universität wien

MASTERARBEIT / MASTER'S THESIS

Titel der Masterarbeit / Title of the Master's Thesis

„Optimised Centroiding of Stars for Space Applications“

verfasst von / submitted by

Roman Ferstl, BSc

angestrebter akademischer Grad / in partial fulfilment of the requirements for the degree of

Master of Science (MSc)

Wien, 2016 / Vienna 2016

Studienkennzahl lt. Studienblatt / degree programme code as it appears on the student record sheet:	A 066 861
Studienrichtung lt. Studienblatt / degree programme as it appears on the student record sheet:	Masterstudium Astronomie UG2002
Betreut von / Supervisor:	Ao.Univ.Prof. Dr. Franz Kerschbaum

Contents

Abstract English 6

Abstract German 7

1. Introduction 9

2. Data Simulation with StarSim 11
2.1. Stellar Field . . . 12
2.2. Point Spread Function . . . 13
2.3. Shot Noise . . . 15
2.4. Background . . . 15
2.5. Bias and Read Noise . . . 16
2.6. Dark Current and Hot-Pixels . . . 16
2.7. Variations in Pixel Sensitivity . . . 17
2.8. Glitches and Cosmic Rays . . . 19
2.8.1. Van Allen Radiation Belts and the South Atlantic Anomaly . . . 21
2.8.2. Simulation of Glitches in StarSim . . . 22
2.9. Conclusion . . . 23

3. Centroiding Algorithms 25
3.1. Cramer-Rao Bounds . . . 27
3.2. Centre of Gravity Algorithms . . . 31
3.2.1. Standard Centre of Gravity (CoG) . . . 31
3.2.1.1. Implementation . . . 31
3.2.1.2. Error Estimation . . . 31
3.2.1.3. Performance . . . 33
3.2.1.4. Summary . . . 33
3.2.2. Weighted Centre of Gravity (WCoG and IWC) . . . 34
3.2.2.1. WCoG - Implementation . . . 34
3.2.2.2. WCoG - Performance . . . 35
3.2.2.3. WCoG - Summary . . . 36
3.2.2.4. IWC - Implementation . . . 37
3.2.2.5. IWC - Performance . . . 37
3.2.2.6. IWC - Summary . . . 38
3.2.3. Iteratively Weighted Centre of Gravity (IWCoG) . . . 39
3.2.3.1. Implementation . . . 41
3.2.3.2. Error Estimation . . . 41
3.2.3.3. Performance . . . 43

3.2.3.4. Summary . . . 43
3.3. Correlation-Based Centroiding . . . 45
3.3.1. Correlation and Convolution . . . 45
3.3.2. Application to Centroiding . . . 46
3.3.3. Centroid Estimation by Interpolation . . . 47
3.3.3.1. Implementation . . . 47
3.3.3.2. Error Estimation . . . 48
3.3.3.3. Performance . . . 48
3.3.3.4. Summary . . . 49
3.3.4. A New Method: Centroid Estimation by Upsampling . . . 50
3.3.4.1. The Upsampling Factor . . . 51
3.3.4.2. Implementation . . . 53
3.3.4.3. Error Estimation . . . 54
3.3.4.4. Performance . . . 54
3.3.4.5. Summary . . . 56
3.4. Direct Fitting Strategies . . . 57
3.4.1. Gaussian Three-Point Fit (G3P) . . . 57
3.4.1.1. Implementation . . . 58
3.4.1.2. Performance . . . 59
3.4.1.3. Summary . . . 59
3.4.2. Two-Dimensional Non-Linear Least-Squares Minimisation . . . 60
3.4.2.1. Levenberg-Marquardt Algorithm (LMA) . . . 60
3.4.2.2. Implementation . . . 62
3.4.2.3. Application to Centroiding . . . 62
3.4.2.4. Error Estimation . . . 63
3.4.2.5. Performance . . . 64
3.4.2.6. Summary . . . 65
3.5. Comparison of Algorithms . . . 66
3.5.1. Dependency on Signal-to-Noise Ratio . . . 66
3.5.2. Dependency on Region of Interest and Start Position . . . 67
3.5.3. Dependency on Thresholds . . . 70
3.5.4. Impact of Cosmic Ray Hits . . . 72
3.5.5. Flat-Field: Intra-Pixel Star Positions and Pixel Sensitivities . . . 77
3.5.6. Computation Time . . . 79
3.6. Conclusion . . . 81

4. Space Applications 83
4.1. EChO - Exoplanet Characterisation Observatory . . . 83
4.1.1. Mission Description . . . 83
4.1.2. Fine Guidance Sensor . . . 85
4.1.3. Point Spread Function . . . 87
4.1.4. Centroiding in Different Operating Modes . . . 88
4.1.4.1. Handover from Star Trackers . . . 91
4.1.4.2. Centroiding during Science Mode . . . 92
4.1.5. Conclusion . . . 94

4.2. CHEOPS - CHaracterising ExOPlanets Satellite 95
4.2.1. Mission Description 95
4.2.2. Observations in the South Atlantic Anomaly 96
4.2.2.1. MOST - Microvariability and Oscillations of STars 97
4.2.2.2. Observations of HD 189733 98
4.2.2.3. Results and Statistical Statements 100
4.2.3. Point Spread Function 103
4.2.4. Selection of Centroiding Algorithms for Fine-Guiding 105
4.2.4.1. Nominal Observation Mode 108
4.2.4.2. Observations Inside the South Atlantic Anomaly 110
4.2.4.3. Observations of Crowded Fields 114
4.2.5. Conclusion 115

5. Conclusion **117**

A. Appendix **119**
A.1. Configuration of StarSim 119
A.1.1. Entries of stars_config.xml 119
A.1.2. Entries of datasim_config.xml 119
A.2. Abbreviations 124
A.3. Implementations of Centroiding Algorithms 125
A.3.1. Centre of Gravity 125
A.3.2. Weighted Centre of Gravity 127
A.3.3. Iteratively Weighted Centre of Gravity 133
A.3.4. Correlation-based Centroiding 135
A.3.5. Gaussian Three-Point Fitting 138
A.3.6. Levenberg-Marquardt Algorithm 141
A.4. Data Tables on Cosmic Ray Hit Analysis 146

Abstract English

High precision pointing information is a vital input for the attitude control of space telescopes. This pointing information can be extracted from images by applying centroiding algorithms. Such images may be obtained by dedicated fine guidance sensors or by the science instrument itself. This thesis includes an investigation for the selection of centroiding algorithms for two space missions, the Exoplanet Characterisation Observatory (EChO) and the CHaracterising ExOPlanets Satellite (CHEOPS). The tool *StarSim* was developed to test various centroiding algorithms based on unique mission requirements. It allows to simulate telescopic observations of stars based on the specific instrumental features of the individual space missions. Since noise sources in astronomical images represent random processes, a statistical evaluation of the centroiding performance has been carried out with Monte-Carlo analyses that included tens of thousands of simulated images.

