 0.07$, $\alpha_0 = 0.005 s_0$, and we made a single pass over the data set (i.e. 3000 SGD updates).

We plot the model image and the TLI sharpest 1% and 50% coadds in Figure 3.5; to highlight the noise in the images, they are all displayed on a log-scale, with a small positive offset added for visual clarity. The coadd of the sharpest 1% images is sharp, but at very low signal-to-noise. The point sources in the coadd of the sharpest 50% have high signal-to-noise, but this comes at the expense of resolution, and the PSF exhibits a broad, extended halo characteristic of shift-and-add methods.

The Thresher, after having been fed 100% of the images in the spool, delivers the best of both worlds; the image model is as sharp as the best images in the data set, but at substantially higher signal-to-noise. Plots of the radial profile of the central brightest star – normalised to the same scale – are shown in Figure 3.6. The width of the PSF in our image model is comparable in sharpness to the PSF in the TLI sharpest 1% coadd, but appears much smoother due to the significant improvement in signal-to-noise.

It is important to note that we have not run our algorithm to some converging resolution. Unlike in the highly idealised scenario in Section 3.4.1, as the image model starts to approach something close to the diffraction limited scene, we encounter issues with representing s on this now coarsely sampled grid. This in turn badly affects the per-image kernel inference, leading to a positive feedback loop of bad updates to the image model. This issue could be mitigated by representing s and each k_n at a finer pixel sampling, and including a down-sampling operation in the forward model to match the resolution of the data (i.e. a super-resolution approach). However, for typical astronomical images, the associated computational expense can become prohibitive, and in practice, it is not always necessary to achieve this resolution. The utility of our algorithm is that it can do *better* than TLI in terms of resolution and signal-to-noise. For most science goals, this is generally a more useful objective to aim for than a perfect deconvolution.

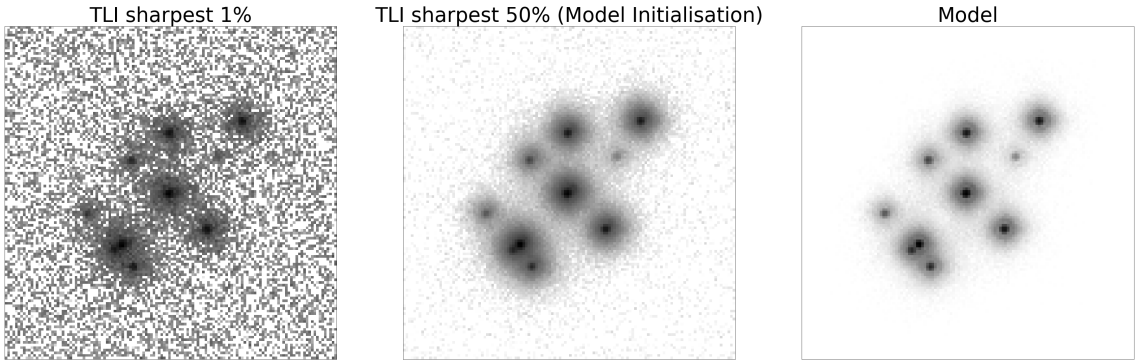


Figure 3.5: A comparison of the TLI sharpest 1% and sharpest 50% of images, and the image model returned by *The Thresher* after a single pass over a stack of 3000 simulated short exposures. For comparison with the model, the TLI coadds have had their respective median pixel values subtracted (as estimates of their sky levels) and non-negativity enforced. All images are on a log-scale.

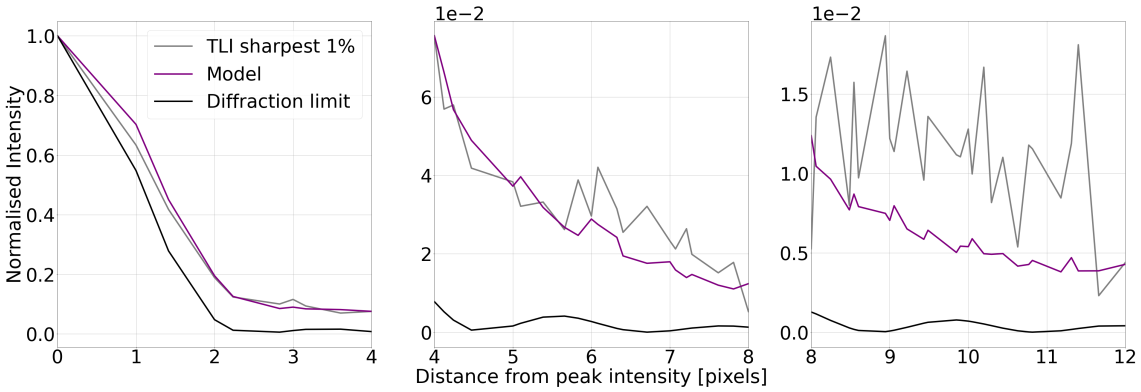


Figure 3.6: Radial profiles of the bright, central star in Figure 3.5 in the sharpest 1% TLI coadd and the image model returned by *The Thresher*. The theoretical diffraction limited PSF is plotted for comparison. Each (normalised) intensity value is the mean of all pixels at the given distance from the peak. The pixel scale is ~ 0.05 arcseconds per pixel.

3.5 Real Image Tests

Here, we run *The Thresher* over a spool of 4800 real 0.1 second DK154 'red' (i.e. $\sim$ 800nm wavelength) EMCCD exposures, streaming through the y_n images chronologically. The target is a central 20×20 arcsecond (i.e. 256×256 pixel) region of the globular cluster NGC 7089, and the time-averaged seeing was ~ 0.8 arcseconds. We adopt the noise model parameters estimated from the fits to dark images in Section 3.3.3 to calibrate our likelihood function (Equation 3.7). All raw images were dithered (as described in Section 3.3.3), and then bias subtracted (including the per-frame drift level) and flat-fielded. A cosmic ray was identified on a single image in the spool; the affected pixels were fairly isolated from any sources, and so their values were simply replaced with the median value of the image, as an estimate for the sky background.

s_0 was again initialised as the shift-and-added sharpest 50% of images. Next, we subtracted an estimate of the 2D sky background[7] of s_0 and enforced non-negativity. The step-size was initialised as $\alpha_0 = 0.005 s_0$, and the tuning constant controlling the L1 regularisation on the kernel was set to $\phi = 0.03$. The kernel array was set to be 25×25 pixels, and we make a single pass over the data (i.e. 4800 SGD updates).

In TLI, the SNR of the coadd can only be improved by stacking increasingly blurrier images, and so the resolution rapidly worsens. Provided *The Thresher* is able to accurately account for the individual frame PSFs, feeding more data to the algorithm will not degrade the resolution of the image model and the SNR should increase. This is exactly what we see in Figures 3.7 and 3.8[8]; the image model returned by our algorithm is of comparable sharpness to the sharpest 1% of images in the stack, but at substantially higher signal-to-noise.

Indeed, many faint sources that are either blended and/or barely distinguishable above the background in the coadds are clearly visible in the image model. This capability of *The Thresher* to reconstruct even very faint sources was an important goal during its design, and this directly motivated the development of the robust SGD procedure and accurate, physically motivated EMCCD noise model.

We note that some sources close to the edges of the TLI coadds cannot be clearly viewed in

[7] This was estimated using the background tools in the photutils package (Bradley et al., 2016).

[8] In reality, the Danish 1.54m is not diffraction limited, and the limiting resolution is set by the mirror support and optical system. Consequently, the sharpest achievable PSFs in any given short exposures appear somewhat triangular, and the diffraction limit PSF in Figure 3.8 is purely instructive.

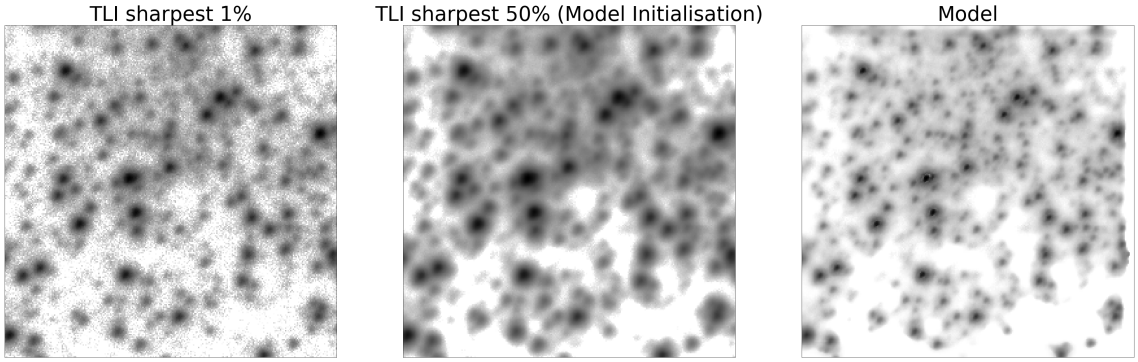


Figure 3.7: A comparison of the TLI sharpest 1% and 50% of images, and the image model returned by *The Thresher* after a single pass over a stack of 4800 real short exposures. For comparison with the model, the TLI coadds have had their respective sky background's subtracted and non-negativity enforced. All images are on a log-scale.

the model, and there is a prominent absence of flux along the model's right-hand edge. This is due to issues with the convolution operation being undefined for these border pixels, and data images being aligned with the model, as explained in Section 3.3.5.

3.6 Current limitations of *The Thresher* and the scope for future work

3.6.1 Flux Non-linearity

One attractive property of coadding astronomical images is that source fluxes are preserved. In the general case, where the sky background level of the data are non-zero (i.e. every $b_n > 0$), deconvolution algorithms provide no such guarantee. Historically, this was an area of interest during the early years of the Hubble Space Telescope (HST), where a variety of deconvolution algorithms were used to partially compensate for the aberrations introduced by the defective primary mirror (see White & Allen 1991 for an overview). Some of these studies indicated that the deconvolution introduced systematic non-linearities in flux in the reconstructed image, and so an additional correction was needed to calibrate the photometry. This issue motivated the development of deconvolution algorithms adopting a "two-channel" approach, which have been shown to preserve source fluxes by decomposing the deconvolved image model into point sources and background (Hook & Lucy, 1994; Magain et al., 1998, for example).

The Thresher, as presented in this chapter, is not designed to explicitly distinguish between point sources and sky background in the scene, and so we should not expect the relative photometric scale of the data to be preserved in the reconstructed image if the background is

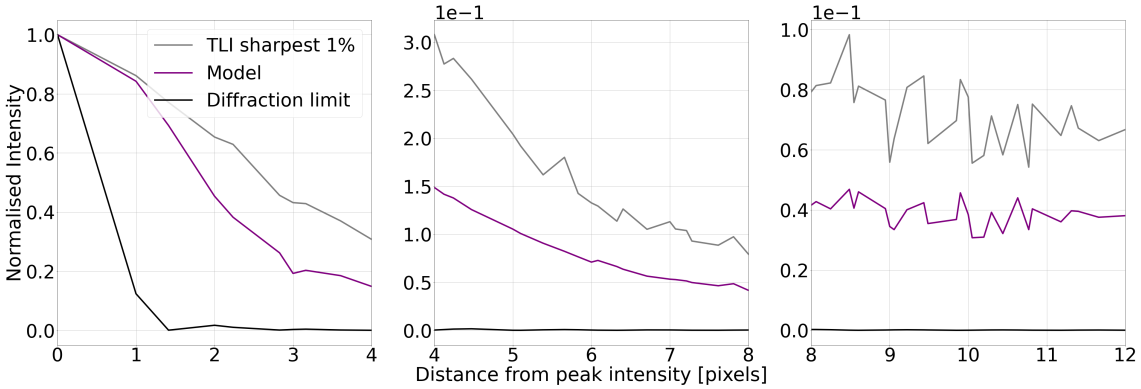


Figure 3.8: Radial profiles of a bright, fairly isolated, centrally located star in Figure 3.7 in the sharpest 1% TLI coadd and the image model returned by *The Thresher*. The theoretical diffraction limited PSF is plotted for comparison. Each (normalised) intensity value is the mean of all pixels at the given distance from the peak. The pixel scale is ~ 0.09 arcseconds per pixel.

non-zero. We probe this behaviour using the results of Sections 3.4.2 and 3.5, where $b_n > 0$ for every y_n image. We do this by fitting image models (of the form in Equation 3.1) to the TLI coadds of all the images in each data set. If a reconstructed image has preserved the relative photometric scale of sources in the data, then a plot of pixel values in (the kernel convolved) s against their counterparts in the coadd should be consistent with a straight line with a slope of 1, and intercept 0. As we are now fitting an image model to the shift-and-added stack of the short exposure images, Y, we can adopt a Gaussian approximation for its noise, where only the photon contributions are significant. We adopt upper-case notation here to make clear that we are referring to coadds of the y_n short exposures, so Equation 3.1 now becomes

$$M_{ij} = [K \otimes s]_{ij} + B \,. \tag{3.13}$$

Ignoring the irrelevant normalisation constant, the negative log-likelihood takes the form,

$$\begin{aligned} l_{\mathrm{N}}(\,Y\,;K,B,s,f,G\,) &= -\sum_{ij} \ln p_{\mathrm{N}}\left(\,Y_{ij}\,|\,K,B,s,f,G\,\right) \\ &= \frac{1}{2}\sum_{ij}\left(\frac{Y_{ij}-M_{ij}}{\sigma_{ij}}\right)^2 + \sum_{ij} \ln\,\sigma_{ij}\,, \end{aligned} \tag{3.14}$$

with photon shot-noise limited per-pixel uncertainties[9]

$$\sigma_{ij}^2 = 2M_{ij}\frac{G}{f}\,. \tag{3.15}$$

[9]Note that although we are in the Gaussian limit, there is no analytical approach to minimising this function due to the photon shot noise. The factor of two in Equation 3.15 is referred to as the 'excess noise' factor, and accounts for the probabilistic cascade amplification (Korevaar et al., 2011; Hirsch et al., 2013).

We can then return the MLEs of the PSF-matching kernel and sky background that fit s to the coadd,

$$[\hat{K},\ \hat{B}] = \underset{K,\,B}{\arg\min}\, l_{\mathrm{N}}(Y; K, B, s, f, G) \tag{3.16}$$

and compute $\hat{M}$ via Equation 3.13. We do this for both of the final estimates of s returned by the tests on simulated and real images (Sections 3.4.2 and 3.5), fitting each of these s estimates to their respective stacks of 100% of the shift-and-added data. Plots of pixel counts in the image model, $\hat{M}$, against their respective counts in the coadds are shown in Figure 3.9. In the middle and bottom panels we plot the residuals from the straight line representing photometric consistency between the model and data; residuals in the bottom panels are expressed as a percentage of the model value (i.e. as a percentage of the model brightness). Uncertainties on the points are equal to the square root of the variances given by Equation 3.15.

The two plots suggest that *The Thresher* does indeed introduce non-linearities into the flux of the reconstructed image. Specifically, fluxes of bright objects are systematically overestimated; in both plots, the residuals from a consistent photometric scale become more negative with increasing model brightness. This is most clearly seen in the right panel of Figure 3.9, as the dynamic range of sources in the crowded field is large. The asymmetry in the residuals at the faint end is an artefact of the 2D background subtraction of the scene initialisation.

Some experimentation has suggested that this problem decreases by allowing *The Thresher* to run for longer, making several passes over the data (similar to findings by Lindler et al. 2013). However, this has to be balanced with computational expense and issues with model representation as the reconstructed sources become increasingly sharper. Given the importance of preserving the local photometric scale of the data in the reconstructed image for many science cases, this is highlighted as something to address for future work on our algorithm.

3.6.2 Computational expense

The computational bottleneck of our algorithm is set by the speed with which we can fit k_n and b_n (Equation 3.12), as this requires computing a convolution for the estimate of the forward model (Equation 3.1) at each step in the optimisation as l is minimised. The convolution operation is embarrassingly parallel, and so *The Thresher* leverages GPUs if they are available.

Lucky Imaging data sets consist of thousands of images, but since the algorithm only ever has to access a single image at a time, memory management is not an issue. Nonetheless, it

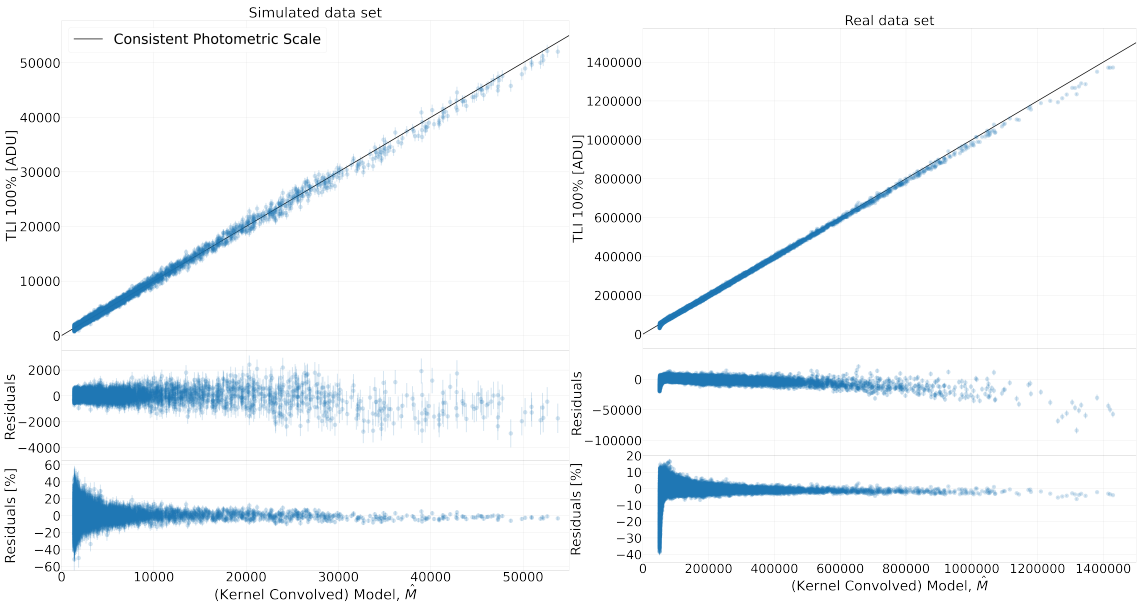


Figure 3.9: A plot of pixel values of the (kernel convolved) reconstructed image vs. the respective counts in a TLI 100% selection coadd from the tests in (left) Section 3.4.2 and (right) Section 3.5. The straight line – with gradient 1 and intercept 0 – represents photometric consistency between the model and data as a function of flux. The residuals in the bottom panel are expressed as a percentage of the model value.

must process these many images, and so must perform *very many* convolutions. All tests in this paper were run on modest hardware – a laptop with a single GeForce GTX 1050 – and so execution times were fairly slow, at about 2.5, 6 and 20 hours for the tests in Sections 3.4.2, 3.5 and 3.4.1 respectively. Desktop GPUs will of course help to beat-down those times, particularly if the images are large, but the technique remains computationally expensive. Indeed, although the code is designed to be agnostic to the hardware, if only a CPU is available to the user, it may prove to be computationally intractable for most problems.

3.7 Conclusions

We have presented *The Thresher*, a new algorithm for processing Lucky Imaging data. It fundamentally differs from Traditional Lucky Imaging (TLI) shift-and-add procedures in that it optimises a physically motivated likelihood function for the entire set of imaging data to recover the underlying astronomical scene. Because it uses the full data set, *The Thresher* outperforms TLI in signal-to-noise; because it accounts for the individual-frame PSFs, it outperforms TLI in angular resolution.