Eight centroiding algorithms have been analysed with respect to their applicability for the fine guidance of space telescopes. These algorithms have been categorised into centre of gravity algorithms, correlation-based centroiding and direct fitting strategies. An analytical description, implementations and error estimation equations have been provided for each algorithm. The scope of applicability has been tested under various circumstances, such as in the observation of faint stars with high noise in the images. Furthermore, the influence of cosmic ray hits on the centroiding performance has been tested with models that were derived from real observations. For the EChO mission, fine pointing errors below ten milliarcseconds are required to maintain the necessary photometric stability during observations. Only two out of eight centroiding methods have met this stringent requirement in simulated worst-case scenarios. During the CHEOPS mission, the satellite will temporarily experience an increased rate of cosmic ray hits during some phase of its low-Earth orbit, due to the inner Van Allen radiation belt. For this case, three centroiding techniques have been presented that are capable of providing pointing errors below one arcsecond. In general, this thesis serves as a compendium for the implementation of different kinds of centroiding algorithms in astronomical space missions.

Abstract German

Für die Ausrichtung und Nachführung von Weltraumteleskopen sind meist hochpräzise Positionsangaben des Zielobjekts notwendig. Diese Positionsinformationen können aus Bildern durch die Anwendung von Zentrierungsalgorithmen extrahiert werden. Oftmals werden zur Gewinnung der Bilder optische Sensoren eingesetzt, welche speziell für die Feinausrichtung des Teleskops entwickelt wurden. Im Rahmen dieser Masterarbeit wurden für die zwei Weltraummissionen EChO (Exoplanet Characterisation Observatory) und CHEOPS (CHaracterising ExOPlanets Satellite) Untersuchungen für die Auswahl von Zentrierungsalgorithmen zur Feinausrichtung durchgeführt. Die Auswahl erfolgte hierbei anhand von simulierten Beobachtungen, in welchen auf die spezifischen Missionsanforderungen eingegangen wurde. Hierfür wurde der Datensimulator *StarSim* entwickelt. Dieser ermöglicht es Teleskopbeobachtungen von Sternen, unter Berücksichtigung spezieller Teleskopeigenschaften, zu simulieren. Da Rauschquellen in astronomischen Bildern durch Zufallsprozesse abbildbar sind, erfolgte die Auswertung der Zentrierungsalgorithmen mittels statistischer Methoden. Insbesondere wurden hierfür Monte-Carlo-Simulationen durchgeführt, in welchen zehntausende, simulierte Bilder ausgewertet wurden.

Acht Zentrierungsalgorithmen wurden auf ihre Anwendbarkeit in der Feinausrichtung von Weltraumteleskopen geprüft. Es erfolgte eine Kategorisierung der Methoden in die Gruppen Massenschwerpunkt-Algorithmen, Korrelationsalgorithmen und der direkte Fit von mathematischen Funktionen. Es wurde pro Algorithmus eine analytische Beschreibung inklusive Möglichkeiten zur Fehlerabschätzung angegeben. Für jede Methode wurden Implementierungen bereitgestellt und ihre Anwendbarkeit wurde unter verschiedensten Bedingungen getestet. Der Einfluss von hochenergetischer, kosmischer Strahlung auf die Zentrierungsgenauigkeit wurde mit realistischen Modellen untersucht, wobei die Modelle von echten Beobachtungen abgeleitet wurden.

Für EChO werden Positionsangaben mit einer Genauigkeit von mindestens zehn Millibogensekunden benötigt, um die photometrische Stabilität während der Beobachtungen aufrechterhalten zu können. Nur zwei von acht Zentrierungsalgorithmen waren imstande diese Bedingung in Extremfällen zu erfüllen. Aufgrund des inneren Van-Allen-Gürtels, erfährt CHEOPS auf seiner niedrigen Umlaufbahn um die Erde, zwischenzeitlich einen erhöhten Anteil an kosmischer Strahlung. Unter diesen Bedingungen war es nur noch drei Zentrierungsmethoden möglich, Positionen mit einem Fehler unter einer Bogensekunde zu liefern. Im Allgemeinen dient diese Arbeit als Handbuch zur Implementierung verschiedenster Zentrierungsalgorithmen, welche für die Feinausrichtung von Weltraumteleskopen einsetzbar sind.

Acknowledgements

Firstly, I want to thank my parents for all the support throughout my entire life. I would definitely not be able to live it in such a good way without the two of you.
I would like to express my sincere gratitude to my thesis advisor Franz Kerschbaum for giving me the opportunity to work on this exciting project. I am very glad about all the things that I learnt as a part of such an excellent work group. A special thanks goes to Roland Ottensamer for all the discussions and helpful advices regarding my thesis. I will never forget the day when he was worried about my code, because I was wasting a single digital bit. On that day, I finally realised the degree of cautiousness that is needed to programme a spacecraft.
I also want to thank my fellow students Christoph, Markus and Philipp for all the fun we had on facing challenges together. In particular thanks to Philipp for proofreading my thesis on short notice. Last but not least, thanks to my study abroad friends Sandra, Juho, Emily, Marielle, Eline, Laura, Sanjay, Tim and Zach for all the joy moments that we shared in New Zealand.

Parts of this work were supported by the ESA PRODEX (PROgramme de Développement d'Expériences scientifiques) contract C4000112123 granted to the University of Vienna for the CHEOPS Instrument Flight Software (PI. Franz Kerschbaum).

1. Introduction

Space telescopes are essential instruments that allow the gathering of scientific data in wavelength bands which are absorbed by Earth's atmosphere. Such spacecraft often need to operate entirely autonomous for hours or days without communication with ground stations. After the initial target identification, some telescopes must be stabilised for a long period of time, particularly in precise photometric missions. In general, an Attitude and Orbit Control System (AOCS), which controls different kinds of actuators (e.g reaction wheels, thrusters, magnetorquers), is utilised to manoeuvre the vehicle during its mission. However, before changing the orientation of a spacecraft, the current attitude must be known. The attitude estimation of space telescopes can be obtained by star trackers (Liebe, 1992; Padgett & Kreutz-Delgado, 1997). In addition to star trackers, space telescopes may perform fine guidance using optical sensors that receive light from the optics of the payload instrument. This allows to compensate for the small scaled thermoelastic deformations of the telescope in order to increase the pointing stability. Fine guiding can be carried out using dedicated Fine Guidance Sensors (FGSs) which receive only a fraction of the total incident light, or using the science instrument itself. For example, three FGSs are used on the Hubble Space Telescope and additionally one of them can function as a scientific instrument that performs astrometric measurements (Nelan et al., 1998). The fine guidance task requires centroid estimations that reveal the target's position on the sensor with high precision. In the context of this thesis, I analysed and tested such centroiding algorithms in particular for their application in fine guiding.

For most telescope observations, regardless of whether in space or on the ground, guiding is a crucial task to obtain sharp images. In particular, for longer exposures, the telescope must be realigned consistently to hold the target position fixed in the field of view. Typically, the position of a guide star is frequently evaluated during the entire observation. One particular application of centroiding is in ground-based observations with adaptive optics, which are used to compensate for the effect of atmospheric distortions. The first step in this technique is wavefront sensing using Shack Hartmann sensors, consisting of a two-dimensional lenslet array which creates a spot pattern on an optical sensor. To be able to draw inferences from the spot patterns about the incident wavefront, image centroiding must be carried out to accurately determine the position of the spots (Thomas, 2004; Poyneer et al., 2003; Vyas et al., 2009a; Nicolle et al., 2004; Baker & Moallem, 2007; Uhm et al., 2008). Such ground-based techniques can also be

applied to spacecraft if they meet the stringent requirement of real-time computations on space hardware.

Within the scope of this thesis, an investigation for the selection of centroiding algorithms for fine guiding was carried out for two space missions, the Exoplanet Characterisation Observatory (EChO) and the CHaracterising ExOPlanets Satellite (CHEOPS). During the progress of my research, another mission than EChO was selected for development by the Science Programme Committee of the European Space Agency in February 2014, but EChO now competes for M4 under the name ARIEL. Since each space mission is unique in terms of science goals and instrumentation, the centroiding methods must be selected with respect to the individual mission requirements. The biggest obstacle in implementing fine guiding techniques for a telescope that has yet to be built, is the lack of images which are necessary for comprehensive tests. This gave rise to the development of *StarSim*, a tool for simulating observations of stars incorporating specific instrumental features. By knowing the mission specific requirements and hardware characteristics, *StarSim* is able to produce images that represent real telescopic observations with high accuracy. In fact, observations of stellar fields can be simulated with respect to the instrument's point spread function and the sensor characteristics, such as detector size and read noise. The resolution of the detector as well as noise are limiting factors for the centroiding performance (Vyas & Vohnsen, 2013). In Chapter 2, various effects that influence image quality are introduced together with the models that were applied in *StarSim* simulations. In Chapter 3, eight centroiding algorithms are introduced. An analytical description as well as the scope of application is provided for each centroiding method respectively and source codes are provided in languages C and Python in the Appendix A.3. Monte Carlo simulations were performed with tens of thousands of images, to determine the dependency of centroiding performance on individual image components (e.g. background, photon noise etc.). With the knowledge of such dependencies, the selection of the centroiding methods for a specific space mission becomes easier as some of the algorithms may be excluded due to the mission requirements. The remaining algorithms can be tested against the exact mission requirements for further selection. This has been done in Chapter 4 for the two space missions CHEOPS and EChO, respectively. For CHEOPS, the impact of cosmic ray hits is a crucial topic. In order to simulate such events as accurately as possible, the impact rate as well as other characteristics were derived from real observations. These were obtained from the MOST (Microvariability and Oscillations of STars) satellite and results are presented in Section 4.2.2.

2. Data Simulation with StarSim

The data simulator *StarSim* was designed to simulate observations of stellar fields as observed by real telescopes. In fact, the simulator is capable of producing images that include predefined instrumental features, such as detector size, point spread function and many others that are described in the sections below. These features are crucial for the performance analyses of centroiding algorithms with respect to specific space missions. *StarSim* is capable of simulating characteristics of charged-couple devices (CCD) like dark current, hot-pixels, bias, read-noise and variations in pixel sensitivity. Figure 2.1 depicts sample images generated with *StarSim*. The data simulator is an entirely autonomous Python-programme that can be used for any purpose, although it was initially designed for centroiding problems.

The desired stellar field can be generated by specifying the position and signal of each star. These signals are defined on sensor level in units of photons per second, which means that signal reduction caused by light passage through the optical components of the instrument (like aperture, beam splitters, dichroics, etc.) have to be pre-calculated. The final star signals in the image are affected by the configured quantum efficiency and exposure time.

Furthermore *StarSim* is able to simulate the spacecraft's jitter and rotation. The final output of *StarSim* contains the images as well as meta information primarily about preconfigured instrumental parameters. Output files can be generated in a compact FITS-format for further processing. A sample simulation of a stellar field observation including thirteen stars is shown in Figure 2.1. The image in the left panel is completely untouched, which means that no calibration steps have been performed. It contains a clearly visible gradient of pixel sensitivities as well as hot-pixels and other noise sources. Colour scales of both images were set equal for better comparison. Additionally, the pixel positions with highest and lowest signal may automatically be displayed on the top and at the bottom of the colour bar, respectively.

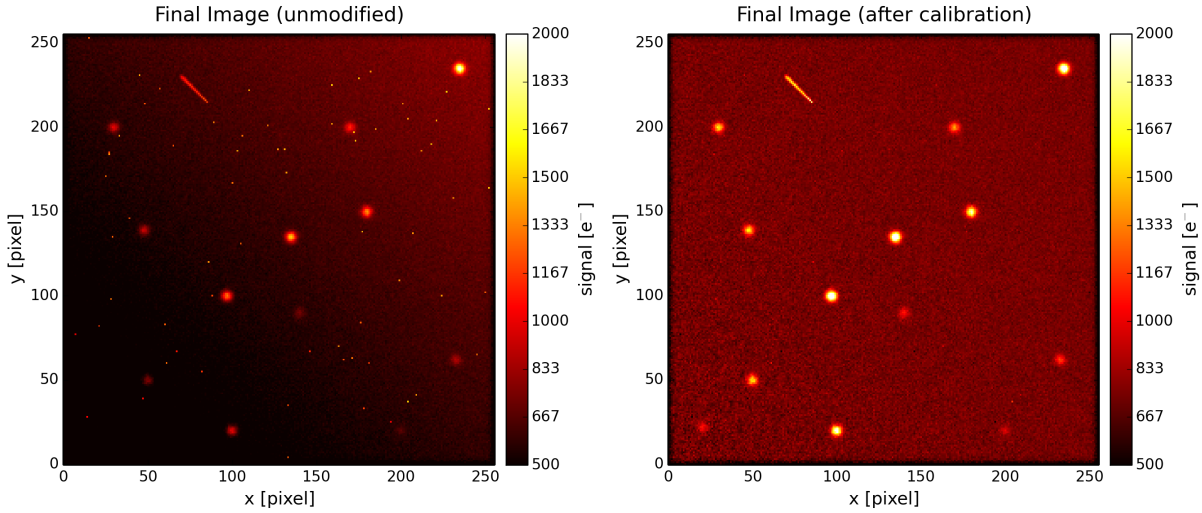

Figure 2.1.: Sample output images of *StarSim* showing thirteen stars and a glitch at position (70, 85). The detector's resolution was set to 256×256 pixels and the exposure time was set to five seconds. A symmetrical point spread function with a FWHM of five pixels was applied. The colour bar indicates the signal held by each pixel in units e^-. The left image depicts a linear flat field gradient with values increasing from position (0, 0) to position (255, 255). Additionally, it contains dark current and hot-pixels as well as other noise sources. The image in the right panel is the result after several calibration steps were performed. The image reduction included flat fielding as well as noise reduction via dark frames.

2.1. Stellar Field

Observations of existing stellar fields may be simulated via *StarSim* by specifying the position and signal for each star. The locations are defined on a pixel-centred coordinate system, which is illustrated in Figure 2.2. The origin of the coordinate system is defined at the bottom left corner of the simulated image. This coordinate system was also applied for the analyses of centroiding techniques provided in the next chapter.

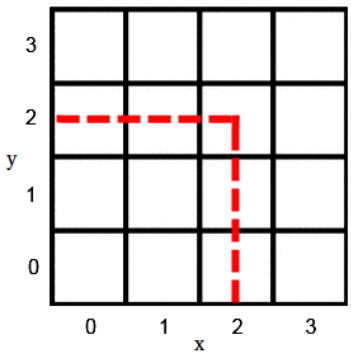

Figure 2.2.: Grid illustrating the coordinate system applied by *StarSim* in units pixel. Each square indicates a single pixel. Integer coordinates represent pixel centres. The cross section of the red dashed lines depicts the point (2.0, 2.0).

Images are created by convolving a star mask template with a discrete model of the point spread function (see Section 2.2). This process involves a Fast Fourier transform (see Section 3.3.1). The template of the star mask contains the signal with respect to the exposure time, where

the stars can be positioned on intra pixel scales. This can be achieved by oversampling the star mask as well as the point spread function before the convolution is computed. However, an alternative solution was implemented for constructing the star mask with less computation time. Each star in the star mask is treated as a point source. If a star is situated exactly at a pixel centre, this pixel holds the entire signal in the star mask. In any other case, the signal is spread over the adjacent pixels. In fact, the signal is weighted by the distance between the configured star position an the pixel centre.