We implement an accurate noise model for low light level, Electron Multiplying CCD data. When combined with a robust stochastic gradient descent procedure and appropriate regulari-

sation of model parameters, *The Thresher* can cleanly reconstruct even extremely faint sources, which may be below the detection threshold in any given short exposure. *The Thresher* is entirely general to the choice of image model and noise model, as it makes use of automatic differentiation tools, and has been designed to anticipate near future high-frame rate imaging on sCMOS devices. Furthermore, it makes use of GPUs to massively accelerate the many convolution computations.

While the image model returned by the current version of *The Thresher* can be used for, among other things, astrometry and source detection, systematic non-linearities in flux must be corrected for if accurate photometry is needed. Addressing this issue will be a focus for future development of our algorithm.

We end by commenting that the ideas in *The Thresher*, while particularly well suited to high frame-rate imaging data, can also be useful for fitting models to conventional images. To this end, we also include the option to adopt a Gaussian log-likelihood for the noise in the imaging (of the form in Equation 3.14) suitable for conventional, ground-based CCD exposures, as part of our software implementation. While the deconvolution would no longer benefit from the high frequency information uniquely available to short exposures, *The Thresher* provides an interesting alternative to co-addition approaches when the image-to-image PSF is varying.

Development of the software implementation of our algorithm is ongoing, and we refer the reader to the following Github repository `https://github.com/jah1994/TheThresher`.

4

A data system for the serendipitous detection of small outer solar system objects

4.1 Declaration

The work in this chapter has been done in collaboration with Prof. Richard Gomer (Texas A&M University). It describes the results of a pilot study, and the work has not been published nor submitted for review.

4.2 Introduction

The serendipitous occultation of background stars by outer solar system objects is recognised as a key approach to understanding the size and quantity of small (on the scale of <10 km radius) objects in the Kuiper belt, and perhaps even the Oort cloud (Nihei et al., 2007; Bickerton et al., 2008; Bianco et al., 2009; Siraj & Loeb, 2020). Because of their small light reflecting area these objects are extremely faint and direct detection by even the largest telescopes remains

unfeasible.

Understanding the population of these far out objects is crucial for testing planetary system formation models (Duncan & Levison, 1997; Davis & Farinella, 1997; Benz & Asphaug, 1999; Pan et al., 2005; Benavidez & Bagatin, 2009; Raymond et al., 2018; Hands et al., 2019). For the solar system, such models predict that the objects populating the Kuiper belt are the members of the far-out objects of the system's protoplanetary disk that failed to form planets, due to the slow rates of collisions at these distances. The Oort cloud is thought to be populated by planetesimals that initially formed closer to the Sun, but were ejected to distant orbits by gravitational perturbations from the young giant planets, and is likely a resevoir of long-period comets that enter the inner solar system (Hills, 1981; Weissman, 1990; Dones et al., 2004). Ascertaining the size, spatial and mass distributions of these far-out objects is therefore a necessary step in understanding the solar system's evolutionary history.

To date, from both ground and space-based observations, a small number of sub-kilometre-sized Kuiper Belt Object (KBO) candidates have been reported (Chang et al., 2011; Liu et al., 2015; Arimatsu et al., 2019b), and a likely sub-kilometre KBO has been detected (Schlichting et al., 2009). There have however been no confirmed detections of Oort cloud objects. The Transneptunian Automated Occultation Survey (TAOS II) is a dedicated serendipity survey for detecting these occultation events (Lehner et al., 2012). The project will use three 1.3 m telescopes equipped with sCMOS cameras to observe $\sim 10,000$ stars close to the ecliptic simultaneously at 20 Hz. After a delay due to the COVID 19 pandemic, observations are expected to begin shortly.

The use of high frame-rate imaging in both predicted and serendipitous occultation surveys is a popular approach to detecting small outer solar system objects (Roques et al., 2009). These include dedicated missions, such as the aforementioned TAOS II and its precusor, the Taiwanese American Occultation Survey (TAOS) (Bianco et al., 2010). These surveys are complemented by other high frame-rate astrophysics missions – in which stellar occultations are just one of many science cases — where medium to large-sized telescopes are equipped with fast frame-transer CCDs (Dhillon et al., 2007; Doressoundiram et al., 2013), Electron-Multiplying CCDs (Dhillon et al., 2014), and most recently sCMOS cameras. The low noise and rapid frame rates that characterise these new cameras make them an exciting choice for both dedicated occultation surveys (Pratlong et al., 2016; Wang et al., 2016; Mazur et al., 2022) and

observations targeting predicted occultation events (Arimatsu et al., 2019a).

In this chapter, we document the initial phase of work for an occultation survey complementary to TAOS II. Many of the next generation of ground-based telescopes will have unused areas within the circular optical focal surface e.g. the 11.25 m Maunakea Spectroscopic Explorer (MSE) will sample light within a circumscribed hexagonal area within this region (Marshall et al., 2019). We propose to place high frame-rate sCMOS cameras within these otherwise unoccupied parts of the focal surface. The image quality within these regions will be appropriate for flux measurements of objects. The disadvantage of this is that we cannot prioritise observations of dense star fields close to the ecliptic to maximise the chance of detecting KBO or inner Oort cloud object occultations, but we believe this is more than offset by our unique practical advantage of taking up no dedicated telescope time; we will just take high frame-rate imaging of whatever is in the FoV as the telescope works through its schedule of observations.

There are many challenges associated with this work. Several overlap with TAOS II, so we have looked to them for some inspiration. Some are however unique to this project. In particular, we propose to take high-cadence photometry at frame rates on the order of 100 Hz, which is considerably higher than that in TAOS II. The justification for running a survey at these high frame rates is that the majority of far out solar system objects are thought to be sub-kilometre in size (Kenyon & Luu, 1999; Kenyon, 2002; Bottke Jr et al., 2005). For the Kuiper belt, there are predicted to be 1-2 orders of magnitude more 1 km bodies than 10 km bodies, and 1-2 orders of magnitude more 0.1 km objects than 1 km objects, and it's speculated that the size distribution of objects in the Oort cloud should be similar. Such small objects produce shorter duration occultation signals which would be poorly sampled at 20 Hz, attenuating the depth of the occultation, and making it more likely to be missed in a survey, let alone characterising the event, which is also made challenging if not impossible due to the sampling resolution. Further, the occultation signals associated with such small objects are weak, and even if they were sufficiently well sampled at 20 Hz, they would be challenging to detect with a 1.3 m telescope.

For this initial phase of the work we have used Teledyne Photometrics' new "Kinetix" camera, which can readout a 3200x3200 pixel area as fast as 498 Hz[1]. At these frame rates, just

[1]https://www.photometrics.com/products/kinetix-family/kinetix

a second of observing time produces $\sim$ 5 Gb of imaging data. Storing the imagery is therefore totally impractical for any long-term serendipity survey. We must therefore process the images in real time, and once we've extracted the information useful for our purposes, *throw the image away*, never to be seen again. This will make many astronomers uncomfortable, and if it doesn't, it probably should, but needs must. Outside of astronomy, there are precedents for doing this in research limited by data storage capabilities, such as the use of a "trigger" in particle physics experiments [2].

This chapter is arranged as follows. Section 4.3 briefly presents the expected signatures produced in photometric time series of occultation events by small solar system objects. We then describe our approaches to real time image processing and occultation event detection in Section 4.4. In Sections 4.5 and 4.6 we report the results from trialling our data system over 7 nights of observations with the 2.1m telescope at McDonald Observatory, Texas. Our conclusions are given in Section 4.7.

4.3 Occultation Light Curves

As comprehensively outlined in Nihei et al. (2007), the occultation light curves of small outer solar system objects can display signatures associated with Fresnel diffraction effects. The signal is not just a function of the size, distance and impact parameter of the object (the distance of closest approach between the observer and the centre of the occultation shadow), but also the angular size of the background star and observation wavelength, and consequently the stellar type. For reference, we list the symbols of the occultation event parameters and their associated definitions in Table 4.1.

In Figures 4.1 - 4.3, we show the expected noise-free occultation signals for a variety of parameter configurations at the sampling frequencies of TAOS II (20 Hz) and the fiducial frequencies used in the observations presented in this chapter (80 and 300 Hz)[3]. The blue curves correspond to a KBO objects at 40 AU, and the orange curves are for objects at 1000 AU i.e. at the scale of the supposed inner edge of the Oort cloud.

Note the diffraction effects seen in the occultation light curves of the more nearby objects.

[2] `https://www.lhc-closer.es/taking_a_closer_look_at_lhc/0.lhc_trigger`

[3] The code used to generate these signals was adapted from `https://github.com/ekpass/colibri/blob/master/fresnelModeler.py`. Modifications were made to model occultations viewed outside of opposition, in addition to the effects of integrating over the Kinetix response curve convolved with a particular stellar spectrum.

Symbol	Definition
r	Radius of an occulting object
a	Distance to an occulting object
b	Impact parameter of an occulting object
θ_*	Angular size of the background star
ϕ	Observation opposition angle
f	Observation sampling frequency
t	Time
I	Relative brightness

Table 4.1: Table of occultation event parameters and their definitions

The background star is a main sequence star of spectral type A0V, whose spectrum was obtained from the Pickles (1998) catalogue. In this work, we will observe with no filter to maximise our data signal-to-noise ratio (SNR), and we model this by convolving the A0V star's spectrum with the Kinetix camera's response curve, and then integrate the occultation intensity pattern over the wavelength range $200-1050$ nm. The effect of this is to smooth the diffraction fringes in the occultation light curves that would otherwise be observed at a single wavelength, or with a narrow filter (Nihei et al., 2007)

The bigger and closer the object is, the deeper the associated dip in relative brightness. Less obviously, as seen in Figure 4.3, the larger the angular size of the background star, the more the event depth is dampened. Indeed, the occultation signature of a fairly large 2.5 km radius object at a distance of 1000 AU is neglible when $\theta_* \geq 0.2$ mas, which is approximately the size of a Red Giant at a distance of 50,000 light years.

These are the expected signals if viewing the event at opposition (i.e. $\phi = 0$), where the Earth's orbital velocity is orthogonal to the background star. It is worth noting that the length of the events would be increased at shallower viewing angles, since the Earth then spends a longer time in the occultation shadow. This consideration, in addition to wanting to minimize the angular size of the background star, factored in to our choice of targets in Section 4.5, where we prioritised young open clusters close to the line of the Earth's orbital velocity, and containing fairly young main sequence stars.

With an eye to the future of this project, we also include a plot of occultation signals produced by sub-kilometre diameter objects in Figure 4.4. This highlights the benefit of us running

a survey at sampling frequencies substantially higher than the 20 Hz used for TAOS II. Due to the shorter timescales of these events, the occultation depths are reduced by roughly half when sampled at 20 Hz, and the signal is more likely to be missed in the survey data. However, the overall depths are small compared to those in Figure 4.1 – with even very nearby Oort cloud objects producing negligible occultation signals – and detection is likely only feasible in data acquired on the larger, next generation of telescopes such as the MSE.

The improvement in SNR by mounting the Kinetix on telescopes with large mirrors such as the MSE has the added benefit of allowing us to collect sufficient flux from distant stars, with small angular sizes. As hinted at above, this is of particular importance for the detection and characterisation of solar system objects further out than the Kuiper belt, (see the right-hand column of Figure 4.3). For a telescope with a mirror of diameter D observing a star of radius r_* at distance a_*, the observed flux, F, is

$$F \propto D^2/a_*^2\,, \tag{4.1}$$

and the stellar angular size is approximately equal to

$$\theta_* \approx r_*/a_*\,. \tag{4.2}$$

Consequently, for observing a star of the same spectral type ($r_{*,1} = r_{*,2}$) at a greater distance ($a_{*,2} > a_{*,1}$), with a larger telescope ($D_2 > D_1$) such that the observed flux is the same ($F_1 = F_2$), the angular size would be smaller by a factor of D_1/D_2[4].

For this pilot study, we used the 2.1 m McDonald telescope and targeted nearby open clusters, whose bright main sequence A0V stars have angular sizes of about 0.02 mas. By mounting the Kinetix on the 11.25 m MSE, we could observe such stars at the same SNR at a distance such that their angular size would be $0.02 \times (2.1/11.25) \approx 0.004$ mas. The benefit of using these more distant background stars for searching for far out solar system objects is made clear by Figure 4.5, showing the occultation signals of a 5 km radius object at a distances of 5000 AU and 10,000 AU, passing in front of an A0V type star with angular sizes of 0.004, 0.02 and 0.2 mas. For stellar angular sizes of 0.02 and 0.2 mas, the occultation light curve

[4]With the caveat that there would be increased extinction for the more distant object, which is strongly dependent on its galactic latitude. This will decrease the brightness of the star and redden its spectrum.

is smoothed, and the resulting dip depth is greatly attenuated, complicating detection. The occultation of this object in front of the smaller, more distant star however would produce a detectable signal.

Regardless of the parameter configurations, the distinguishing features of these events is that they occur on sub-second timescales. To search for occultations, we therefore need a highly sensitive imaging detector with a fast readout and low noise, as exemplified by the latest generation of sCMOS cameras.

4.4 The Data System

We define the "data system" as comprising of the camera, PC and software. The 29.4 mm diagonal FOV 3200 × 3200 pixel "Kinetix" sCMOS camera was released by Teledyne Photometrics in early 2021. A distinguishing feature of this camera is that it can be operated in four distinct observing modes[5]. For this study, we work in either Sensitivity or Speed mode: the former has a maximum frame-rate of 88 Hz at 12 bits/pixel while maintaining low readout and dark current, while the latter can achieve 498 Hz at the expense of increased noise and a decreased 8 bits/pixel depth. The Kinetix is connected to the PC with two 50 m fibre optic PCI cables. In 2021, our colleague Darren DePoy at Texas A&M University fabricated an adaptor for this $10 \times 10 \times 13$ cm camera for the 2.1 m telescope at McDonald Observatory, which was used for a series of successful observations in September 2021, but due to the immense data volume produced at these frame rates, we were only able to take data for a few minutes at a time when running the camera at these high frame rates. Below, we explain our solution to this problem.

4.4.1 Real Time Data Processing

Because the Kinetix camera generates gigabytes of image data within seconds of observing, we have written software to extract the information useful for our project – namely the aperture photometry and sky background vales in the FoV – before throwing the image away.

The main computational challenge is to extract the stellar aperture photometry from the images on the timescale of ~ 1 millisecond. This requires calibration steps such as bias subtraction and flat correction, the summing of pixels at each aperture, and the determination of sky levels in the field. To achieve this, we were motivated by two important insights 1) We

[5]https://www.photometrics.com/wp-content/uploads/2021/11/Kinetix-Datasheet-Rev-B2-02112021.pdf

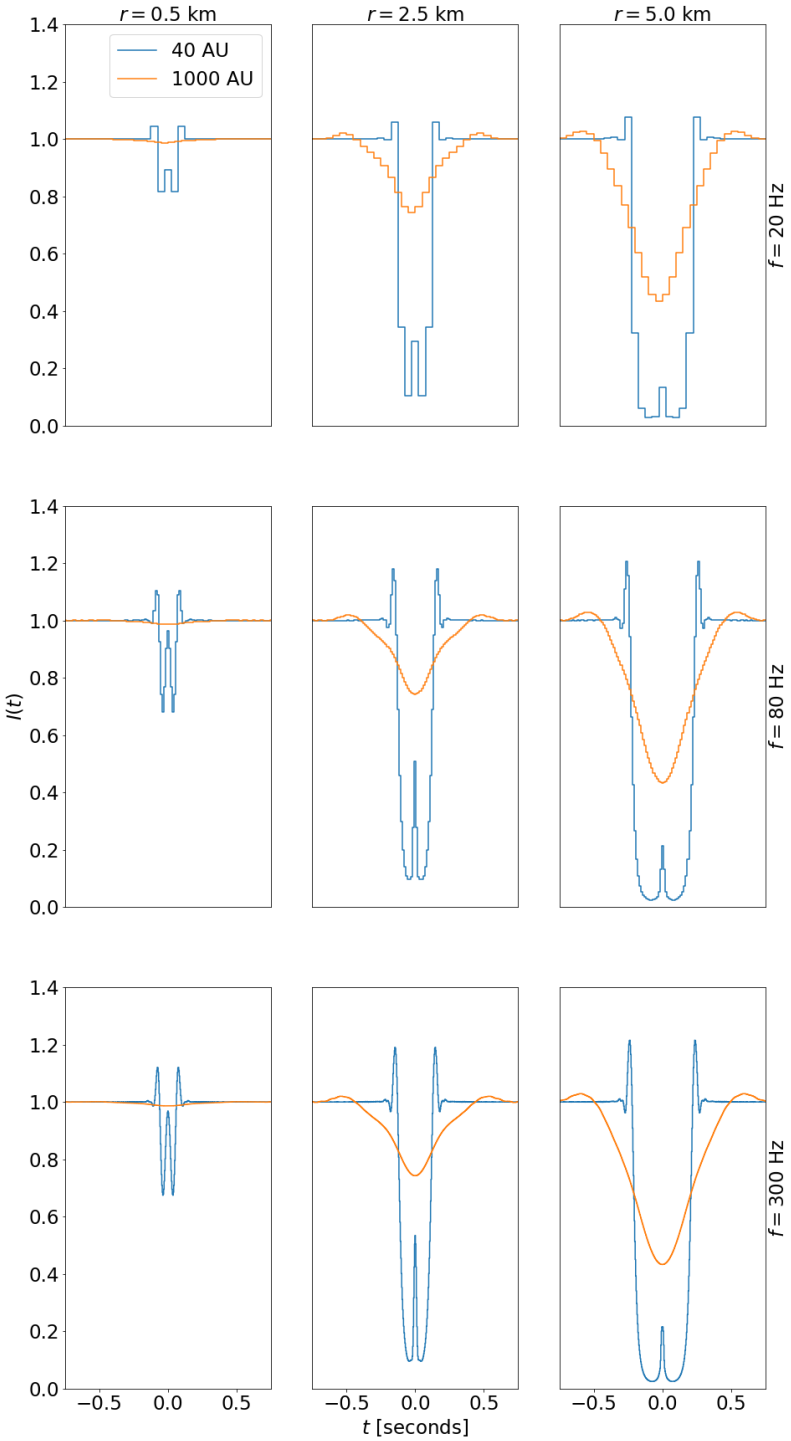


Figure 4.1: Noise-free occultation light curves that would be observed with the Kinetix with no filter for circular objects of radii 0.5, 2.5 and 5 km at distances of 40 and 1000 AU, passing in front of an A0V background star, viewed at opposition. The impact parameter and background star angular size are set to $b = 0$ km and $\theta_* = 0.02$ mas respectively. Each row corresponds to the fiducial sampling frequencies of $f = 20$, 80 and 300 Hz for the TAOS II mission, and the Sensitivy and Speed modes of the Kinetix camera used in this work.