2.2. Point Spread Function

The point spread function (PSF) plays a significant role in the object localisation on optical sensors (Winick, 1986). Airy patterns are often described by a Bessel function of the first kind and order one. Here we approximate the point spread function with a Gaussian profile. Such approximation is reasonable as the information required for the object localisation is extracted by regions close to the central peak.

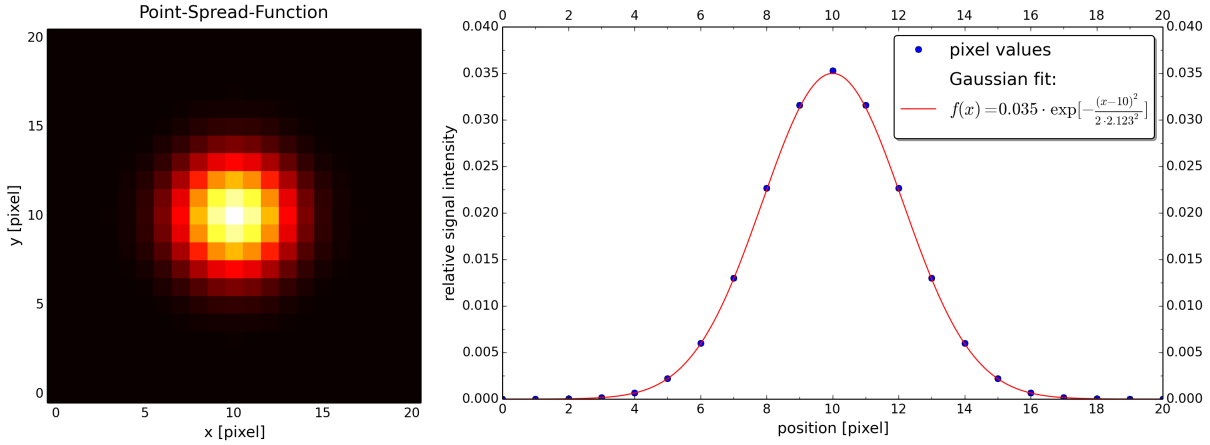

Figure 2.3.: *Left*: Normalised circular point spread function with a FWHM radius of five pixels on a two-dimensional sensor area. *Right*: One-dimensional cross section through the maximum of the PSF that is illustrated in the left panel (for row $y = 10$). A Gaussian fit has been applied for better illustration of the PSF's structure. Images were generated by *StarSim*.

The model of a two-dimensional Gaussian PSF can be defined as follows

$$f(x, y) = \frac{1}{2\pi\sigma_x\sigma_y} \exp\left[-\left(\frac{(x - x_0)^2}{2\sigma_x^2} + \frac{(y - y_0)^2}{2\sigma_y^2}\right)\right] \tag{2.1}$$

where x_0 and y_0 indicate the position of the central peak and σ_x, σ_y are the standard deviations for both dimensions. Equation (2.1) represents a bivariate normal distribution with independent variables x and y. Further treatment of multivariate normal distributions is provided by Tong (1990) and Patel & Read (1996). The distribution is normalised to unity in order to conserve the

signal on image operations. The normalisation factor $1/(2\pi\sigma_x\sigma_y)$ was obtained by integrating over the whole profile. Such normalisation is indispensable if images are further processed for photometry or other scientific applications. Figure 2.3 illustrates a sample PSF model created by *StarSim* with Equation (2.1). In some cases the PSF may strongly differ from such a Gaussian profile. Such a case is introduced in the description of the CHEOPS mission in Section 4.2.3. For this reason, *StarSim* also offers the opportunity to import external files that describe PSF models.

Full Width at Half Maximum and Standard Deviation

The full width at half maximum ($FWHM$) is often used to describe a Gaussian PSF. It is directly related to the standard deviation as shown below. Assume a one dimensional Gaussian function,

$$f(x) = A\ \exp\left[-\left(\frac{(x-\mu)^2}{2\sigma^2}\right)\right] \tag{2.2}$$

where A is the normalising constant. The maximum of Equation (2.2) is located at $x_{max} = \mu$. Therefore the locations of half-maximum, x_{half}, can be derived by solving:

$$\frac{1}{2}\ A\ f(\mu) = A\ \exp\left[-\left(\frac{(x_{half}-\mu)^2}{2\sigma^2}\right)\right] \tag{2.3}$$

Evaluating Equation (2.2) leads to $\frac{1}{2}f(\mu) = \frac{1}{2}$.

$$\begin{aligned} \exp\left[-\left(\frac{(x_{half}-\mu)^2}{2\sigma^2}\right)\right] &= \frac{1}{2} \\ -\left(\frac{(x_{half}-\mu)^2}{2\sigma^2}\right) &= \ln 2 \\ x_{half} &= \pm\,\sigma\sqrt{2\ln 2} + \mu \end{aligned} \tag{2.4}$$

The $FWHM$ is proportional to the standard deviation, σ, and can be expressed as Equation (2.5).

$$FWHM = x_{half,+} - x_{half,-} = \sigma\sqrt{2\ln 2} \approx 2.355\,\sigma \tag{2.5}$$

2.3. Shot Noise

Shot noise is introduced to imaging devices as the arrival of photons represents a random Poisson process (Blanter & Büttiker, 2000). Therefore, a Possion distribution was applied to simulate the shot noise in each pixel

$$P(x) = \frac{(\sqrt{S})^x}{x!} \exp(-\sqrt{S}) \tag{2.6}$$

where S is the signal inside the pixel. Shot noise was implemented in *StarSim* in such way that the overall signal is conserved.

2.4. Background

Signals from unresolved objects can also be modelled by Equation (2.6). In this case, S represents the mean signal from the night sky per pixel. Since this signal also passes through the entire telescope, a convolution with the point spread function must be performed. An illustration of a simulated sky background is depicted in Figure 2.4. The illustrated image is not affected by the flat field, which is simulated at a later stage. The slightly lower signal on the edge of the frame is caused by a side effect of the convolution. This effect could be removed by applying reflecting boundaries for the convolution of the PSF with the background instead of using zero-value boundaries.

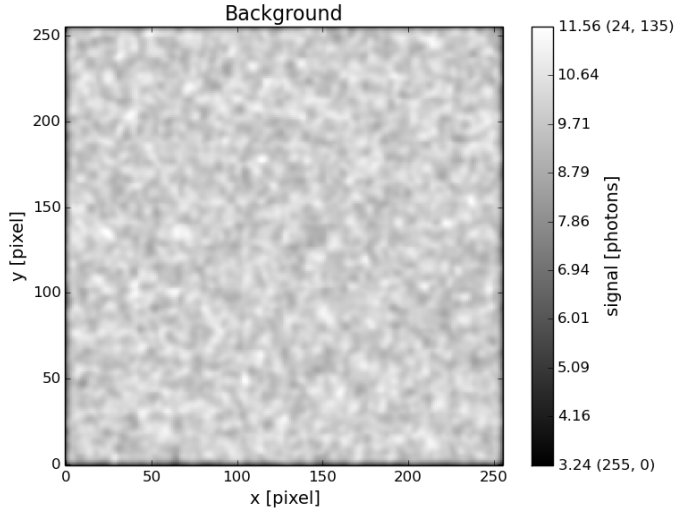

Figure 2.4.: The sky background simulated by *StarSim* for a mean background signal of 10 ph s^{-1} rms and an exposure time of one second. The convolution was carried out with a Gaussian PSF model that features a FWHM of 5×5 px.