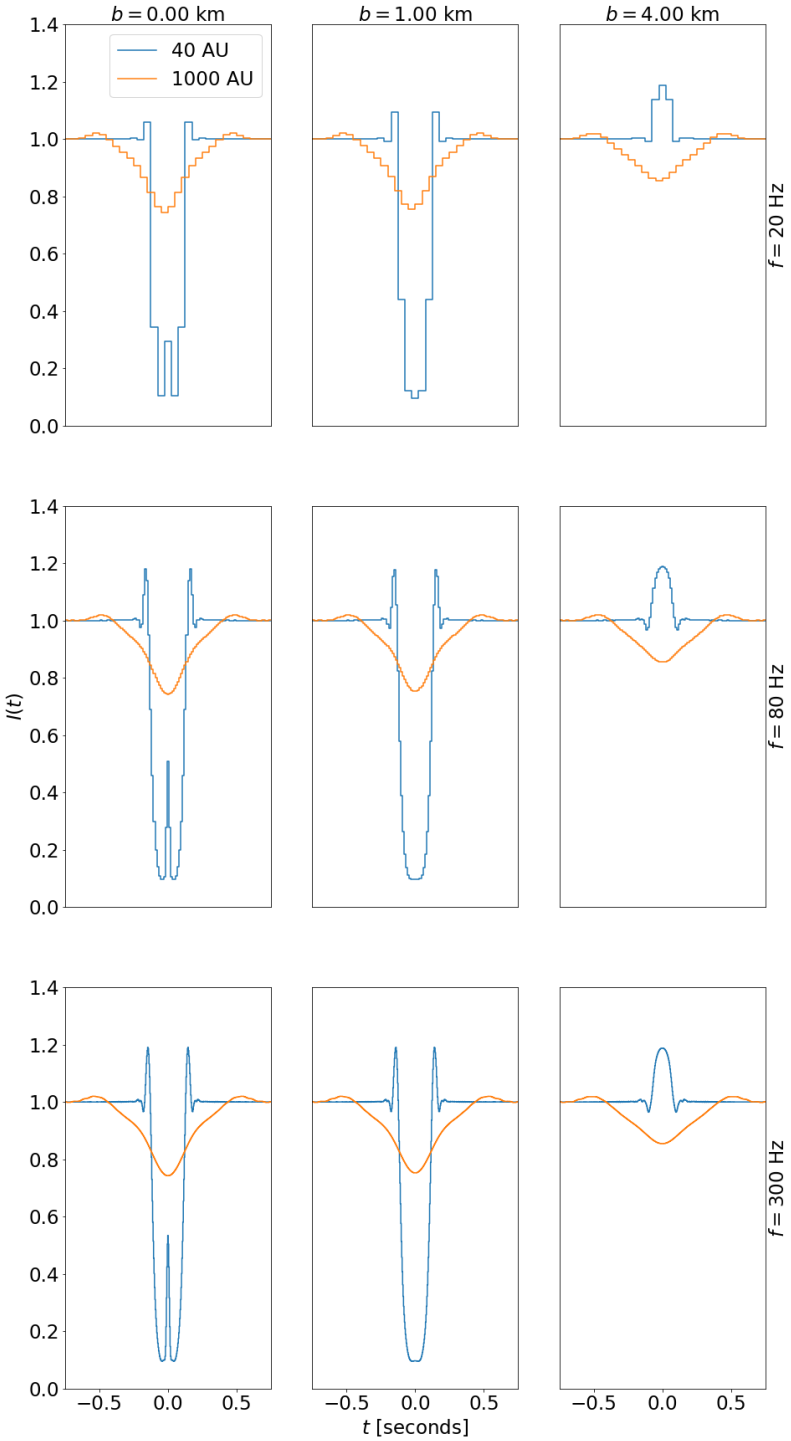


Figure 4.2: Noise-free occultation light curves that would be observed with the Kinetix with no filter for a 2.5 km radius circular object at distances of 40 and 1000 AU, passing in front of an A0V background star, viewed at opposition. From left to right, the impact parameter is set to $b = 0, 1$ and 4 km respectively. The background star angular size is set to $\theta_* = 0.02$ mas. Each row corresponds to the fiducial sampling frequencies of $f = 20$, 80 and 300 Hz for the TAOS II mission, and the Sensitivy and Speed modes of the Kinetix camera used in this work.

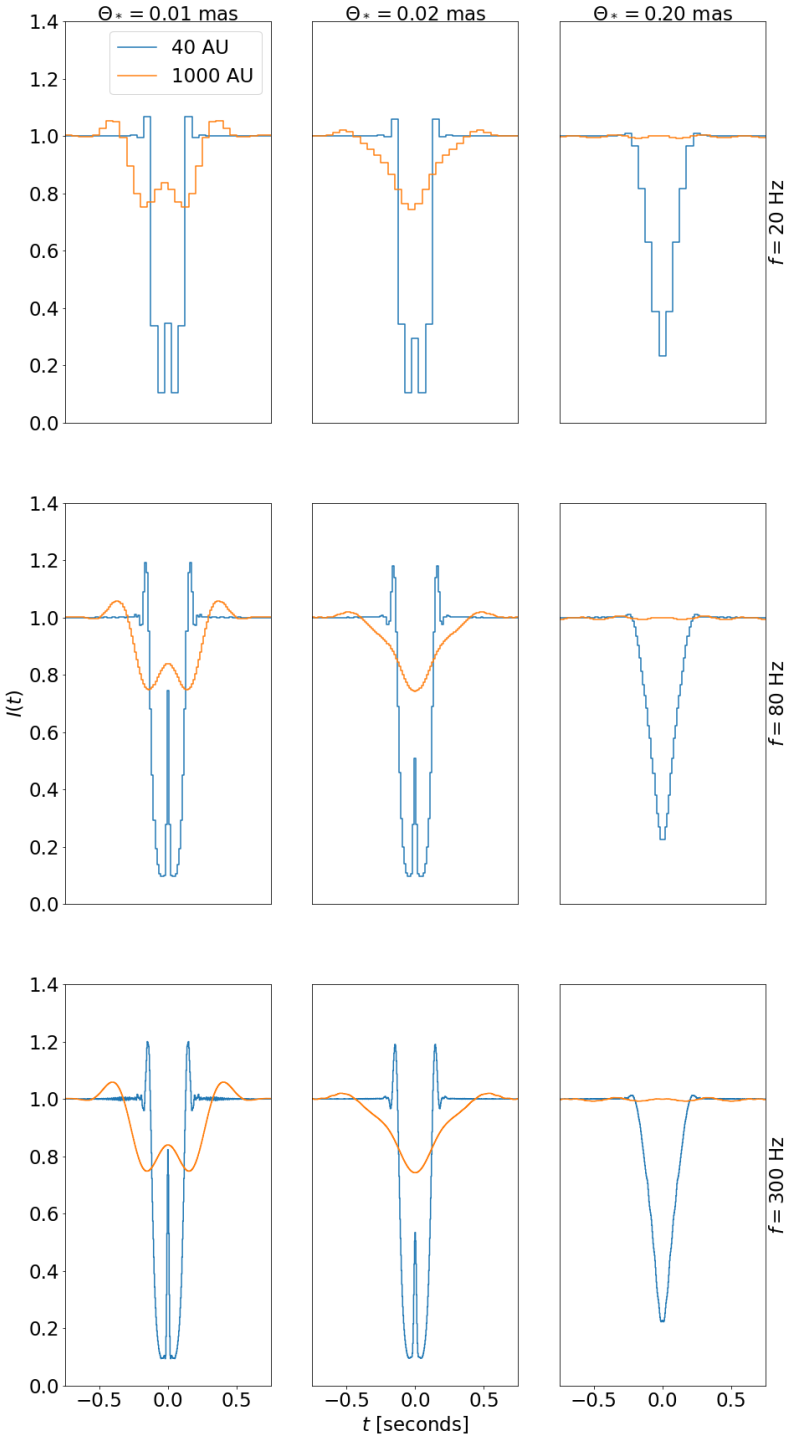


Figure 4.3: Noise-free occultation light curves that would be observed with the Kinetix with no filter for a circular object of radius 2.5 km at distances of 40 and 1000 AU, passing in front of an A0V background star, viewed at opposition. From left to right, the background star angular size is set to $\theta_* = 0.01, 0.02$ and 0.20 mas respectively. The impact parameter is set to $b = 0$ km. Each row corresponds to the fiducial sampling frequencies of $f = 20$, 80 and 300 Hz for the TAOS II mission, and the Sensitivy and Speed modes of the Kinetix camera used in this work.

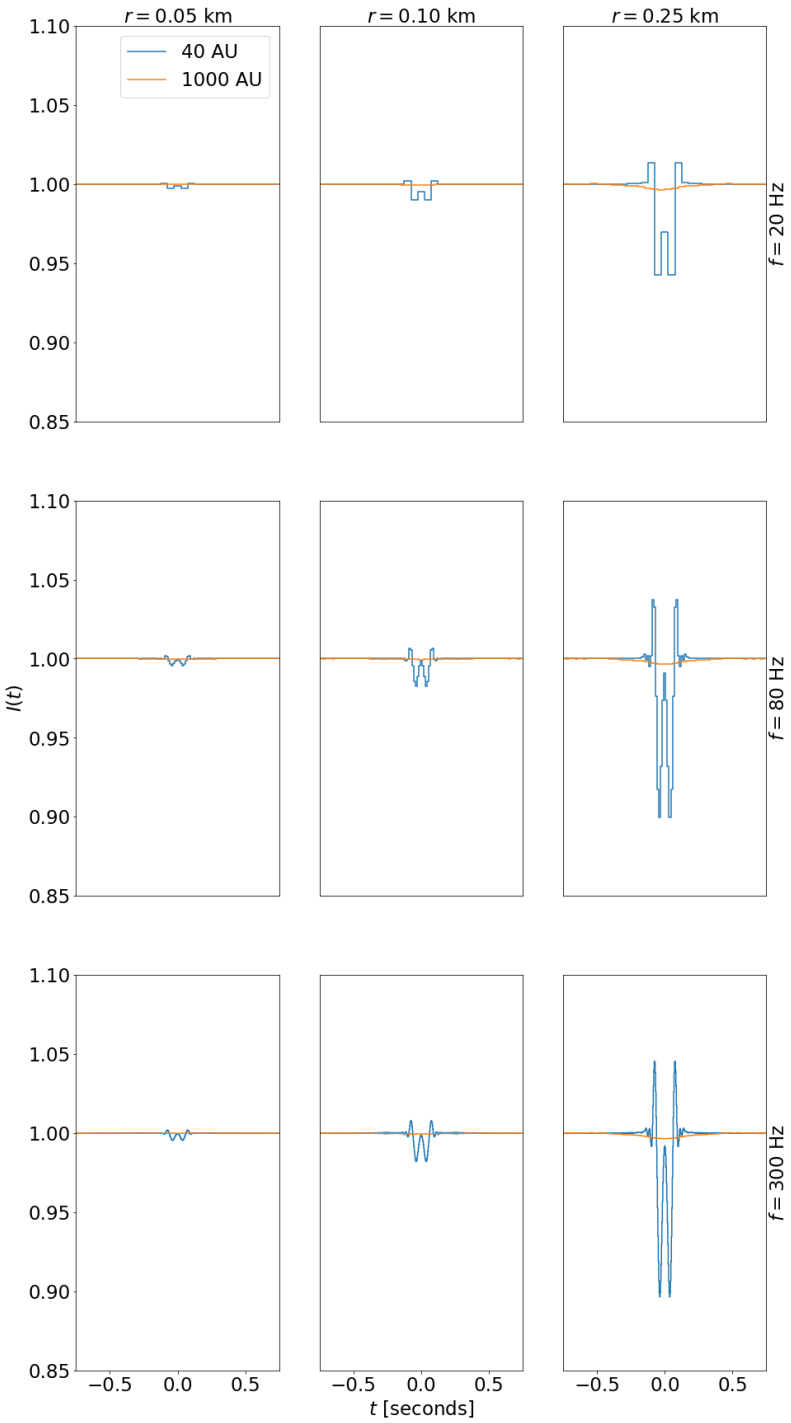


Figure 4.4: Noise-free occultation light curves that would be observed with the Kinetix with no filter for circular objects of radii 0.05, 0.1 and 0.25 km at distances of 40 and 1000 AU, passing in front of an A0V background star, viewed at opposition. The impact parameter and background star angular size are set to $b = 0$ km and $\theta_* = 0.02$ mas respectively. Each row corresponds to the fiducial sampling frequencies of $f = 20$, 80 and 300 Hz for the TAOS II mission, and the Sensitivy and Speed modes of the Kinetix camera used in this work.

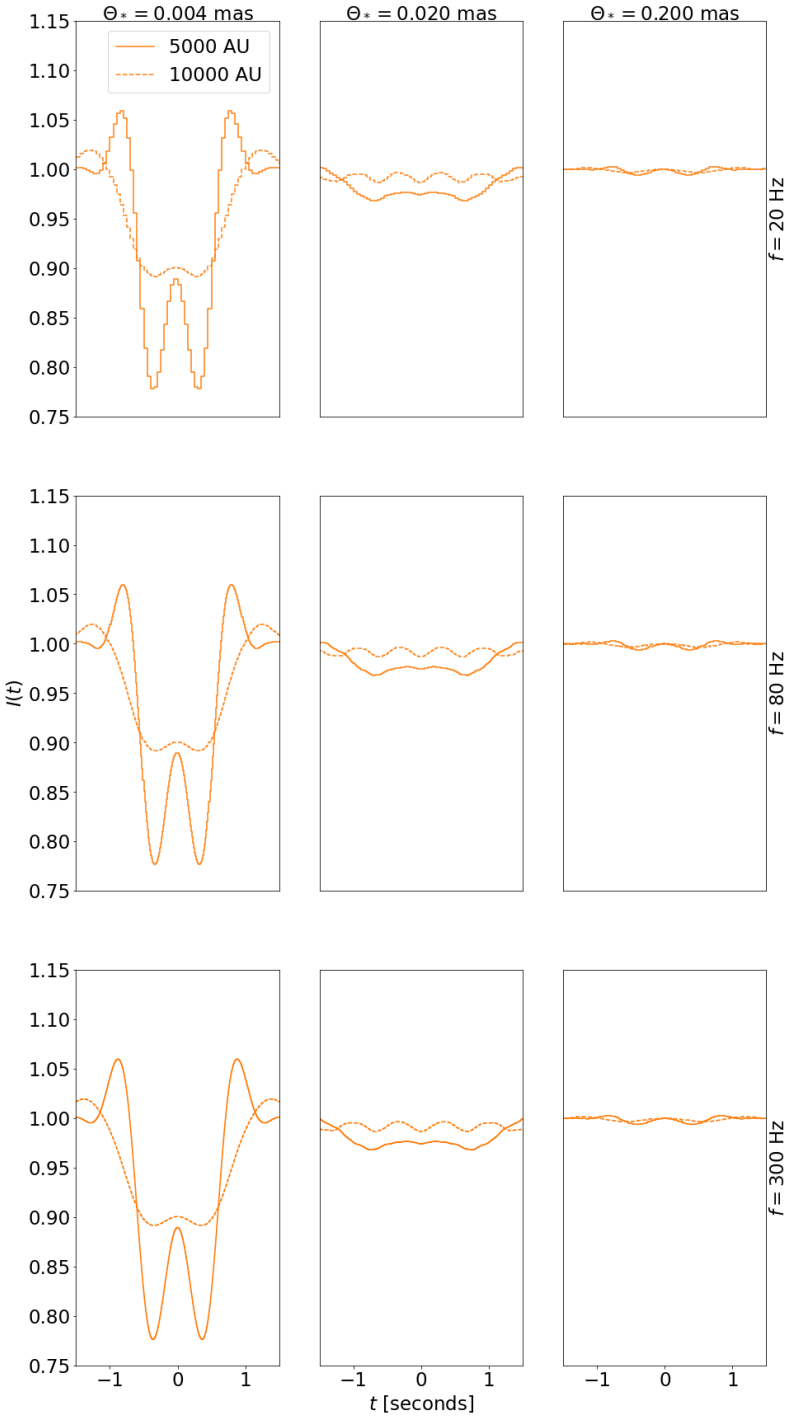


Figure 4.5: Noise-free occultation light curves that would be observed with the Kinetix with no filter for a circular object of radius 5 km at distances of 5000 AU and 10,000 AU, passing in front of an A0V background star, viewed at opposition. From left to right, the background star angular size is set to $\theta_* = 0.004, 0.02$ and 0.20 mas respectively. The impact parameter is set to $b = 0$ km. Each row corresponds to the fiducial sampling frequencies of $f = 20$, 80 and 300 Hz for the TAOS II mission, and the Sensitivy and Speed modes of the Kinetix camera used in this work.

can exploit the inherent parallelism in this image processing problem, and 2) Most pixels in the FoV are not particularly useful.

Firstly, an initial long integration "reference image" is obtained and saved to disk, from which a PSF model is inferred for the sake of determining aperture sizes and performing source detection. We model the PSF as an asymmetric 2D Gaussian, with standard deviations of ϕ_x and ϕ_y. For computational efficiency, we use a rectangular aperture for each star, and typically set the two side lengths to be $\sim 2.5 \times \phi_x$ and $\sim 2.5 \times \phi_y$. Next, the reference image is cross-correlated with the fitted PSF model to produce a point source "detection map", on which we run a peak finding routine to determine the source positions with an approach analogous to that of Lang & Hogg (2020) for the case of a single image. The detected sources are given IDs, and then ordered in descending order of brightness. Once the sizes and positions of the apertures are known, we cut out aperture “stamps” of the master flat and bias frames at these pixel positions. Next, a plot of the reference image is displayed on the GUI which allows the user to select several rectangular regions from which local sky values will be computed for each image. Once this is done, a plot of the reference image annotated with the sky apertures and background regions is displayed; an example is shown in Figure 4.6. The software then switches the Kinetix into live acquisition mode, and images are read out into a circular FIFO buffer, from which they are sequentially "popped" for processing, one image at a time.

For each popped image, we cut out the aperture stamps at the positions of the detected sources and the sky background regions (thus ignoring the vast majority of non-useful pixels). All pixels in the aperture stamps and sky regions are bias subtracted and flat corrected with their corresponding dark and flat stamps. Then, the calibrated pixel values in the aperture stamps are summed to give source flux estimates, and the median pixel value for each sky region is computed to give estimates of the local background levels. Arrays of these numbers and their associated IDs are then saved to disk, and the image is removed from computer memory forever. In this way, we are saving MBs of photometric data to disk rather than GBs of imagery, and so can comfortably observe many 1000s of stars over several nights.

Crucially, since operations on a particular stamp or sky region are independent of all others, we can do this processing in parallel. The PC used in this work was equipped with an 18 core Intel Xeon W-2295 processor[6], and for most targets observed in this work, the mean image

[6]We cannot use dual processor systems for this work, since access of memory held by the other processor can significantly throttle the data rates possible with the Kinetix.

processing time was $\sim 1-2$ ms per image. The PC was also equipped with 64GB of DDR4 RAM, allowing us to easily store ~ 100 images in the FIFO buffer at any time.

In order to assess the telescope guiding and quality of the incoming data, plots of a couple of exemplar apertures and their associated light curves are generated and displayed on the GUI every 5 seconds or so. To generate these figures quickly, we use 'blitting'[7] to cache all parts of the plot that remain the same in each figure (e.g. the axis labels, grid lines and whitespace etc.), such that it takes only $\sim 10-100$ ms to render the figure. An example display from this real time plotting is shown in Figure 4.7. Finally, while the software runs, time-stamped housekeeping information is saved to a log file to record the positions and sizes of the sky and aperture stamps, the camera exposure time and mean image processing times. A configuration file for a given data run is also saved, which records additional information such as the names of the calibration frames used in the reduction, and the subset of brightest sources used to determine the PSF model.

Having saved aperture flux measurements of stars in the FoV to disk, we can proceed to analyse the data for candidate occultation events offline by first correcting for systematic noise in the light curves, and then applying a matched filtering procedure to sift through the corrected data to detect events.

The software was written in Python, and makes grateful use of the following packages: PyVCAM[8], Numba, Numpy and Matplotlib (Lam et al., 2015; Harris et al., 2020; Hunter, 2007).

4.5 Observations at McDonald Observatory

We trialled our data system with the Kinetix camera on the 2.1 m telescope at McDonald Observatory, Texas, over 7 nights between the 16 - 22 September 2022. The Kinetix is 3200×3200 pixels, with a plate scale of 0.08"/pixel with a focal reducer lens at the Cassegrain focus, and so has a FoV of 4.3'. To maximise the signal-to-noise of the data, we observed with no filter. In total, we observed for more than 35 hours over the 7 nights, resulting in ~ 30 GB of star light curves and sky background measurements of bright open clusters. Open clusters were chosen as the bright main sequence stars within them have an angular size an order of magnitude smaller than red giants at a comparable distance, which are the brightest stars populating globular clusters, and would smooth out the occultation signature for all but the

[7] https://matplotlib.org/stable/tutorials/advanced/blitting.html
[8] https://github.com/Photometrics/PyVCAM

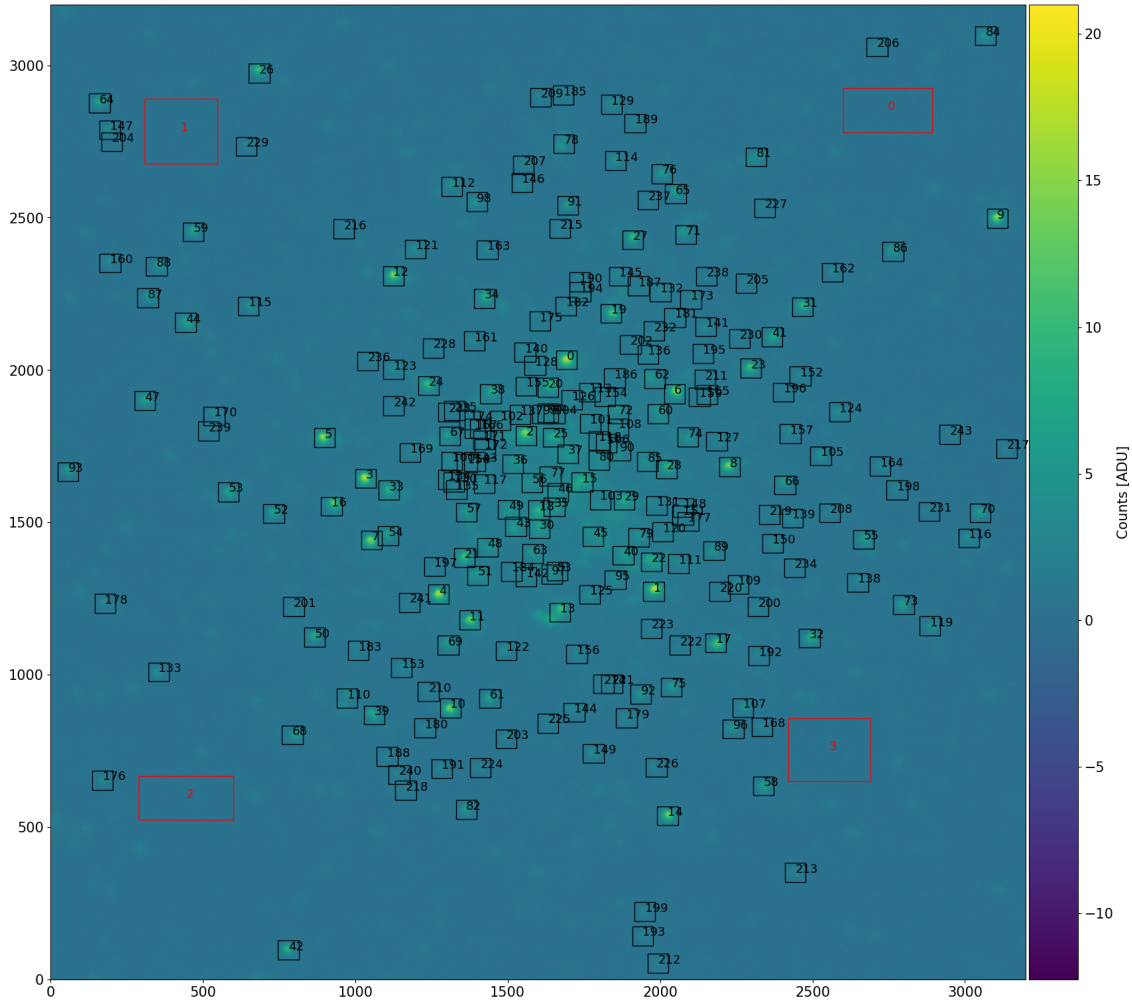


Figure 4.6: Example of a reference image annotated with the (black) source aperture stamps and (red) regions in which to compute sky background levels. The image is of M2, taken in high-speed mode with the Kinetix camera on the McDonald 2.1 meter in September 2021.

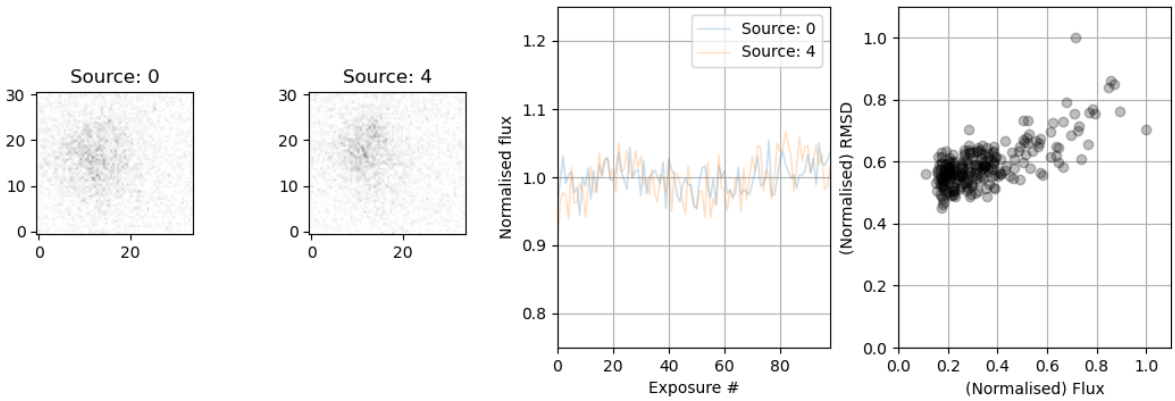


Figure 4.7: The real time plots consist of up to four panels; during the continuous live acquisition, this is updated once every ~ 5 seconds. The first two panels are bright, well-separated aperture source stamps from the most recently acquired image; axes indicate pixels. We also include the option to time-average over many aperture source stamps if the targets are faint. Data are from sources 0 and 4 from Figure 4.6. The third panel shows the normalized light curves from these two stamps for the last 100 images. The final panel shows the light curve root-mean-square deviation (RMSD) as a function of source flux for all detected sources over the last 100 images. Collectively, this information helps us with guiding the telescope and assessing the choice of aperture size.

largest objects (see Figure 4.3). Because the length of an occultation is determined mostly by the effective speed that the Earth passes at right angles through the shadow, we observed clusters that were as close as possible to the line tangent to the Earth's orbit; this effectively slows the speed that the Earth passes through the shadow, and thus lengthens the occultation time.

The targeted clusters, and an inventory of the observations, are recorded in Tables 4.2 and 4.3 respectively. The frame rate we record in Table 4.3 is our software's estimate of the true real time data processing rate. Note that for 3 ms exposures (i.e. when the Kinetix was in Speed mode) we do not achieve the theoretical maximum of 333 Hz. This was found to be due to imperfections in the optic fibre cables connecting the Kinetix to the PC that we couldn't completely remove. Regardless, even when this problem was at its worst between 18 - 21 September, the frame rate still exceeded 300 Hz.

Master dark frames encoding the pixel bias patterns and dark currents for the Speed and Sensitivity modes were obtained by averaging 1000 frames with no light on the camera, taken at the same frame rate as the data. A similar series of sky flats were obtained and then median averaged immediately after sunset with the telescope tracking off, and slewing the telescope

Object	RA	Dec	Distance [kly]	Apparent Magnitude [V]	Apparent Dimensions [V]
M11	18:51:05	-06:16:12	6.1	5.8	22.8’
M37	5:52:18	33:32:02	4.5	6.2	24’
NGC884	02:22:0	57:08	7.6	3.8	30’
NGC6811	19:37:17	46:23:18	3.6	6.8	13’
NGC7788	23:56:45	61:23:59	7.8	9.4	5’

Table 4.2: The open clusters targeted for these observations. In addition to hosting many bright, main sequence stars, these targets were chosen as they were fairly close to the line tangent to the Earth's orbit at the time of observing.

N-S while taking data in the two modes with the exposure time set to give an average pixel value in each frame of approximately 50% of the maximum pixel value.

4.5.1 Correcting systematic noise

For easy reference in what follows, Table 4.1 provides definitions of the symbols introduced in this chapter pertaining to our approaches to correcting our raw photometry and flagging candidate occultation events in the data. We define a "data set" as being the set of aperture photometry light curves corresponding to a single row of the observation inventory shown in Table 4.3.

The raw aperture flux time series of stars in our data sets are affected by correlated noise. This arises due to sub-second tip-tilt wavefront motions, where the speckle pattern PSF wanders within the fixed source aperture. Since this noise is systematic – that is, common to all stars in the FoV to some greater or lesser extent – we can use their collective behaviour to correct this effect. Each data set consists of $n = 1, ..., N$ stars, each with $t = 1, ..., T$ concurrent flux measurements, and stars are ordered by brightness such that $n = 1$ is the brightest and $n = N$ is the faintest. Typical data sets consist of $10^7 - 10^9$ individual flux measurements, so as a first approach, we use a computationally inexpensive “classical” comparison star approach, where we average over the time series of a subset of stable stars.

The tip-tilt induced noise is common to all light curves to some greater or lesser extent (see the raw aperture flux measurements in Figure 4.8). For the purposes of this work, it is convenient to work in relative brightness rather than ADU flux measurements (e.g. for making decisions about what star light curves are amenable to detecting occultation events of a given depth, what star light curves should be used as comparison stars etc.). Figure 4.9 shows

Object	Date	Exposure time [ms]	Duration [Hrs]	Detected stars	Frame rate [Hz]
M11	2022-09-16	12	0.25	373	83
M11	2022-09-16	12	3.5	468	83
NGC884	2022-09-16	12	3	73	83
M37	2022-09-16	12	0.4	55	83
M11	2022-09-17	3	0.15	198	322
M11	2022-09-17	12	2	326	82
M11	2022-09-17	3	1	313	328
M11	2022-09-17	3	0.75	291	328
NGC884	2022-09-17	12	0.15	56	82
M11	2022-09-18	3	0.3	292	301
M11	2022-09-18	12	1	421	83
M11	2022-09-19	12	1.3	642	83
M11	2022-09-20	12	0.5	296	83
M11	2022-09-20	12	1	282	83
M11	2022-09-20	3	0.15	149	301
M11	2022-09-20	3	1	222	301
M11	2022-09-20	3	1	269	301
M37	2022-09-20	12	1.8	87	83
NGC884	2022-09-20	12	0.4	58	83
NGC7788	2022-09-20	12	2.65	78	83
M11	2022-09-21	12	0.8	185	83
M11	2022-09-21	3	2.8	135	301
M11	2022-09-21	3	0.25	158	301
M11	2022-09-21	3	0.1	118	301
NGC884	2022-09-21	3	1.5	35	301
NGC884	2022-09-21	3	2.5	41	301
NGC884	2022-09-21	3	1	35	301
M11	2022-09-22	12	1.5	125	83
M11	2022-09-22	3	1	16	330
M11	2022-09-22	3	1	23	330
NGC6811	2022-09-22	12	0.65	28	83

Table 4.3: Observation inventory 16th - 22nd September. 3 and 12 ms exposures were recorded in the Kinetix's Speed and Sensitivity modes respectively.

Symbol	Definition
n	Star number in a data set
N	Total number of stars in a data set
t	Exposure number in a data set (proxy for time)
T	Total number of exposures in a data set
m	Comparison star number in data set
M	Total number of comparison stars, where $M < N$
$F_{\mathrm{Raw},n}$	Aperture photometry flux in ADU of star n
$I_{\mathrm{Raw},n}$	Uncorrected relative brightness of star n
w_m	Weight of comparison star m
$\bar{I}_{\mathrm{Sys}}$	Estimate of the systematic noise in a data set
$I_{\mathrm{Corr},n}$	Corrected relative brightness of star n
H	Matched filter template
f	Frequency
S	Unnormalised matched filter signal
σ	Matched filter signal-to-noise ratio

Table 4.4

the same data in Figure 4.8, but each time series has now been normalised w.r.t its median brightness i.e. $I_{\mathrm{Raw},n}(t) = F_{\mathrm{Raw},n}(t)/\mathrm{Median}[F_{\mathrm{Raw},n}]$. It is clear that the tip-tilt induced noise affects all stars in a similar way in terms of their relative brightness, and we can compute the time-wise weighted average to describe this behaviour using some subset of M (i.e. $M < N$) stable comparison stars,

$$\bar{I}_{\mathrm{Sys}}(t) = \sum_{m=1}^{M} \frac{1}{M}[w_1 I_{\mathrm{Raw},1}(t) + ... + w_M I_{\mathrm{Raw},M}(t)]\,, \tag{4.3}$$

with weights equal to the reciprocal of the standard deviation of the particular star's time series,

$$\frac{1}{w_m} = \sqrt{\frac{1}{T}\sum_{t=1}^{T}[I_{\mathrm{Raw},m}(t) - \bar{I}_{\mathrm{Raw},m}]^2}\,, \tag{4.4}$$

where $\bar{I}_{\mathrm{Raw},m}$ is the mean relative intensity of comparison star m[9]. The corrected flux

[9]N.B. $\bar{I}_{\mathrm{Raw},m}$ is not in general equal to 1, as the flux time series were normalised by their median, and not the mean,

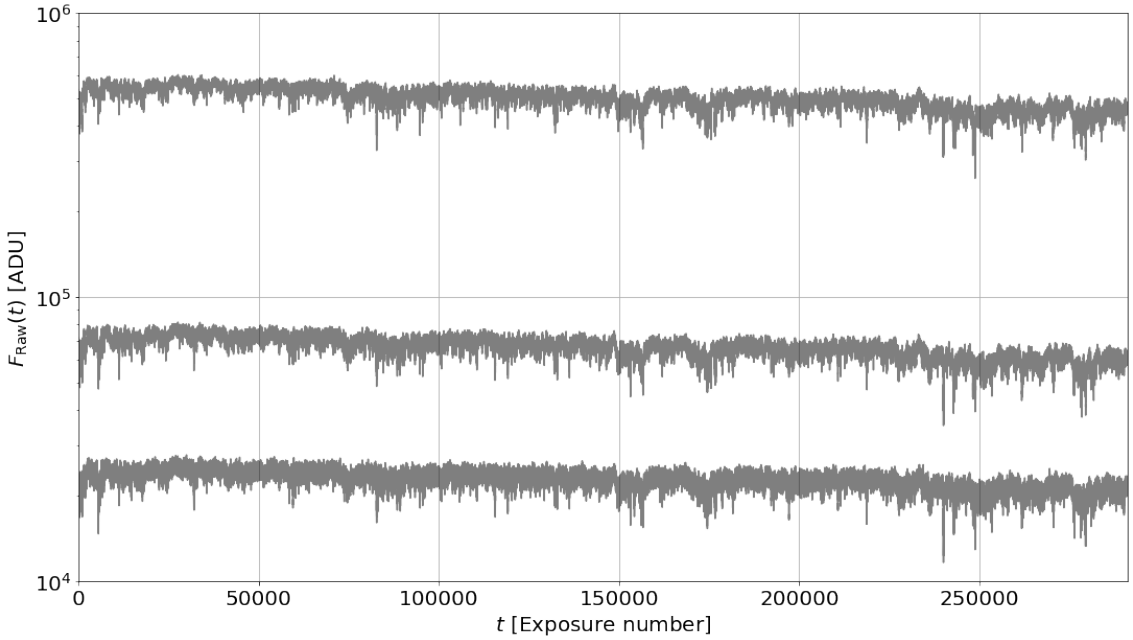


Figure 4.8: The raw aperture flux measurements in ADU of a few sources of different brightnesses in an example Sensitivity mode data set. Note the log scale of the y-axis.

measurements for some given star n at time t are then simply,

$$I_{\mathrm{Corr},n}(t) = \frac{I_{\mathrm{Raw},n}(t)}{\overline{I}_{\mathrm{Sys}}(t)} \,, \tag{4.5}$$

where n is not in the set of M stars used to compute $\overline{I}_{\mathrm{Sys}}(t)$.

In Figure 4.10 we show a comparison of the $I_{\mathrm{Raw},n}$ and $I_{\mathrm{Corr},n}$ time series for stars over a range of brightnesses, where the lighter coloured raw light curves have been vertically offset for clarity. This highlights the importance of the correction, since many features in the $I_{\mathrm{Raw},n}$ could be otherwise easily confused for candidate occultation events.

We plot the SNR of the corrected data light curves against the stellar magnitude from two example Sensitivity (circles) and Speed (crosses) mode runs targeting M11 in Figure 4.11[10]. We also plot the predicted SNR of the data for each run, using a noise model which includes estimates of the readout noise, dark current, photon shot noise and scintillation. The predicted scintillation noise, which is computed via the popular approximation given in Young (1967), clearly limits the achievable SNR at the bright end, and this limit is lower for shorter exposure

to guard against spurious measurements affected by e.g. cosmic ray hits

[10]The reference images for each of these runs were astrometrically calibrated with the *astrometry.net* tool (Lang et al., 2010), and the detected sources were cross-matched against the Pan-STAARS catalogue to obtain the r-band magntiudes (Magnier et al., 2013).