2.5. Bias and Read Noise

The bias of the CCD camera can be configured in units of e^-. It is implemented as uniform fixed offset, whereas read noise is modelled by a normal distribution with specified standard deviation in e^- rms (root mean square). This is the on-chip read noise that is usually specified by CCD manufacturers. Additional periodic random noise (off-chip) is introduced during the process of converting the charges in each pixel to digital units. This task is usually carried out by an Analog to Digital Converter (ADC). Any off-chip noise sources like reset-noise (or kTC noise), white noise (or Johnson noise), Flicker noise (or 1/f noise) are currently not treated explicitly in *StarSim*, since they can be strongly reduced by built-in camera electronics. One way to reduce the non-periodic read noise in science frames is simply by averaging over multiple stacked images. These images may be obtained by running *StarSim* several times or by configuring multiple image output before the execution. Alternatively, there exists a built-in process for image calibration, which uses an averaged bias frame (master bias).

2.6. Dark Current and Hot-Pixels

The total dark current in a CCD is comprised of surface dark current, depletion or bulk dark current and diffusion dark current (Widenhorn et al., 2002). Thermal energy is the main source of dark current, thus cooling the camera vastly improves the image quality. Nevertheless, dark current calibrations are often necessary for long-time exposures. In *StarSim*, the total dark current is modelled by applying Poisson statistics similar to Equation (2.6) as well as hot-pixels, which feature exceptional high sensitivities to dark current. *StarSim* allows to specify the amount of hot-pixels together with factors that define their increased sensitivity. The noise originating from dark current and hot-pixels can be almost entirely reduced by the calibration techniques that were implemented in *StarSim*. These techniques include the subtraction of averaged dark frames (master darks). Figure 2.5 shows a typical dark current frame that includes hot-pixels.

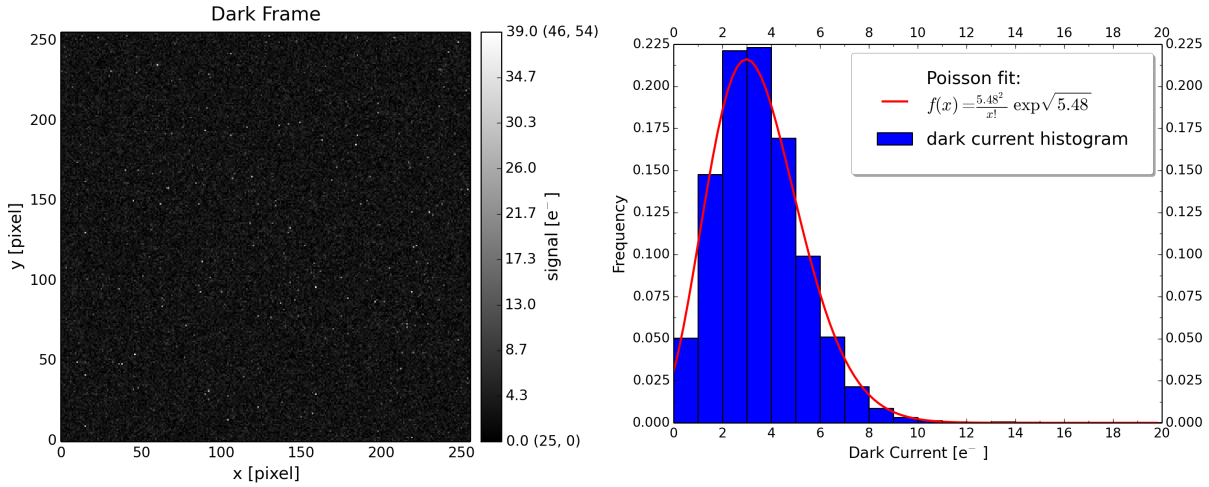

Figure 2.5.: *Left*: A raw frame including dark current and hot-pixels simulated by *StarSim* for an average dark current of three $e^- s^{-1} px^{-1}$ and one second exposure time. The amount of hot-pixels is about one percent. These pixels produce increased dark current with values ranging from two to five times the mean dark current. *Right*: The distribution of dark current values in left panel conforms to a Possion distribution. The simulated dark frame does not contain a bias or read-noise component.

2.7. Variations in Pixel Sensitivity

It is a common issue that astronomical images that are obtained by a CCD feature uneven illumination. Such variations can be caused by non-uniform light transmission through the optics, differences in pixel-gain or even by dust grains on the detector. Another effect is the variation in quantum efficiency, due to temperature difference in individual pixels. In *StarSim*, variations in pixel sensitivities are represented by a flat field model that consists of three distinct components: random *pixel-to-pixel variations*, a *spatial gradient* and *intra-pixel sensitivity variations*. Vignetting that is inevitable in some optical designs, is currently not supported. *Pixel-to-pixel variations* are modelled by a normal distribution by configuring a mean pixel sensitivity between 0 and 1 and a related standard deviation. The *spatial gradient* is considered to be linear on the detector and the configured lower and upper limits may slightly vary if multiple observations are simulated. The gradient's angle can be defined in degrees, where $0°$ indicates horizontal alignment. This feature can be seen clearly in Figure 2.1. *StarSim*'s post-processing techniques include the calibration with a master flat that is composed of multiple single flat fields for the current configuration.

Variations on Intra-Pixel Scale

The detection of a photon by a CCD-pixel depends on the impact location in the pixel (Toyozumi & Ashley, 2013). Therefore, a low sampling rate of stars that is induced by small point spread functions, may lead to significant errors in photometry and astrometry. The intra pixel sensitivity variations are directly related to the CCD electrode structure. A three-phased design

is the most common configuration for CCDs, where three electrodes which are aligned in strips are assigned to each pixel. Figure 2.6 depicts measured photometric sensitivity maps of pixels for different filters. Additionally, simulated intra-pixel sensitivities are shown in the right panel. Here, the sensitivity map is almost uniform for all investigated pixels. *StarSim* is capable of simulating images with respect to such intra-pixel sensitivities by oversampling the images during the generation process. In fact, an oversampled observation is multiplied with a high resolution flat field of equal size that contains the intra-pixel sensitivity variations. After the application of the flat field the image is downscaled to the final image size. Hereby, the factor of oversampling is defined by the resolution of the intra-pixel sensitivity map.