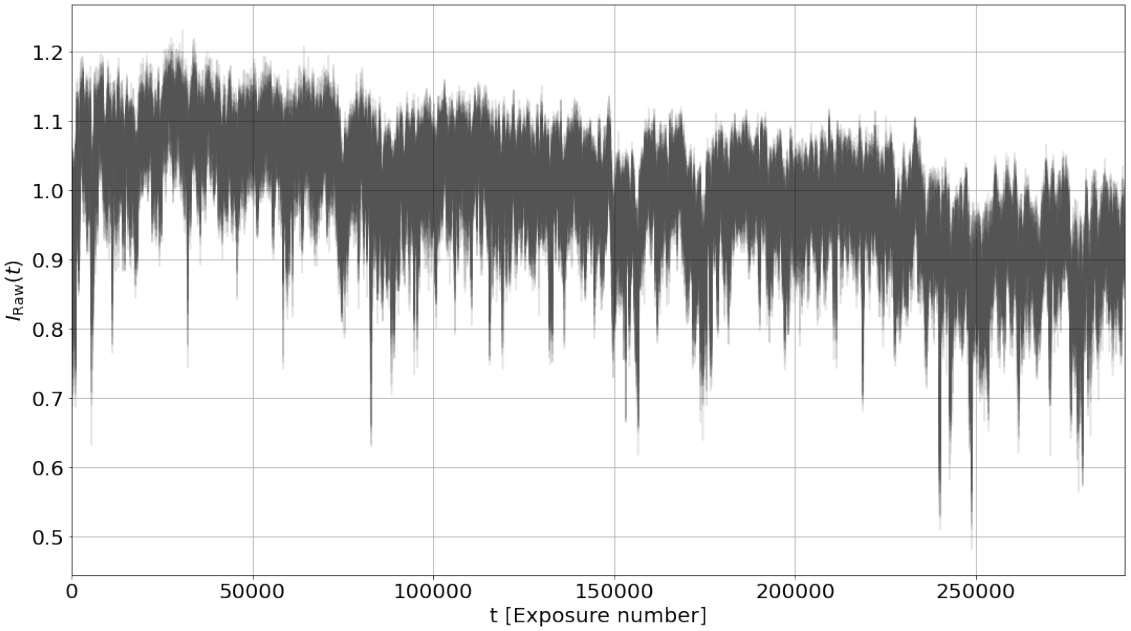


Figure 4.9: The data in Figure 4.8 where each star light curve has been normalised w.r.t its median flux measurement.

data. The corrected Speed mode data shows reasonable, albeit noisy agreement with the model prediction, but the SNR of the corrected Sensitivity mode photometry actually exceeds the theoretical limit at the bright end. While the data does appear to be scintillation limited as predicted by the noise model, this limit is more than 50% higher than expected. We are still investigating the source of this, though suspect it is a limitation of our simple photometric correction. Whether the scintillation noise is correlated amongst comparison stars or not, it is possible that by dividing each comparison star light curve by their collective weighted average, some of the noise proportional to the signal is removed. And as the brightest stars dominate the weighted average, we are in effect over correcting their raw time series, resulting in a higher than expected SNR.

We also plot the theoretical SNR predictions for TAOS II at 20 Hz and observations with the Kinetix on the MSE in both camera modes in Figure 4.11. From these SNR estimates, we generate some occultation light curves that would be observed for small objects occulting a fiducial r magnitude 14 star – of stellar type A0V with an angular size of 0.02 mas – by TAOS II and our planned mission to mount the Kinetix on the MSE. These are shown in Figure 4.12. The benefits in sampling frequency and SNR of our planned project over TAOS II for detecting the small, but presumed to be numerous objects of radius < 1 km are evident. TAOS II's aim is to detect occultations for KBOs down to 0.5 km diameter - this is certainly possible for brighter targets, but for the more numerous faint stars it is extremely challenging, and

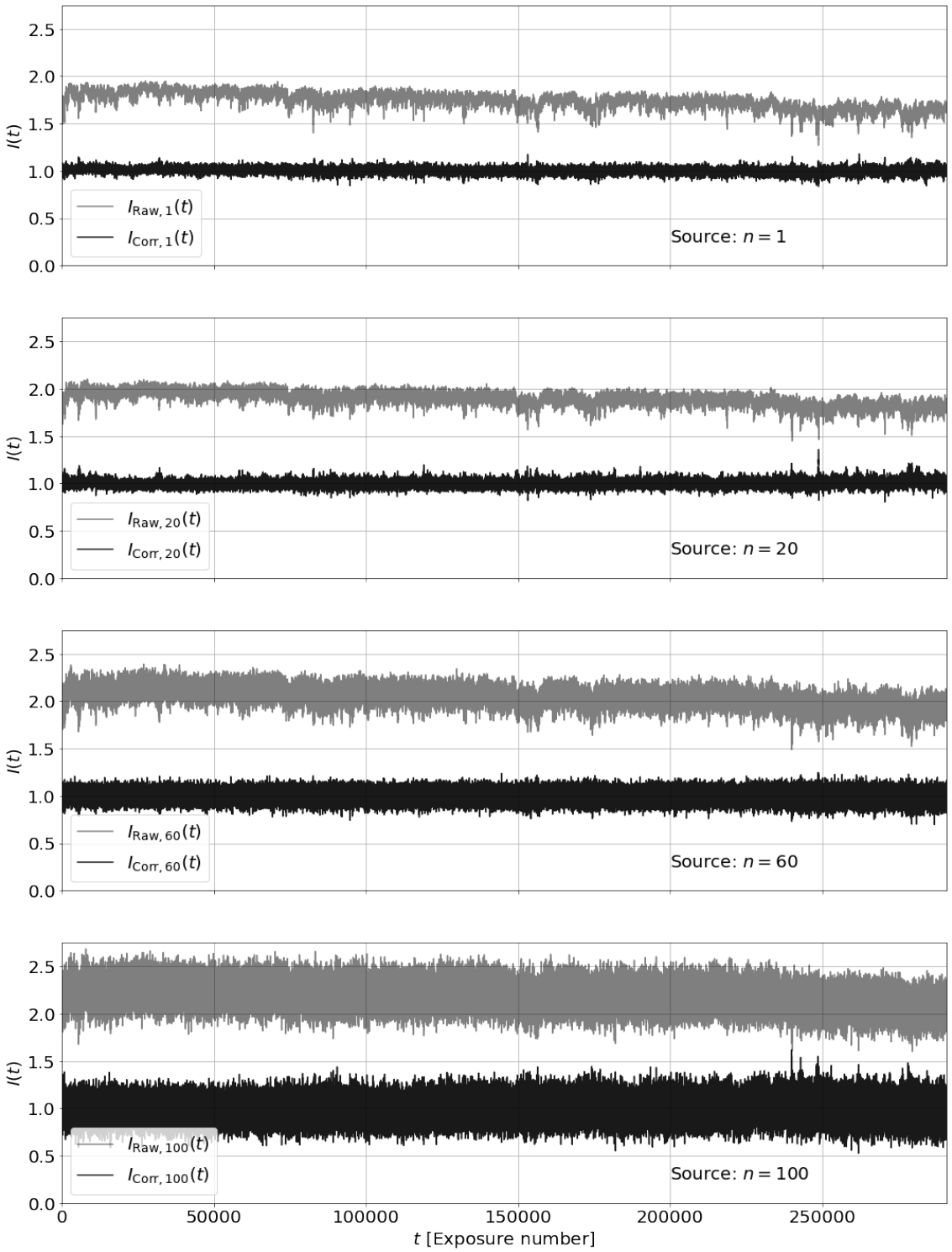


Figure 4.10: A comparison of the raw and corrected relative brightnesses of some star light curves in a data set. The lighter coloured raw light curves are vertically offset for clarity.

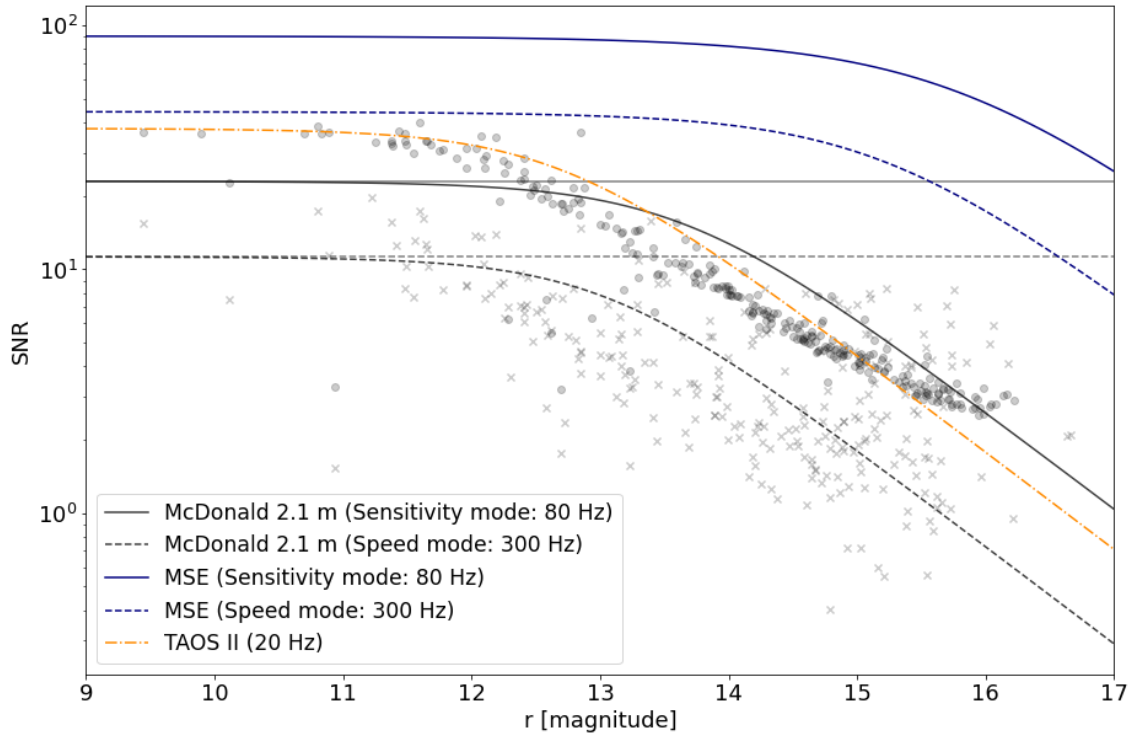


Figure 4.11: Stellar r-band magnitude vs SNR for stars in M11 in typical Sensitivity (circles) and Speed (crosses) mode datasets from our observations on the 2.1 m at McDonald Observatory. We plot the predicted SNR of stars across this brightness regime for our observations on the 2.1 m, the upcoming TAOS II mission and our planned observations on the 11.25 m Maunakea Spectroscopic Explorer (MSE). The noise model predictions include appropriate estimates of the readout noise, dark current, photon shot noise and scintillation. Scintillation noise clearly limits the achievable SNR at the bright end for these short exposure time series, and horizontal lines are marked on the plot showing where this limit should exist for the Sensitivity (undashed) and Speed (dashed) observations.

effectively impossible when coupled with the slower 20 Hz sampling frequency, as shown in the top left panel of Figure 4.12. In comparison, such events could be clearly resolved by Kinetix observations on the MSE, allowing us to probe objects that the TAOS II mission is not optimised to detect. This is true for more distant occulting objects too, as although the duration of these occultation events is quite long, the dips in brightness are nonetheless shallow, and benefit from both the improved sampling frequency and SNR of mounting a Kinetix on the MSE. Indeed, Figure 4.12 suggests we would have a good chance of detecting an occultation of an object with a 1 km radius at 1000 AU, while such events would be invisible to TAOS II for all but the brightest of background stars. However, Figure 4.12 was generated with the optimistic assumption that we would have perfectly corrected the systematic noise sources in the light curve data. This motivates experimenting with more advanced methods to the simple, first approach used here.

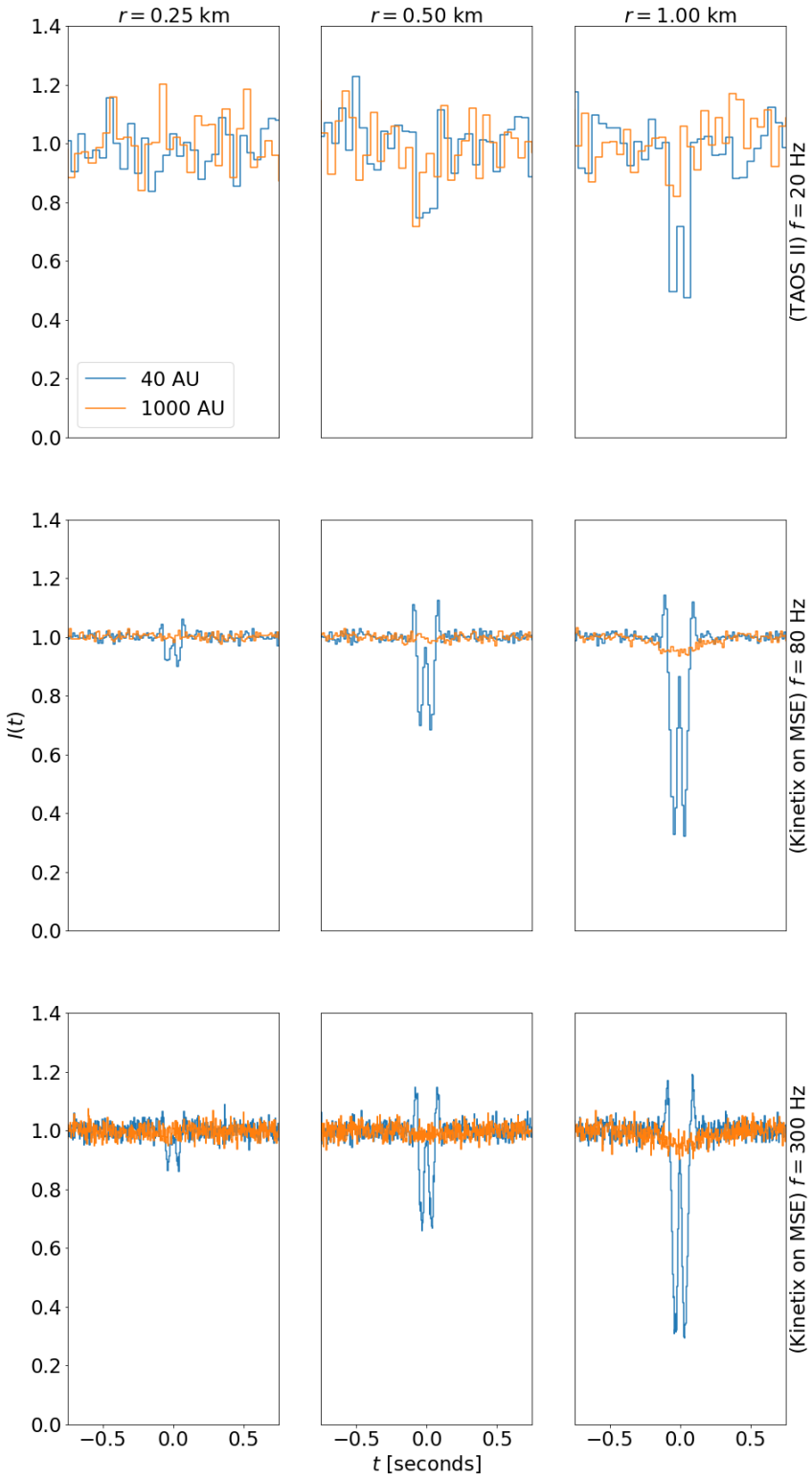


Figure 4.12: Occultation light curves for 3 small objects of $r \leq 1$ km at distances of 40 AU (blue) or 1000 AU (orange) passing in front of a r-band magnitude 14 star of spectral type A0V, with an angular size of 0.02 mas. The top row shows how these events would appear in TAOS II data, and the middle and bottom rows show how the same occultation signals would appear in Kinetix observations on the 11.25 m Maunakea Spectroscopic Explorer (MSE). Gaussian white noise was added to the simulated events based on the predicted SNR estimates in Figure 4.11 for each situation.

4.5.2 Occultation Event Detection

We adopt a matched filtering approach to detect candidate occultation events in the corrected light curves. There is a huge amount of data to sift through, and because flagged candidate events will require by-eye review for validation, the following decisions were made to produce a manageable number of events for human inspection.

As outlined in Section 4.3, the occultation light curves produced by small solar system objects are dependent on several parameters, and while the variety of possible signals is diverse, very different parameter configurations can result in similar observed signals in the data. This complicates the generation of the template bank for matched filtering, and so like Huang et al. (2021), we attempt to simply detect U-shaped dips in the data. These templates have only two parameters; the depth d (in relative intensity), and duration Δt (in number of exposures as a proxy for time). In this work we trial templates with $d = 0.1 - 1$ at intervals of 0.1. Then, depending on the Kinetix camera mode, we trial $\Delta t = [2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 15, 17, 18, 19, 20, 30, 40, 50, ..., 500]$ (i.e. 10 exposure intervals between 20 - 500) for Sensitivity mode, and $\Delta t = [2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 15, 17, 18, 19, 20, 40, 60, 80, ..., 2000]$ (i.e. 20 exposure intervals between 20 - 2000) for Speed mode. In this way we generate two banks consisting of templates for Sensitivity and Speed mode observations. We show below that in practice, adopting these simple approximations to real, complex occultation signals is comparably effective to using a more accurate template.

We start with a given template $H(t)$ (padded to be the same length as the data) and corrected data $I_{\mathrm{Corr},n}(t)$, and transform these into the frequency domain, by taking their Fourier transforms e.g. $H(f) = \mathscr{F}[H(t)]$. We then compute the unnormalised matched filter signal in the time domain as

$$S_n(t) = \mathscr{F}^{-1}[H^*(f) I_{\mathrm{Corr},n}(f)]\,, \tag{4.6}$$

where $*$ denotes the complex conjugate. We compute a robust, empirical estimate of the matched filter signal-to-noise ratio (SNR) timeseries as

$$\sigma_n(t) = \frac{S_n(t)}{\kappa \times \mathrm{MAD}[S_n]}\,, \tag{4.7}$$

where $\mathrm{MAD}[S_n]$ is the median-absolute-deviation of the unnormalised matched filter signal in the time domain for source n. We set $\kappa = 1.4826$, which is appropriate for approximately normally distributed data (refer to Equation 2.34). If $\sigma_n(t)$ exceeds some threshold, it can be used to flag a candidate occultation event in the light curve of star n at time t.

To test the efficacy of this procedure for detecting candidate occultation events, we injected simulated occultation signals into the time series of real data, and attempted to detect the events by matched filtering using both the exact template that generated the signal and a simple U-shaped template. This approach means we do not have to make any assumptions about the noise in the data. The results are shown in Figures 4.14 - 4.16. All data were corrected using the approach described in Section 4.5.1, and for this particular data set, $M = 71$ comparison stars were used, which were chosen on the basis that the standard deviation of their uncorrected normalised relative intensity light curves was less than 0.1. The $\sigma_n(t)$ time series shown are those corresponding to the template in the bank which gave the highest peak.

As should be expected, for occulting objects in the Oort Cloud, the best matching U-shaped template results in a comparable detection significance to using the true occultation signal as there are less pronounced fringe diffraction effects associated with these signals. There is however reduced efficiency when using U-shaped templates for KBOs, but as seen in Figure 4.15, these simple U-shaped templates can still confidently detect even modest KBO occultation events. Indeed, this approach can still recover even very shallow KBO associated dips (see Figure 4.16).