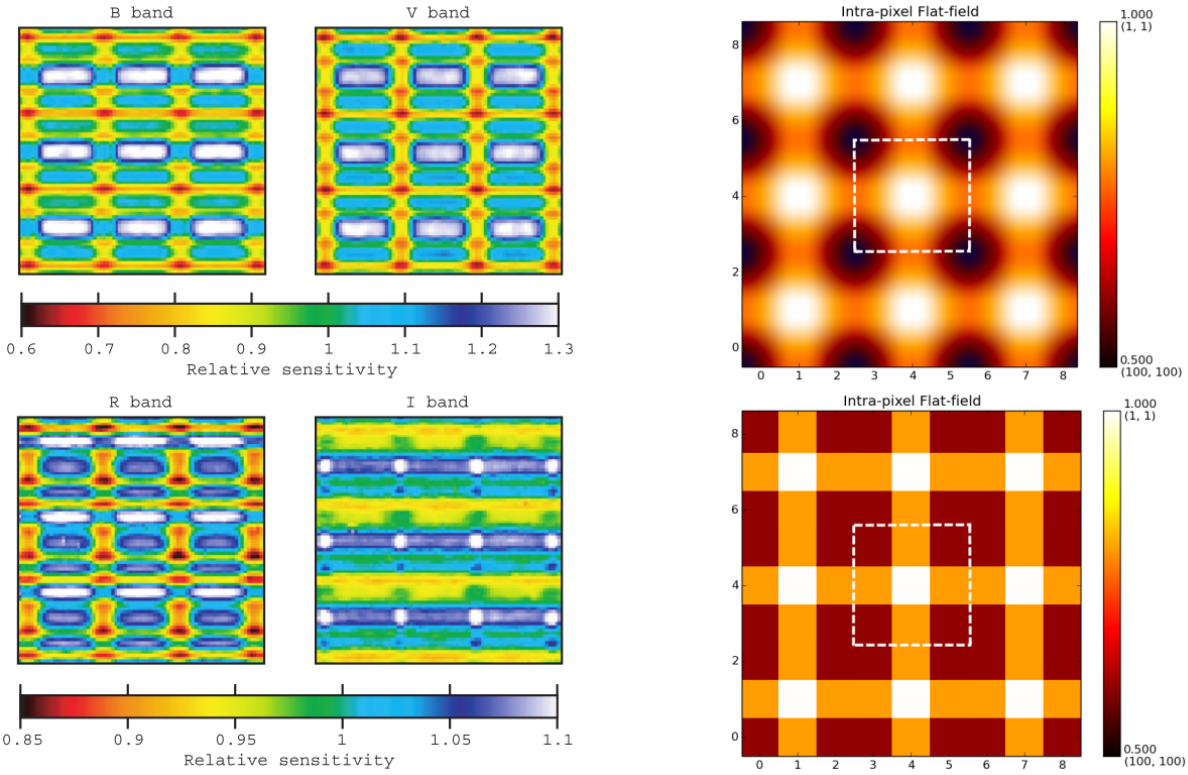

Figure 2.6.: *Left*: The photometric sensitivity map for a 3.2×3.2 pixel array for different filters (Toyozumi & Ashley, 2013). *Right*: A sample intra-pixel flat field for a 3×3 pixel grid generated by *StarSim*. The upper panel illustrates a sinc interpolation of the lower panel. The size of a single pixel is illustrated by the area of the dashed white frame. Each pixel was oversampled with 3×3 intra-pixel values in order to model the photometric sensitivity map. Colour bars indicate relative sensitivities.

2.8. Glitches and Cosmic Rays

Astronomical images obtained by CCD detectors may contain unwanted bright spots that affect single or multiple pixels (see Figure 2.1). These phenomena are described by the term 'glitches'. In general any short-lived fault in an electronic system may be referred to as a glitch. However, this thesis particularly refers to glitches that originate from cosmic ray hits. Those include high-energetic particles produced by our Sun as a by-product of stellar nucleosynthesis and by solar flares. However, cosmic rays with highest energies originate in extrasolar sources like supernovae, Active Galactic Nuclei (AGN), Quasars, Pulsars and others. These cosmic rays are mainly composed of protons and alpha particles and are called 'primaries' if no interaction with other particles took place. If such particles pass interstellar gas or Earth's atmosphere, they interact with atoms and trigger decay processes. The result of such processes are mainly Muons which reach Earth's surface as 'secondaries'. Cosmic ray hits on detectors usually release a large number of electrons, thus they significantly affect image quality. Ionizing energy can be deposited in CCDs by x-rays, γ-rays, electrons, protons, α-particles and heavier ions (Esposito, 2002). Charge-less particles like neutrons do not trigger such effects. In general thin CCDs are less affected by cosmic rays hits than thicker detectors (Kitchin, 2003), as the path length of the particle penetrating the detector is crucial. Such hits may result in glitches, in form of unwanted signal in single, or multiple pixels that are located adjacent to the point of impact. Typical multi-pixel characteristics are trails appearing as straight lines or curve-like shapes on images (Groom, 2004). These structures are caused by very small impact angles, which are elongated nearly parallel to the CCD surface.

Figure 2.7 illustrates the cosmic ray flux as a function of kinetic energy. Cosmic rays cover a broad energy spectrum ranging from 10^8 to 10^{20} eV. An upper limit for ultra-high-energy comic rays is defined by the GZK cutoff (Greisen, 1966). The cosmic ray flux can be of heliospheric, galactic or extragalactic nature. The flux for energies between 10^8 and 10^{10} eV is dominated by solar cosmic rays that originate in coronal mass ejections (Starodubtsev & Usoskin, 2010). Galactic cosmic rays are the main source for the flux within 10^8 and 10^{15} eV and extragalactic cosmic rays with energies $>10^{15}$ eV are comparatively very rare. The dotted line in Figure 2.7 represents a $E^{-2.7}$ power law approximation for the cosmic ray flux as a function of kinetic energy. This approximation overestimates the flux of cosmic rays with lower energies by roughly one decade. Additionally, electrons and protons originate from solar wind and get accelerated to energies between 400 keV and 15 MeV. These are not included in Figure 2.7 (Esposito, 2002).

Further investigations revealed that a direct conversion of the energy of a single cosmic ray to the related electron deposition on the detector is not feasible. The reason is that the electron deposition depends on the length of the cosmic ray's path through the CCD's substrate as well as on the hardware characteristics of the CCD. In order to accurately simulate the impact of

cosmic ray hits on centroiding performance, I estimated the following quantities empirically:

- expected glitch rate outside the South Atlantic Anomaly
- expected glitch rate within the South Atlantic Anomaly
- distribution of deposited electrons per glitch
- ratio of single- to multi-pixel events
- expected distribution for the amount of affected pixels in all events

These estimations are based on observations of HD 189733 which had been obtained by the MOST[1] space telescope and details can be found in Section 4.2.2.