4.6 Results and Discussion

For each data set in Table 4.3, we applied corrections to the raw aperture flux measurements via the method outlined in Section 4.5.1. The set of M stable comparison stars for a given data set was chosen as all stars whose uncorrected relative brightness standard deviation was less than 0.1. This typically provided a sample of 10s of bright stars, but in the event one or less stars met this condition, using the top 10 brightest stars was found to be a useful heuristic. From these stars, an averaged description of the light curve systematics was computed (Equation

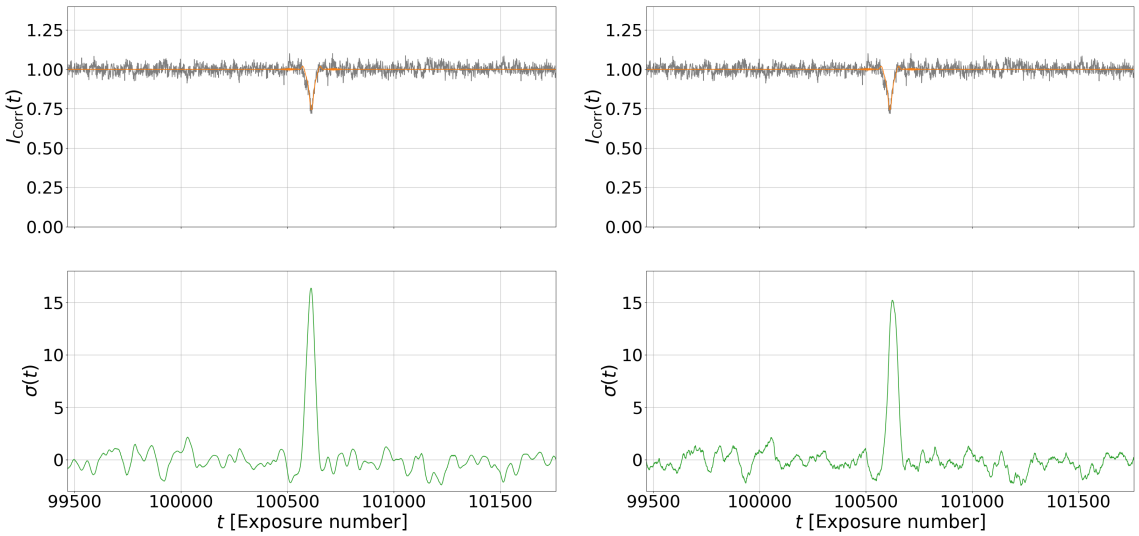


Figure 4.13: [Top] Simulated occultation event for a 2500 m radius object at 1000 AU injected into the real time series of a star in an example Sensitivity mode data set. Other occultation parameters are $b = 0$ m, and $\theta_* = 0.02$ mas, and the star is assumed to be of spectral type A0V. The data has been corrected using the approach outlined in Section 4.5.1. [Bottom] The matched filter signal-to-noise ratio time series of the (left) true occultation signal and (right) best-matching U-shaped template.

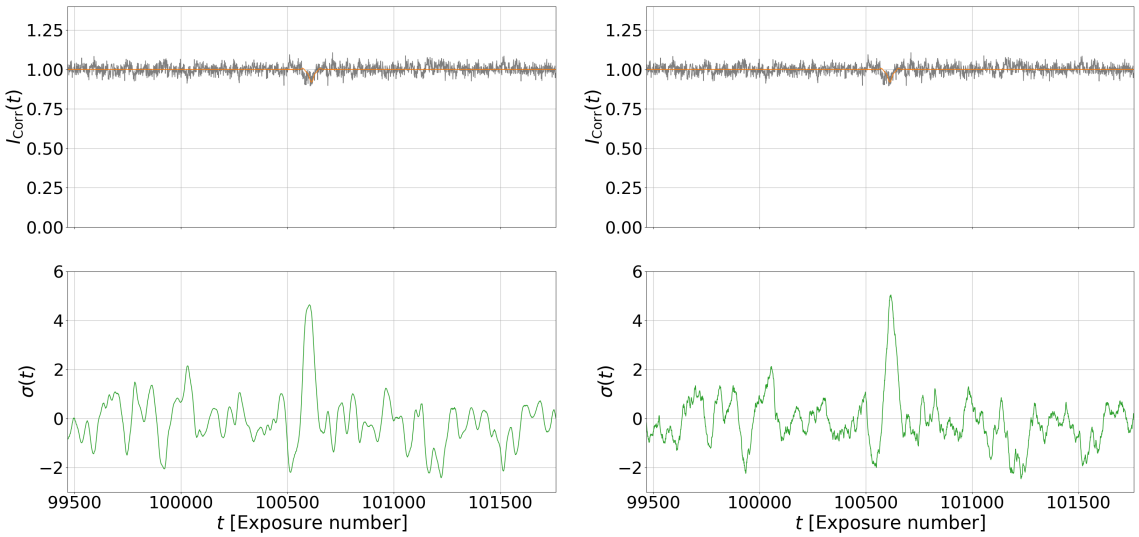


Figure 4.14: [Top] Simulated occultation event for a 1250 m radius object at 1000 AU injected into the real time series of a star in an example Sensitivity mode data set. Other occultation parameters are $b = 0$ m, and $\theta_* = 0.02$ mas, and the star is assumed to be of spectral type A0V. The data has been corrected using the approach outlined in Section 4.5.1. [Bottom] The matched filter signal-to-noise ratio time series of the (left) true occultation signal and (right) best-matching U-shaped template.

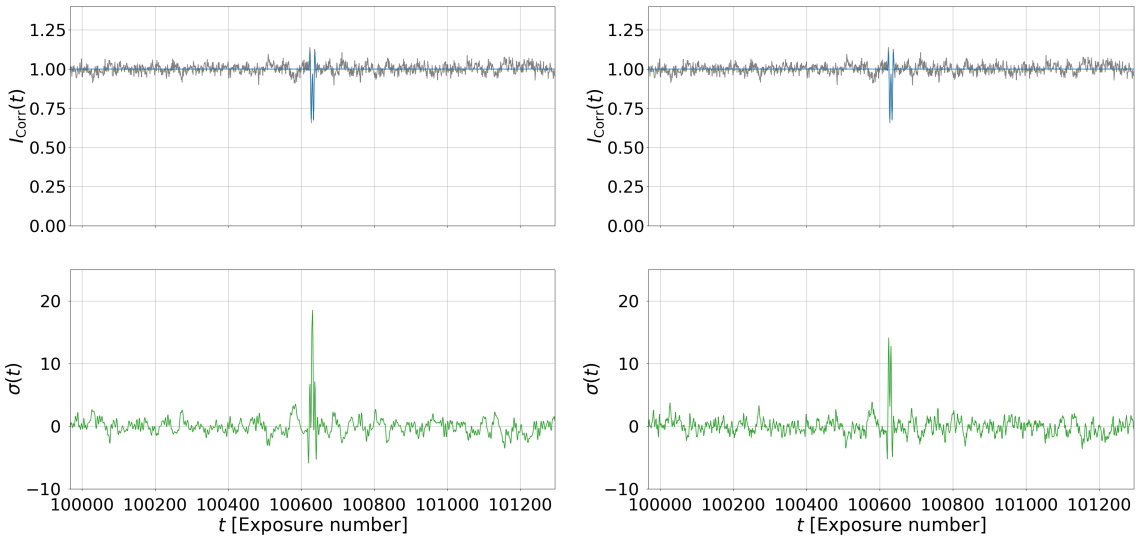


Figure 4.15: [Top] Simulated occultation event for a 500 m radius object at 40 AU injected into the real time series of a star in an example Sensitivity mode data set. Other occultation parameters are $b = 0$ m, and $\theta_* = 0.02$ mas, and the star is assumed to be of spectral type A0V. The data has been corrected using the approach outlined in Section 4.5.1. [Bottom] The matched filter signal-to-noise ratio time series of the (left) true occultation signal and (right) best-matching U-shaped template.

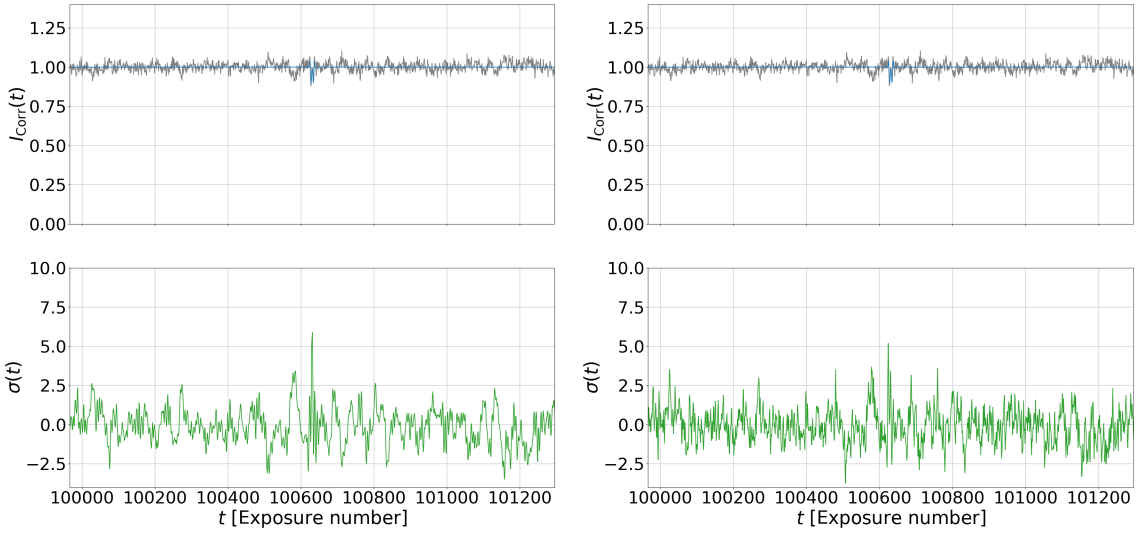


Figure 4.16: [Top] Simulated occultation event for a 250 m radius object at 40 AU injected into the real time series of a star in an example Sensitivity mode data set. Other occultation parameters are $b = 0$ m, and $\theta_* = 0.02$ mas, and the star is assumed to be of spectral type A0V. The data has been corrected using the approach outlined in Section 4.5.1. [Bottom] The matched filter signal-to-noise ratio time series of the (left) true occultation signal and (right) best-matching U-shaped template.

4.3) to correct all N light curves (Equation 4.5).

4.6.1 No detection of real occultations

We next applied the matched filtering approach to the corrected light curves (as described in Section 4.5.2). For either the Sensitivty or Speed mode data, matched filtering the respective template bank against a single light curve usually takes $\sim 10-100$ seconds, and so to save processing time, all corrected light curves whose SNR was less than 3.3 were ignored from this part of the analysis, as these are too noisy to reliably detect an occultation event. For all remaining corrected light curves, we computed the matched filter SNR time series (Equation 4.7) using the templates in the banks appropriate to the sampling frequencies of either the Sensitivity or Speed modes of the Kinetix. Also, we manually examined all corrected light curves by eye to rule out the presence of any obvious, deep dips in the data.

The threshold for a candidate event detection in light curve n at time t was set to be $\sigma_n(t) > 5$. This decision was made after some experimentation, and is a tradeoff between bogging the by-eye verifier down with having to sift through many false positives, while remaining sensitive to modest, potentially real events. In order to guard against multiple detections of the same event in a given n light curve, we divide each light curve into 1000 exposure "windows", and if multiple peaks pass the detection threshold we record only the largest value of $\sigma_n(t)$ and the associated template i.e. that template that best matches the feature in the data in this part of the light curve.

Next, to further reduce the number of the candidates to a manageable size prior to by-eye inspection, if multiple $\sigma_n(t)$ peaks above the detection threshold are identified within 50 exposures of each other in different light curves, these peaks are rejected. This guards against false positives associated with correlated noise that is not fully removed by our correction procedure. This leaves us with a set of isolated $\sigma_n(t)$ peaks that are not seen in any of the other light curves above the detection threshold.

We then sift through the returned events by eye. As part of the human scanning, we checked figures of the candidate event light curve plotted alongside a nearby bright star as a comparison. If the two light curves show similar behaviour in the vicinity of the supposed dip, the event was flagged as spurious and ignored.

Having applied the above procedure to both the Sensitivity and Speed mode data sets, we

did not find any candidate occultation events. This is to be expected given the short length and small coverage of this pilot study. For example, the first TAOS survey, after 7 years of dedicated observations of rich star fields close to the ecliptic, has also reported no detections (Bianco et al., 2010; Zhang et al., 2013).

4.6.2 Computing our Detection Efficiency with Simulated Events

We estimate the detection efficiency of our candidate event detection algorithm by injecting simulated occultation signals into the data, and computing the detection efficiency as a function of the occultation event parameters and the signal-to-noise ratio of the data. This can be used to assess the efficacy of our data correction and event detection procedures, and ultimately can be used to compute upper bounds on object abundances.

As the parameter space of an occultation event is large, we choose to fix some parameters for these simulations. The sampling frequency and wavelength range can be uncontroversially fixed; the former is dependent only on the Kinetix mode (i.e. $\sim$ 300 Hz in Speed mode and $\sim$ 80 Hz Sensitivity), and the latter can be simulated by integrating over the Kinetix response curve convolved with spectrum of a type A0V main sequence star. This stellar type is typical of the bright main sequence stars in the targeted open clusters, and we similarly fix the angular size to 0.02 mas, which is appropriate for an A0V star at the distance of M11, for which we have by far the most data. The two parameters most important for constraining the size and distribution of Oort cloud objects are object radii and distances. We inject objects of radii $r = 0.25, 0.5, 1, 2.5, 5, 7.5, 10$ km at distances of $a = 30, 40, 50, 100, 250, 500, 750, 1000, 2500, 5000, 7500, 10000$ AU. We also vary the impact parameter, $b = 0, 1, 2.5, 5, 7.5, 10$ km and the opposition angle, $\phi = 0, 25, 50, 75$ degrees. We then uniformly sample the time at which to inject an occultation event into a given light curve. We inject exactly one simulated event into each and every light curve whose corrected SNR > 3.3. In total, we injected 84048 simulated occultation events into the Sensitivty mode data set, and 38510 events into the Speed mode data. We note that injecting signals in this way slightly overestimates the Poisson noise of an event, since we do not modify the data noise that would be associated with the reduction in flux during an occultation, and so our reported detection efficiencies should be considered to be conservative.

We make a couple of passes over the Speed and Sensitivity mode data sets in Table 4.3, where we uniformly sample the occultation parameters above, generate the noiseless occulta-

tion light curve, and inject that into the raw flux measurements associated with a given source. We process the data in exactly the same way as in Section 4.6.1, which returns a list of isolated, significant $\sigma_n(t)$ peaks for each n light curve in each data set. If one of these local peaks is within 500 exposures of the mid-point of an injected occultation, then we consider that to be a successful detection. We show some examples of occultation events successfully recovered by our algorithm in both the Sensitivy and Speed mode data in Figures 4.17 and 4.18 respectively.

We compute the detection efficiency as the number of recovered events divided by the sum of the total number of injected events. The detection efficiency as a function of event parameters is shown in Figure 4.19 for both the Speed and Sensitivity mode data sets. As to be expected, the larger the occulting object and the closer it is to the Earth, the greater the likelihood to successfully detect the event. The larger the impact parameter, the shallower the dip, and we indeed see that the detection efficiency drops off slightly. The likelihood of detecting an event is only weakly correlated with the opposition angle, but in both camera modes the best detection efficiencies are associated with the largest angle. The opposition angle only affects the duration of the event – with the greater the angle the longer the duration – and since our slowest observing cadence is still $\sim$ 80 Hz, even the shortest duration events simulated here are well sampled in our data, resulting in minimal sampling related annealing of the dip depth. We note that *characterising* an event is a different problem to detection however, so larger opposition angles may be of benefit in this regard since the details of events are better resolved. Overall, the detection efficiencies of both camera modes are very similar, and Speed mode generally provides marginally better detection success for objects with $r \leq 2.5$ km, and those within 100 AU from the Earth. However, we suspect that this is due in part to serendipitous false positives in the Speed mode data (see below).

We break down the detection efficiency as a function of object size in Figure 4.20. For the smallest 250 m radius objects, we report non-negligible detection efficiencies at distances of 30 - 60 AU in either camera mode. This is markedly better than those reported in Bianco et al. (2010), who injected objects of this size at a distance of 43 AU in 5 Hz TAOS I data. At their data SNR and sampling frequency, they were only able to detect 0.002% of the injected events. With our substantially higher frame-rate, we can recover $\sim$ 4% of these events at a comparable distance of 40 AU in our Sensitivity mode data. However, the Speed mode detection efficiencies for the smaller objects are suggestive of lingering false positives not removed by our rejection criterion e.g. see that there are non-negligible detection efficiencies for 250 m

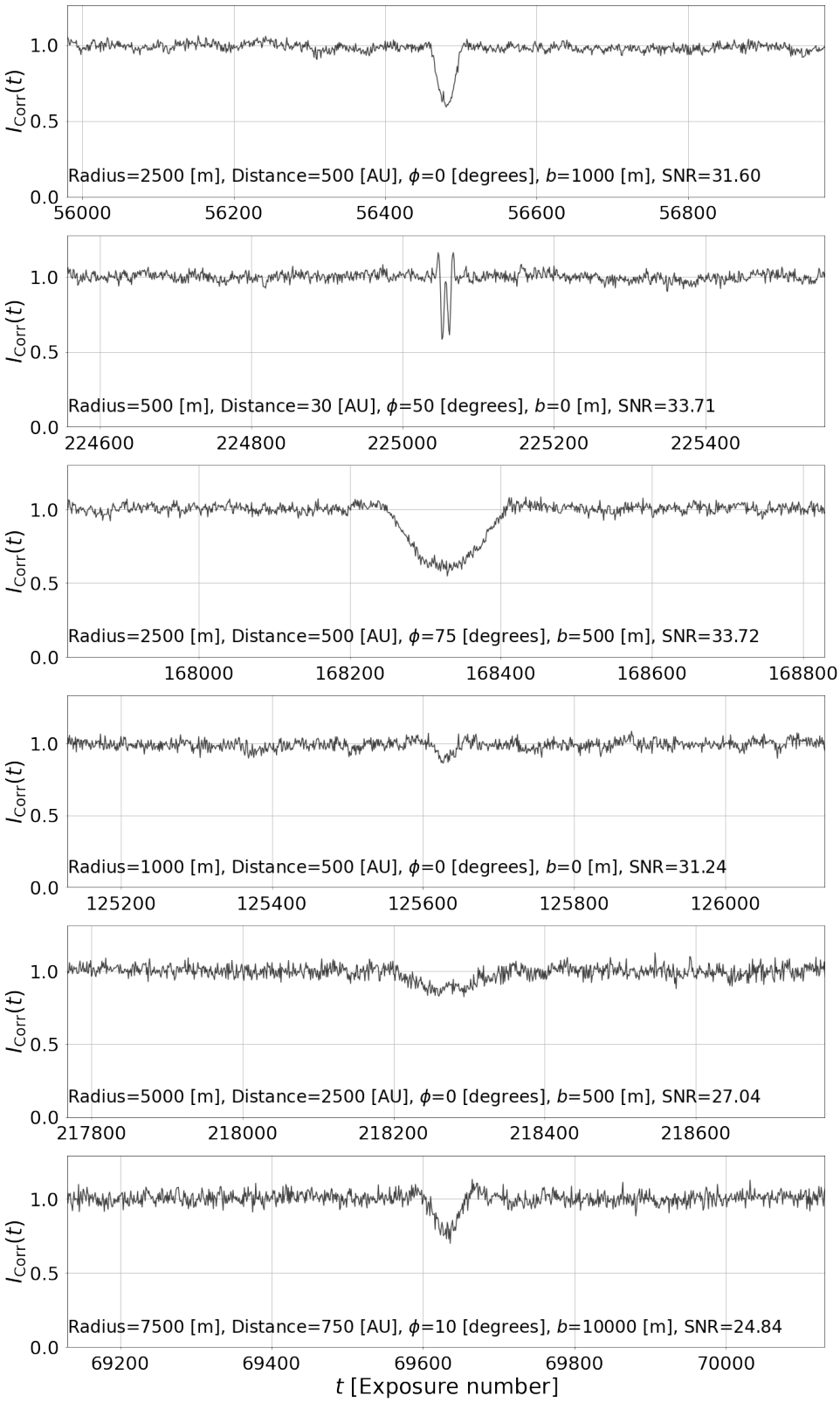


Figure 4.17: Simulated occultation events detected in 83 Hz data acquired in Sensitivty mode. Plots are annotated with the event parameters.

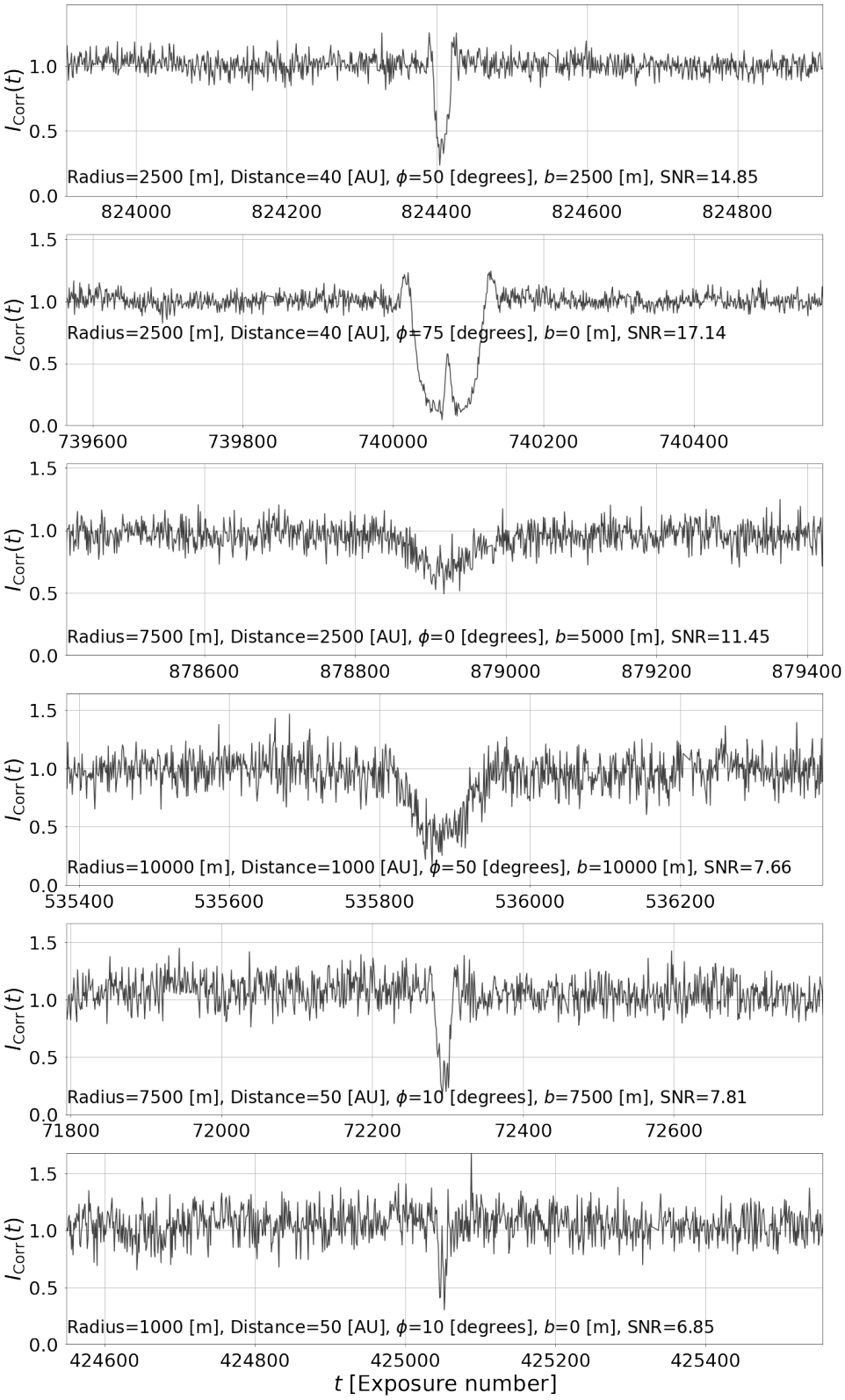


Figure 4.18: Simulated occultation events detected in 300 Hz data acquired in Speed mode. Plots are annotated with the event parameters.

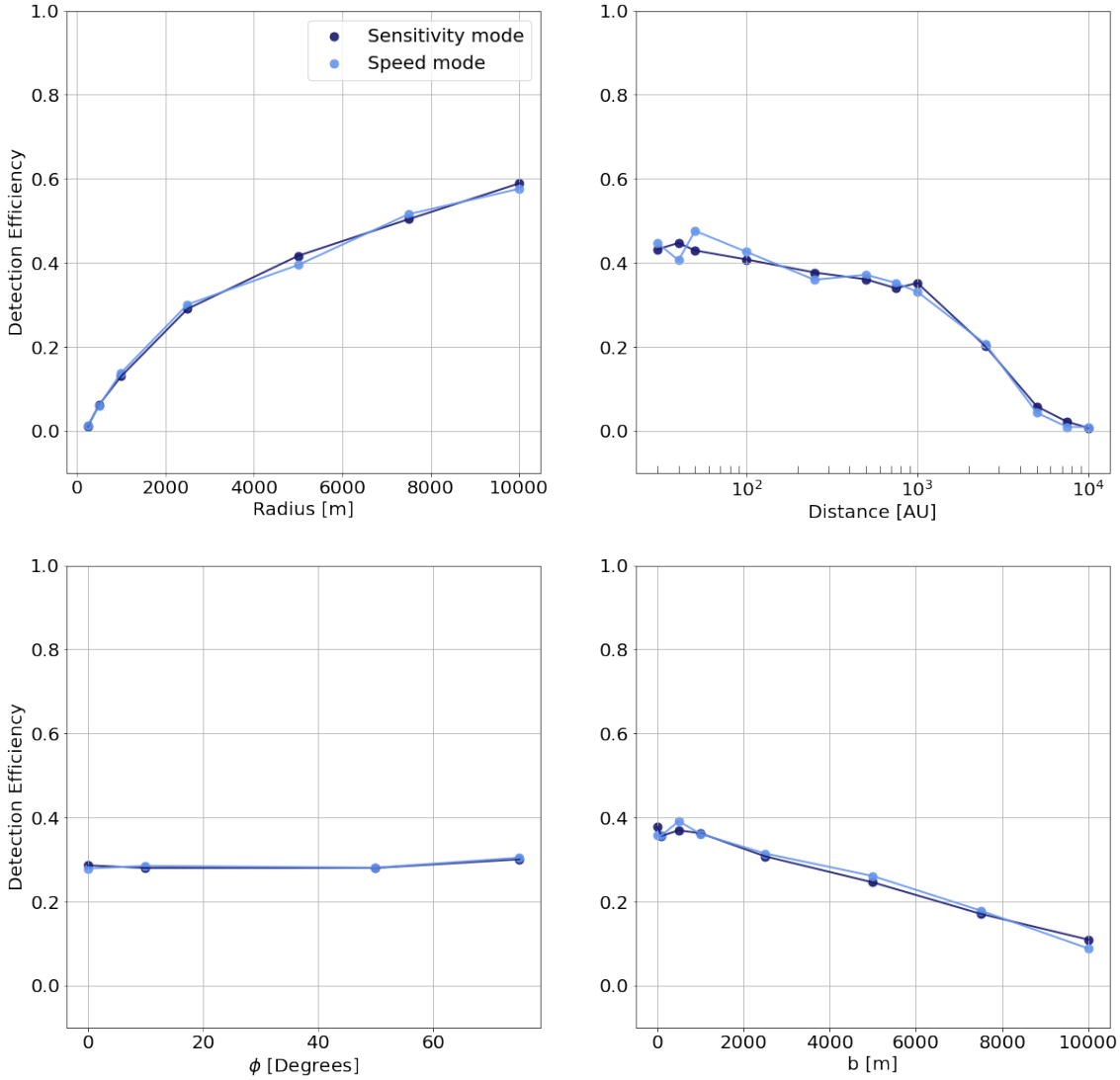


Figure 4.19: The detection efficiency as a function of simulated occultation event parameters for the Sensitivity and Speed mode data sets.

objects at distances greater than 100 AU, where the occultation should be completely buried in the light curve noise, but just by chance, residual correlated noise features in the data have coincided with a real injected event. This not only highlights the necessity of by-eye follow up of flagged detections when searching for real events, but also an approach to quantifying the false positive rate. The simplest approach would be to the run the detection algorithm on our light curves with no injected events, under the assumption these do not contain real occultations, and assess the number of false positives returned. Going further, we could utilise Gaussian Processes (MacKay et al., 1998) to model the correlated noise in our data, and generate synthetic light curves with no injected occultation signals, and run the detection algorithm on these. Both approaches will be considered in future work, and will be a requirement in any analysis claiming a real detection.

We have a $\sim 10\%$ chance to detect objects with a radius of 10 km at 7500 AU in Sensitivity mode data, but this drops to a small, although non-zero chance at 10,000 AU on this 2.1 m telescope. Speed mode provides similar detection efficiencies for these very far out, but fairly large objects. We are then only able to recover occultations by large Oort cloud objects at the closest boundary of the inner disk, thought to be distributed within $\sim 10,000 - 20,000$ AU (Oort, 1950; Hills, 1981; Morbidelli, 2005), but are reasonably sensitive to $r \geq 2.5$ km objects at distances of 1000 AU in either camera mode, corresponding to the smallest predicted semi-major axes for Oort Cloud objects. Objects beyond this in the outer cloud, unless unusually large, are likely beyond our reach in this data set of nearby open clusters, with target stars of angular sizes of about 0.02 mas.

We also note that the detection efficiency does not straightforwardly drop off with increasing distance in the regime of 30 - 1000 AU, particularly for the Speed mode data. The detection efficiency of the larger objects over this regime appears to be somewhat insensitive to the distance. For 2.5 km radius objects the detection efficiency starts to steeply drop past distances of 1000 AU and for the 10 km objects the first big drop in efficiency is around 5000 AU. This makes sense, since for the $r \geq 5$ km objects, their occultations at either 50 or 1000 AU both result in substantial drops in flux of at least 50% and so are amenable to detection when thresholding the SNR to be greater than 3.3, while e.g. occultations by 2.5 km objects at 50 AU result in a $\sim 90\%$ drop in brightness, compared to only $\sim 20\%$ at 500 AU.

We plot (normalised) histograms of the number of recovered and missed events of objects

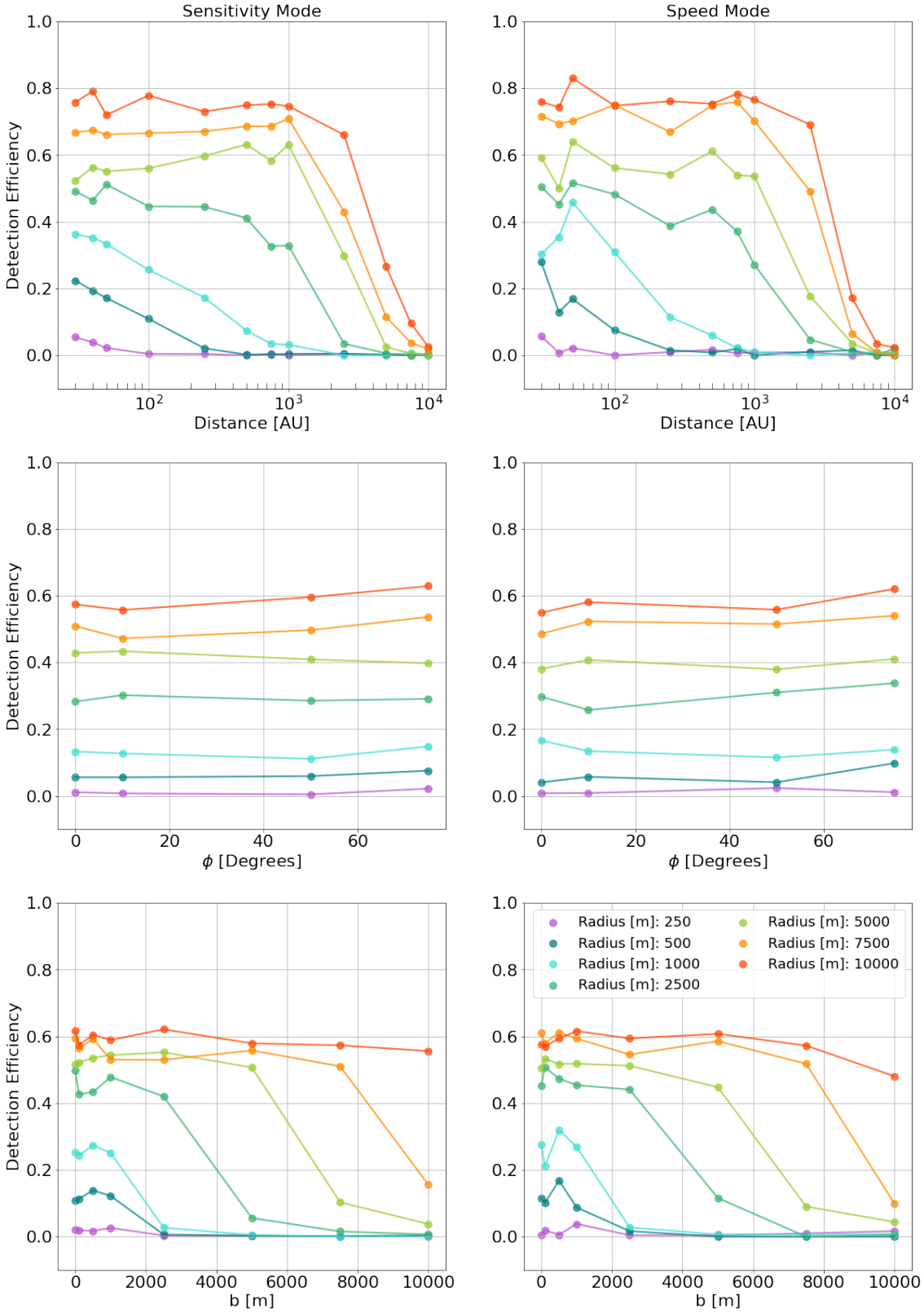


Figure 4.20: The detection efficiency as a function of occultation event parameters for the different object sizes used in the simulations.

of different sizes as a function of data SNR in Figure 4.21. As expected, the number of missed events increases with decreasing SNR, and this effect is most pronounced for small objects, as these produce the smallest dips in brightness. Conversely, at higher data SNR, the (normalised) number of recovered events generally exceeds the missed events, particularly for sub-kilometre diameter objects.

Finally, we note that having now computed our effective detection efficiency via simulation, and not having recovered any real events in our observations, we could compute constraints on the number of objects as a function of the sizes used in the simulation. However, given we only have 7 nights of data to go on, these constraints are not practically useful. Indeed, our preliminary calculations put our constraints on the upper bound of object abundances at a couple orders of magnitude above those in the literature. The main utility of this analysis is for the purpose of testing, and improving upon our data correction and candidate event detection procedures as this project progresses.

4.7 Conclusions

We have outlined a pilot study for a planned occultation survey that will mount high frame-rate sCMOS cameras in the otherwise unused areas in the focal planes of large, ground-based telescopes. Here, we describe the data system used to process the massive amounts of imagery in real time, and a procedure to analyse the recorded photometry for candidate occultation events by outer solar system objects. Because of the immense frame-rates that will be used for this project, the image processing must be done in real time, as gigabytes of imaging data are produced every second, making storage of the imaging impractical for long-term observing campaigns.

We present preliminary results of successfully trialling our data system on 7 nights of "Kinetix" sCMOS observations of nearby open clusters with the 2.1 m telescope at McDonald Observatory, Texas. We then analysed the data for candidate occultation events, and found no detections. We also assessed the detection efficiency of our small pilot survey by injecting simulated events into our observations. These suggest that we have a non-neglibible chance of detecting objects with a radius of 10 km as far out as 7500 AU, and 7.5 km radius objects out to 5000 AU. These detection probabilities increase to $\sim 70-80\%$ for objects of these sizes out to 1000 AU, providing us with a methodology to probe the size distribution of small solar

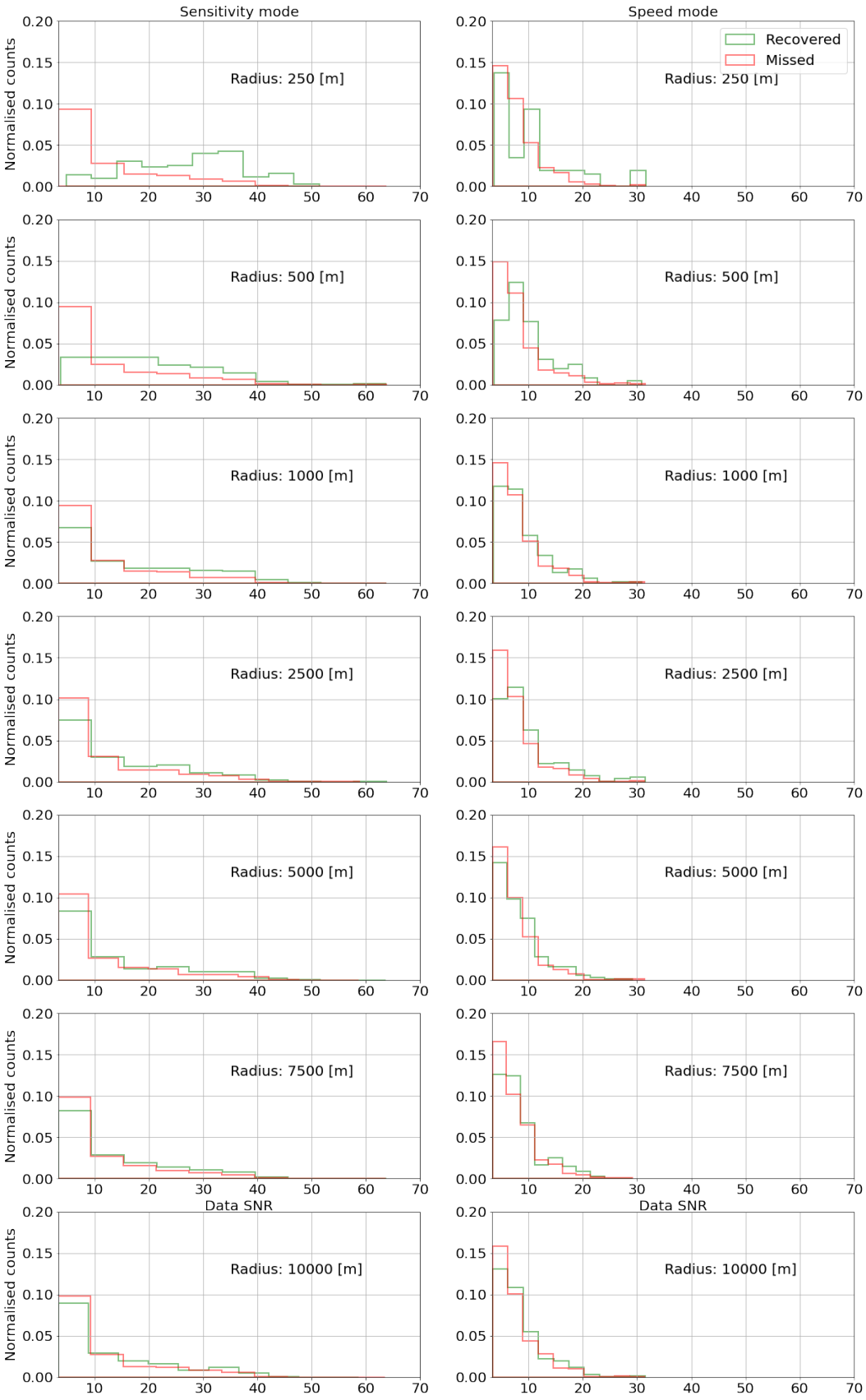


Figure 4.21: The (normalised) number of recovered and missed events for objects of different sizes as a function of data SNR.

system objects in the inner Oort cloud. The procedure used to compute our detection efficiency will be essential in further developing our data correction and occultation event search procedures as this project progresses. Indeed, experimenting with more advanced albeit expensive procedures to correct the systematic noise in our data will be prioritised in the near term due to the similarities between these sub-second, tip-tilt induced correlated features, and the occultation signals we're attempting to detect and ultimately characterise. For the bright observed sources, the systematic contributions to the noise clearly exceed the inherent random noise. In the literature, Principal Component Analysis (PCA) approaches have proved effective in capturing these systematics, where the learned "eigen light curves" in a data set can be used as basis vectors in a design matrix in a linear regression model to detrend the raw photometry (Luger et al., 2016).