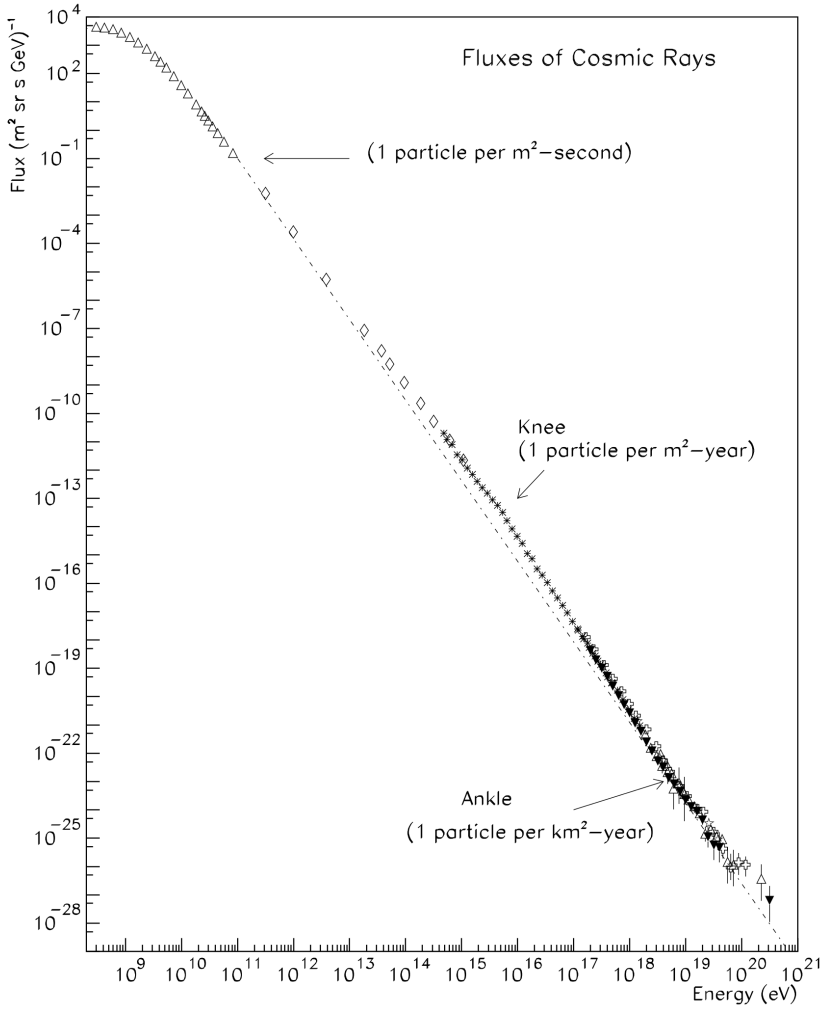

Figure 2.7.: The cosmic ray flux as function of kinetic energy combined for all particles. Combined results from HiRes, Akeno, proton and LEAP are shown. The dottet line represents a $E^{-2.7}$ power-law. (Swordy, 2001)

[1]MOST = Microvariability and Oscillations of STars

2.8.1. Van Allen Radiation Belts and the South Atlantic Anomaly

The South Atlantic Anomaly (SAA) is a region related to the weak geomagnetic field in the South Atlantic Ocean. Its geographical position lies in the south of Brazil and it is illustrated in Figure 2.8. The SAA is spanning from -50° to 0° geographic latitude and from -90° to +40° longitude. Satellites in low-Earth orbit that cross this region in space exhibit an increased radiation flux compared to the radiation level outside the SAA (Heirtzler, 2002). The elongation of Earth's magnetic field gives rise to the inner and outer Van Allen radiation belt. These are toroidal regions containing charged particles that were captured by Earth's magnetic field. The inner belt spans from about 200 to 6 000 km altitude and it mainly contains protons. The formation of these protons is triggered by the Cosmic Ray Albedo Neutron Decay (CRAND) which is the main source for protons in low altitudes with energies up to 30 MeV (Esposito, 2002). Neutrons are created as by-product of cosmic ray interaction with atmospheric nuclei. The radioactive decay of such neutrons leads to protons getting captured by the radiation belt. Additionally, shock waves can inject protons with energies well above 50 MeV. The outer radiation belt is dominated by electrons and it is situated at altitudes between 15 000 and 25 000 km.

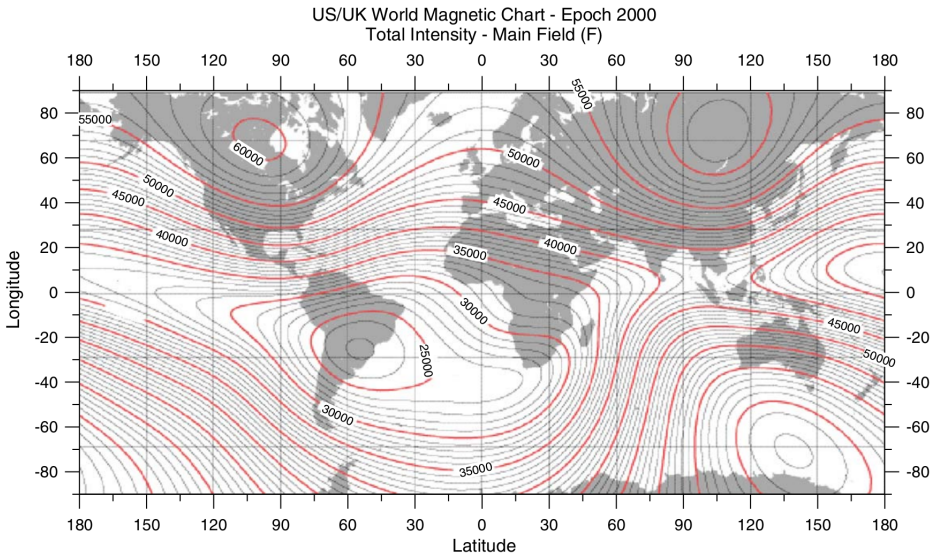

Figure 2.8.: Earth's magnetic field represented by contour lines of units nanotesla (nT). The interval between the gray contours is 1000 nT. The centre of the SAA is located in South Brasil and corresponds to the lowest value of field intensity. Measurements were taken by the Topex/Poseidon spacecraft at an altitude of about 1300 kilometres (Abdu et al., 2005)

.

The SAA occurs where Earth's inner Van Allen belt is closest to the surface approximately at an altitude of 200 kilometres. This in turn is a phenomenon produced by the tilt of 10 degrees of Earth's magnetic dipole with respect to its rotation axis. The higher amount of charged particles within the SAA is accompanied by an increased number of glitch events and therefore signif-

icantly affects image quality. In order to compare the nominal glitch rate with measurements taken inside the SAA, an analysis of glitches in MOST observations was carried out. Both CHEOPS and MOST pass the SAA anomaly in a similar low-Earth orbit. Therefore MOST images were used to estimate the SAA's impact for observations performed by CHEOPS. Further treatment of this topic is provided in Section 4.2.2.

2.8.2. Simulation of Glitches in StarSim

In *StarSim*, glitches are characterised by the following properties: type, spread-type, start/end-points, width, signal and decay-factor. There are currently three supported glitch types that can be simulated. Type '*point*' is indicating a single pixel holding the entire glitch signal. Type '*linear*' represents a line-shaped glitch on the detector, specified by start-point and end-point. Type '*spline*' allows to simulate curve-shaped glitches by defining multiple locations on the detector grid. In this case, the pixels between these locations are calculated using cubic-spline interpolation. Glitches can be added automatically during the image simulation process, based on models which describe their frequency, size and intensity. Such models have been obtained by observations performed by the MOST satellite and are discussed in Section 4.2.2.
The two images in Figure 2.9 represent observations of the G-star HD 189733 which is hosting the hot Jupiter HD 189733 b. The left image shows a real observation performed by the MOST satellite during a SAA passage and the right frame represents a reconstruction created by *StarSim*. The observation includes linear glitches as well as several single-point glitches. This illustration shall stress that *StarSim* is indeed capable of producing output that is largely similar to real observations. In this case, the reconstruction has been applied by configuring fixed glitches, but in principle the exact same image may result in automatic glitch generation if *StarSim* is fed with the appropriate models.