Having verified our data acquisition and processing procedures in this pilot study, the next step in this project will be to mount the Kinetix on larger telescopes. The long-term goal is to place the camera in the unused parts of the focal surface of the planned 11.25 m Maunakea Spectroscopic Explorer (MSE). We have argued that the unique qualities of this project make it complementary to TAOS II. The TAOS II mission is optimised to detect objects in the Kuiper belt, down to diameters of 0.5 km. Because of the Kinetix's higher frame rates and the MSE's much greater primary mirror size, our survey will be sensitive to occultations by both smaller Kuiper belt objects, and much more distant objects in the inner Oort cloud. The former events are badly sampled in the 20 Hz TAOS II data and easily missed, while the latter require faint background stars of a small angular size to produce detectable occultation signatures, which is challenging, if not impossible, with the small 1.3 m TAOS II telescopes.

5

Summary and Future Work

In this thesis, I and co-authors have developed two new image processing algorithms – PyTorchDIA and *The Thresher* – and have introduced a pilot serendipity study for detecting the population of small outer solar system objects. All of this research has required developing performant image processing *tools*, and open source software releases of the algorithms described in Chapters 2 and 3 have been made available to encourage adoption, and further experimentation by the wider astronomy community. Indeed, it has been gratifying to hear back from academics, undergraduates and enthusiastic astrophotographers who are trialing these applications in their research, final year projects or hobbies.

All of these image processing applications make grateful use of the rich suite of Python libraries. In particular, we have shown how machine learning frameworks can be used as generic modelling environments. For image processing in particular there are striking structural similarities between foundational deep learning operations (namely the 2D convolution) and common astronomical image models. In addition to their automatic differentiation capabilities – which is a welcomed feature given the large parameter space of many image models

– frameworks like PyTorch make running applications on hardware accelerators such as GPUs extremely simple, allowing us to exploit the inherent parallelism in image processing. Similarly, in Chapter 4 we have also demonstrated how combining Python libraries such as Numpy and Numba can achieve straightforward parallelism across multiple CPUs to realise time critical image processing tasks.

The time it takes to develop code usually exceeds the total time it's set to run. This is undoubtedly why dynamically typed languages like Python, with its extensive box of packages, is so favoured amongst researchers, as ideas can be sketched out and implemented quickly. As far as image processing is concerned, we show that this ease of use and flexibility need not come at the expense of performance.

The main contributions of this thesis are as follows:

- A new algorithm for difference image analysis that is fast, scalable and flexible
- A new algorithm for processing lucky imaging data (and ground-based imaging in general) that can improve *both* the signal-to-noise and resolution of the observed scene
- The development of a data system that can process in real time the imagery obtained at frame rates on the order of $\sim$ 100 Hz, for use in a serendipity survey to detect the occultations of background stars by small outer solar system objects

Considering future work, I'd first like to thank those that have reached out to suggest improvements to both *PyTorchDIA* and *The Thresher* while they adapt these tools for their own use-cases, and help to keep the code up-to-date on the software engineering front. A great advantage of writing these applications in PyTorch is that these machine learning frameworks are developing in parallel with progress in accelerator hardware due to the importance of GPUs in modern deep learning. As such, it'll be interesting to see how these tools will develop with advances in both software (e.g. using mixed precision training) and hardware (e.g. new GPU models with Tensor cores) to further speed up computations.

The work in Chapter 4 will be used as a springboard for a paper submission, so any and all feedback is strongly encouraged to help us get it into good shape. Looking further ahead, we will attempt to fully automate the real time data processing system, such that the software "knows" when the telescope has changed to a new target, and so can start up a new data run

on the fly. We will also experiment with more advanced procedures to removing systematic noise in the high frame-rate aperture photometry, using our 2.1 m McDonald data as a testbed. We then plan to trial this improved data system on larger telescopes. In the shorter term, we are currently planning another observing run at McDonald later this year to attempt to detect small Jupiter Trojans occulting distant background stars with our data system.

Bibliography

Alard, C. 2000, Astronomy and Astrophysics Supplement Series, 144, 363

Alard, C., & Lupton, R. H. 1998, The Astrophysical Journal, 503, 325

Albrow, M. 2017, MichaelDAlbrow/pyDIA: Initial release on github.

Albrow, M. et al. 2009, Monthly Notices of the Royal Astronomical Society, 397, 2099

Andrae, R. 2010, arXiv preprint arXiv:1009.2755

Arimatsu, K. et al. 2019a, The Astronomical Journal, 158, 236

——. 2019b, Nature Astronomy, 3, 301

Assémat, F., Wilson, R. W., & Gendron, E. 2006, Optics express, 14, 988

Ayers, G., & Dainty, J. C. 1988, Optics letters, 13, 547

Becker, A., Homrighausen, D., Connolly, A., Genovese, C., Owen, R., Bickerton, S., & Lupton, R. 2012, Monthly Notices of the Royal Astronomical Society, 425, 1341

Bellm, E. C. et al. 2018, Publications of the Astronomical Society of the Pacific, 131, 018002

Benavidez, P. G., & Bagatin, A. C. 2009, Planetary and Space Science, 57, 201

Benz, W., & Asphaug, E. 1999, Icarus, 142, 5

Bianco, F. et al. 2010, The Astronomical Journal, 139, 1499

Bianco, F. B., Protopapas, P., McLeod, B. A., Alcock, C. R., Holman, M. J., & Lehner, M. J. 2009, The Astronomical Journal, 138, 568

Bickerton, S., Kavelaars, J., & Welch, D. 2008, The Astronomical Journal, 135, 1039

Bond, I. et al. 2001, Monthly Notices of the Royal Astronomical Society, 327, 868

Borman, S., & Stevenson, R. L. 1998, in 1998 Midwest symposium on circuits and systems (Cat. No. 98CB36268), IEEE, 374–378

Bottke Jr, W. F., Durda, D. D., Nesvornỳ, D., Jedicke, R., Morbidelli, A., Vokrouhlickỳ, D., & Levison, H. 2005, Icarus, 175, 111

Bottou, L., Curtis, F. E., & Nocedal, J. 2018, Siam Review, 60, 223

Bottou, L., et al. 1998, On-line learning in neural networks, 17, 142

Bradley, L. et al. 2016, Astrophysics Source Code Library, ascl

Bramich, D. 2008, Monthly Notices of the Royal Astronomical Society: Letters, 386, L77

Bramich, D., Bachelet, E., Alsubai, K., Mislis, D., & Parley, N. 2015, Astronomy & Astrophysics, 577, A108

Bramich, D., Figuera Jaimes, R., Giridhar, S., & Arellano Ferro, A. 2011, Monthly Notices of the Royal Astronomical Society, 413, 1275

Bramich, D. et al. 2013, Monthly Notices of the Royal Astronomical Society, 428, 2275

Bramich, D., Horne, K., Alsubai, K., Bachelet, E., Mislis, D., & Parley, N. 2016, Monthly Notices of the Royal Astronomical Society, 457, 542

Campisi, P., & Egiazarian, K. 2017, Blind image deconvolution: theory and applications (CRC press)

Chang, H.-K., Liu, C.-Y., & Chen, K.-T. 2011, Monthly Notices of the Royal Astronomical Society, 411, 427

Davis, D., & Farinella, P. 1997, Icarus, 125, 50

Dhillon, V. et al. 2014, Monthly Notices of the Royal Astronomical Society, 444, 4009

——. 2007, Monthly Notices of the Royal Astronomical Society, 378, 825

Dones, L., Weissman, P. R., Levison, H. F., & Duncan, M. J. 2004, in Star Formation in the Interstellar Medium: In Honor of David Hollenbach, Vol. 323, 371

Doressoundiram, A., Roques, F., Boissel, Y., Arenou, F., Dhillon, V., Littlefair, S., & Marsh, T. 2013, Monthly Notices of the Royal Astronomical Society, 428, 2661

Duncan, M. J., & Levison, H. F. 1997, Science, 276, 1670

Figuera Jaimes, R., Arellano Ferro, A., Bramich, D., Giridhar, S., & Kuppuswamy, K. 2013, Astronomy & Astrophysics, 556, A20

Fukugita, M., Shimasaku, K., Ichikawa, T., Gunn, J., et al. 1996, The Sloan digital sky survey photometric system, Tech. rep., SCAN-9601313

Göddeke, D., Strzodka, R., & Turek, S. 2005, Accelerating double precision FEM simulations with GPUs (Univ. Dortmund, Fachbereich Mathematik)

Golub, G., & Van Loan, C. 1996, Press, Baltimore

Goodfellow, I., Bengio, Y., Courville, A., & Bengio, Y. 2016, Deep learning (MIT press Cambridge)

Hands, T. O., Dehnen, W., Gration, A., Stadel, J., & Moore, B. 2019, Monthly Notices of the Royal Astronomical Society, 490, 21

Harpsøe, K. B., Jørgensen, U. G., Andersen, M. I., & Grundahl, F. 2012, Astronomy & Astrophysics, 542, A23

Harris, C. R. et al. 2020, Nature, 585, 357

Hartung, S., Shukla, H., Miller, J. P., & Pennypacker, C. 2012, in 2012 19th IEEE International Conference on Image Processing, IEEE, 1685–1688

Heavens, A. 2009, arXiv preprint arXiv:0906.0664

Hills, J. 1981, The Astronomical Journal, 86, 1730

Hirsch, M., Harmeling, S., Sra, S., & Schölkopf, B. 2011, Astronomy & Astrophysics, 531, A9

Hirsch, M., Wareham, R. J., Martin-Fernandez, M. L., Hobson, M. P., & Rolfe, D. J. 2013, PloS one, 8, e53671

Hitchcock, J. A., Hundertmark, M., Foreman-Mackey, D., Bachelet, E., Dominik, M., Street, R., & Tsapras, Y. 2021, Monthly Notices of the Royal Astronomical Society, 504, 3561

Hook, R., & Lucy, L. 1994, in The Restoration of HST Images and Spectra-II, 86

Huang, C.-K. et al. 2021, Publications of the Astronomical Society of the Pacific, 133, 034503

Huber, P., & Ronchetti, E. 2009, Wiley Series in Probability and Statistics New York, NY, USA Wiley-IEEE

Huber, P. J. 1992, in Breakthroughs in statistics (Springer), 492–518

Hunter, J. D. 2007, Computing in science & engineering, 9, 90

Ivezić, Ž. et al. 2019, The Astrophysical Journal, 873, 111

Jordà, M., Valero-Lara, P., & Peña, A. J. 2019, IEEE Access, 7, 70461

Kains, N., Bramich, D., Figuera Jaimes, R., Arellano Ferro, A., Giridhar, S., & Kuppuswamy, K. 2012, Astronomy & Astrophysics, 548, A92

Kenyon, S. J. 2002, Publications of the Astronomical Society of the Pacific, 114, 265

Kenyon, S. J., & Luu, J. X. 1999, The Astronomical Journal, 118, 1101

Kingma, D. P., & Ba, J. 2014, arXiv preprint arXiv:1412.6980

Knuth, K. H. 2006, arXiv preprint physics/0605197

Korevaar, M. A., Goorden, M. C., Heemskerk, J. W., & Beekman, F. J. 2011, Physics in Medicine & Biology, 56, 4785

Krizhevsky, A., Sutskever, I., & Hinton, G. E. 2012, in Advances in neural information processing systems, 1097–1105

Labeyrie, A. 1970, Astron. Astrophys., 6, 85

Lam, S. K., Pitrou, A., & Seibert, S. 2015, in Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC, 1–6

Lang, D., & Hogg, D. W. 2020, arXiv preprint arXiv:2012.15836

Lang, D., Hogg, D. W., Mierle, K., Blanton, M., & Roweis, S. 2010, The astronomical journal, 139, 1782

Law, N. M., Mackay, C. D., & Baldwin, J. E. 2006, Astronomy & Astrophysics, 446, 739

Lawrence, S., Giles, C. L., Tsoi, A. C., & Back, A. D. 1997, IEEE transactions on neural networks, 8, 98

LeCun, Y., Bengio, Y., & Hinton, G. 2015, nature, 521, 436

Lee, M. A., Budavári, T., White, R. L., & Gulian, C. 2017, Astronomy and computing, 21, 15

Lehner, M. J. et al. 2012, in Ground-based and Airborne Telescopes IV, Vol. 8444, SPIE, 97–108

Levin, A., Weiss, Y., Durand, F., & Freeman, W. T. 2011, IEEE transactions on pattern analysis and machine intelligence, 33, 2354

Li, J., Yu, C., Sun, J., & Xiao, J. 2013, in Proceedings 2013 International Conference on Mechatronic Sciences, Electric Engineering and Computer (MEC), IEEE, 1937–1942

Lindler, D. J., A'Hearn, M. F., Besse, S., Carcich, B., Hermalyn, B., & Klaasen, K. P. 2013, Icarus, 222, 571

Liu, C.-Y., Doressoundiram, A., Roques, F., Chang, H.-K., Maquet, L., & Auvergne, M. 2015, Monthly Notices of the Royal Astronomical Society, 446, 932

Liu, D. C., & Nocedal, J. 1989, Mathematical programming, 45, 503

Luger, R., Agol, E., Kruse, E., Barnes, R., Becker, A., Foreman-Mackey, D., & Deming, D. 2016, The Astronomical Journal, 152, 100

Mackay, C. 2013, Monthly Notices of the Royal Astronomical Society, 432, 702

MacKay, D. J., et al. 1998, NATO ASI series F computer and systems sciences, 168, 133

Magain, P., Courbin, F., & Sohy, S. 1998, The Astrophysical Journal, 494, 472

Magnier, E. et al. 2013, The Astrophysical Journal Supplement Series, 205, 20

Marshall, J. et al. 2019, arXiv preprint arXiv:1907.07192

Martin, F., Tokovinin, A., Ziad, A., Conan, R., Borgnino, J., Avila, R., Agabi, A., & Sarazin, M. 1998, Astronomy and Astrophysics, 336, L49

Mazur, M. J., Metchev, S., Brown, R. A., Gupta, R., Bloch, R., Mills, T., & Pass, E. 2022, arXiv preprint arXiv:2210.05808

Micikevicius, P. et al. 2017, arXiv preprint arXiv:1710.03740

Morbidelli, A. 2005, arXiv preprint astro-ph/0512256

Nihei, T., Lehner, M., Bianco, F., King, S.-K., Giammarco, J., & Alcock, C. 2007, The Astronomical Journal, 134, 1596

Nocedal, J., & Wright, S. 2006, New York

Oort, J. H. 1950, Bulletin of the Astronomical Institutes of the Netherlands, 11, 91

Pan, M., et al. 2005, Icarus, 173, 342

Paszke, A. et al. 2019, arXiv preprint arXiv:1912.01703

Perkel, J. M. 2018, Nature, 563, 145

Pickles, A. 1998, Publications of the Astronomical Society of the Pacific, 110, 863

Por, E. H., Haffert, S. Y., Radhakrishnan, V. M., Doelman, D. S., van Kooten, M., & Bos, S. P. 2018, in Adaptive Optics Systems VI, Vol. 10703, International Society for Optics and Photonics, 1070342

Pratlong, J., Wang, S.-Y., Lehner, M., Jorden, P., Jerram, P., & Johnson, S. 2016, in High Energy, Optical, and Infrared Detectors for Astronomy VII, Vol. 9915, SPIE, 418–429

Qiu, P., Mao, Y.-N., Lu, X.-M., Xiang, E., & Jiang, X.-J. 2013, Research in Astronomy and Astrophysics, 13, 615

Raymond, S. N., Armitage, P. J., Veras, D., Quintana, E. V., & Barclay, T. 2018, Monthly Notices of the Royal Astronomical Society, 476, 3031

Roques, F., Boissel, Y., Doressoundiram, A., Sicardy, B., & Widemann, T. 2009, Earth, Moon, and Planets, 105, 201

Ruder, S. 2016, arXiv preprint arXiv:1609.04747

Schlichting, H., Ofek, E., Wenz, M., Sari, R., Gal-Yam, A., Livio, M., Nelan, E., & Zucker, S. 2009, Nature, 462, 895

Sedaghat, N., & Mahabal, A. 2018, Monthly Notices of the Royal Astronomical Society, 476, 5365

Siraj, A., & Loeb, A. 2020, The Astrophysical Journal Letters, 891, L3

Skottfelt, J. et al. 2015, Astronomy & Astrophysics, 574, A54

Staley, T. D., Mackay, C. D., King, D., Suess, F., & Weller, K. 2010, in Ground-based and Airborne Instrumentation for Astronomy III, Vol. 7735, International Society for Optics and Photonics, 77355Z

Steele, I. A., Jermak, H., Copperwheat, C. M., Smith, R. J., Poshyachinda, S., & Soonthorntham, B. 2016, in High Energy, Optical, and Infrared Detectors for Astronomy VII, Vol. 9915, International Society for Optics and Photonics, 991522

Tsapras, Y. et al. 2019, Publications of the Astronomical Society of the Pacific, 131, 124401

Tubbs, R. N. 2003, arXiv preprint astro-ph/0311481

Vio, R., Bardsley, J., & Wamsteker, W. 2005, Astronomy & Astrophysics, 436, 741

Walker, G. 2020, in American Astronomical Society Meeting Abstracts# 235, Vol. 235, 175–01

Wang, S.-Y. et al. 2016, in Ground-based and Airborne Instrumentation for Astronomy VI, Vol. 9908, SPIE, 1309–1314

Weissman, P. R. 1990, Nature, 344, 825

White, R., & Allen, R. 1991, The Restoration of HST Images and Spectra

Wozniak, P. 2000, Acta Astronomica, 50, 421

Young, A. T. 1967, Astronomical Journal, Vol. 72, p. 747 (1967), 72, 747

Zackay, B., Ofek, E. O., & Gal-Yam, A. 2016, The Astrophysical Journal, 830, 27

Zhang, Z.-W. et al. 2013, The Astronomical Journal, 146, 14

Zhao, Y., Luo, Q., Wang, S., & Wu, C. 2013, in 2013 IEEE 9th International Conference on e-Science, IEEE, 70–77