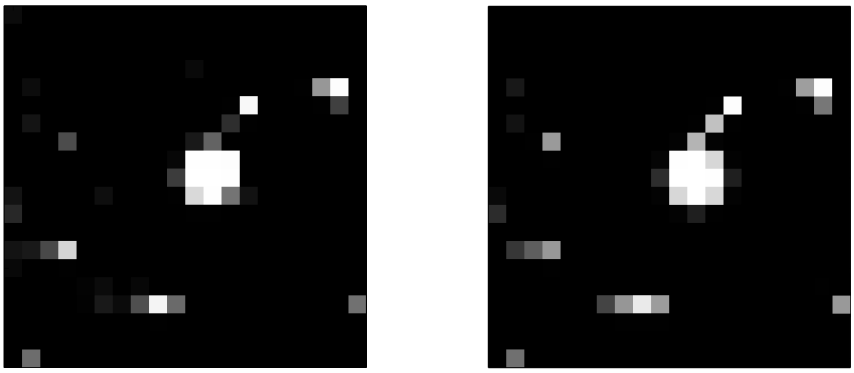

Figure 2.9.: The left image is an observation of HD 189733 carried by the MOST satellite in 2006 during a SAA passage. It is composed of fourteen stacked images whereas a single image was obtained in 1.5 seconds exposure time. The image contains the primary target. It is a 20×20 px sub-area of the full-frame image. The right image depicts a reconstruction of this observation created by *StarSim* by applying a PSF-FWHM of 2×2 px.

2.9. Conclusion

The development of the data simulator *StarSim* was essential in order to investigate centroiding algorithms. The result is a highly sophisticated simulator of astronomical CCD-images, which may be applicable in other research, as well. Therefore, it should be mentioned that *StarSim* currently does not cover all features of a CCD and only simulations of monochromatic light are supported. Front- and back-illuminated CCDs are not distinguished explicitly. Furthermore, the following characteristics were considered to be negligible for centroiding: chip temperature, blooming, fringing, dead pixels, charge transfer inefficiency, read-out speeds, wavelength of incident light and dynamic properties like changes of bias in time. These characteristics may be added manually on demand, since the code of *StarSim* is available as part of the CHEOPS open-source package. Otherwise, they remain untreated until reuse of *StarSim* in future projects. Additional sample images that include the spacecraft's jitter and rotation are provided in the last chapter, such as in Figure 4.22, which represents a simulated observation of the globular cluster Omega Centauri. Only some of *StarSim*'s configuration settings were introduced in this section. For a complete list, that includes a description of all parameters, see Appendix A.1.

3. Centroiding Algorithms

In this chapter, several centroiding methods are presented and their characteristics are discussed based on comprehensive data simulation carried out with *StarSim* (see Chapter 2). Such methods allow to determine the location of stars on an optical sensor. In general, stars are not represented by single pixels on such sensors. Usually they occupy several pixels according to the instrument's point spread function. The term 'centroid' is derived from the later introduced centre of gravity algorithms. However, in this thesis the term 'centroiding' denotes computation of the central position of the stellar spot on the optical sensor. Therefore, the results of all methods which estimate the centre of the stellar spot are labelled as centroids.
Most star tracking algorithms perform centroiding as first task to obtain information for attitude estimation algorithms. Furthermore, if high pointing stability is required in observations, a fine guidance sensor may be used to perform centroiding as an autonomous task. Centroids are then usually required with higher accuracy in comparison to the initial star identification tasks. Various centroiding techniques were analysed for their application in fine-guidance. An implementation of each method is provided in languages Python and C in the Appendix A.3. Some of the following techniques were introduced as part of an earlier work (Ottensamer et al., 2014). However, a complete and more detailed analysis is provided below. Mission specific tests are provided in Chapter 4. Although the main focus of this thesis is on space applications, all the listed centroiding techniques are applicable for ground-based telescopes as well. Magnitudes and terms that are relevant for the analyses in this chapter are introduced below.
The centroid estimation error (CEE) (Vyas et al., 2009b) is introduced to quantify the quality of the computed centroids. It represents the distance between estimated and true centroid and can be defined as follows

$$\mathrm{CEE} = \sqrt{(x_c^* - x_c)^2 + (y_c^* - y_c)^2} \tag{3.1}$$

where, (x_c^*, y_c^*) is the true centroid and (x_c, y_c) is a centroid estimator. It is highly recommended to carry out two steps prior to the execution of centroiding algorithms. The most important step is to define the region of interest (RoI). In literature, this process is often labelled as 'windowing' (Thomas, 2004). In the optimal case, the RoI only contains the target star. However, in particular for large point spread functions, multiple stars may appear in the RoI or even over-

lapping spots may occur in observations of crowded fields. For some algorithms it is crucial that the star is located at the centre of the RoI (see Section 3.5.2). In order to select the RoI, an initial centroid position has to be provided. This information may be obtained by star trackers that carry out attitude estimation algorithms in order to align the telescope to the desired target. After the RoI has been selected, unwanted signal may be removed by thresholding (Lee, 2002). Particularly, for noise-sensitive algorithms, thresholding is a powerful tool to improve the accuracy. These standard procedures are visualised in Figure 3.1. They are indispensable in order to gain accurate position estimations for most centroiding techniques.
The centroiding methods were tested for different signal-to-noise ratios (SNRs). Sample images are illustrated in Figure 3.2. The SNR is defined for an aperture including N pixels, as follows:

$$\mathrm{SNR} = \frac{SQ_e t}{\sqrt{SQ_e t + N(BQ_e t + Dt + N_R^2)}} \tag{3.2}$$

where S is the total target signal in photons per second, Q_e is the quantum efficiency of the detector, t is the exposure time in seconds, B is the average sky background in photons per second per pixel, D is the average dark current in electrons per second per pixel and N_R represents the read noise in electrons per pixel rms (root-mean-square).
In the next section a lower boundary for the accuracy of centroid estimations on optical sensors is derived. In the Sections 3.2, 3.3 and 3.4, eight centroiding algorithms are introduced and an analysis of their performance for different SNRs is presented. Additional features of the centroiding methods are compared to each other in Section 3.5.

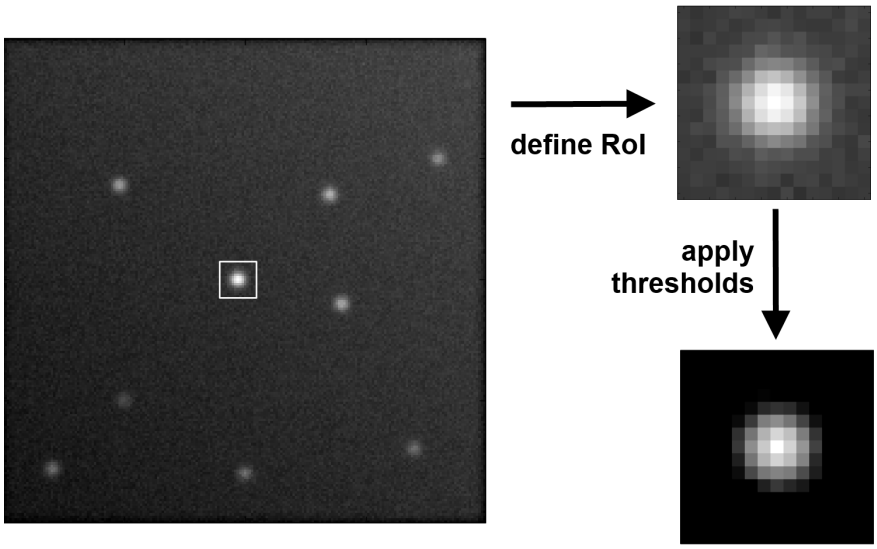

Figure 3.1.: Illustration of standard image manipulating procedures which are carried out prior to the centroiding task. The target star is framed by a white rectangle which determines the region of interest (RoI). Thresholds are then only applied to the RoI. Images were generated with *StarSim*.

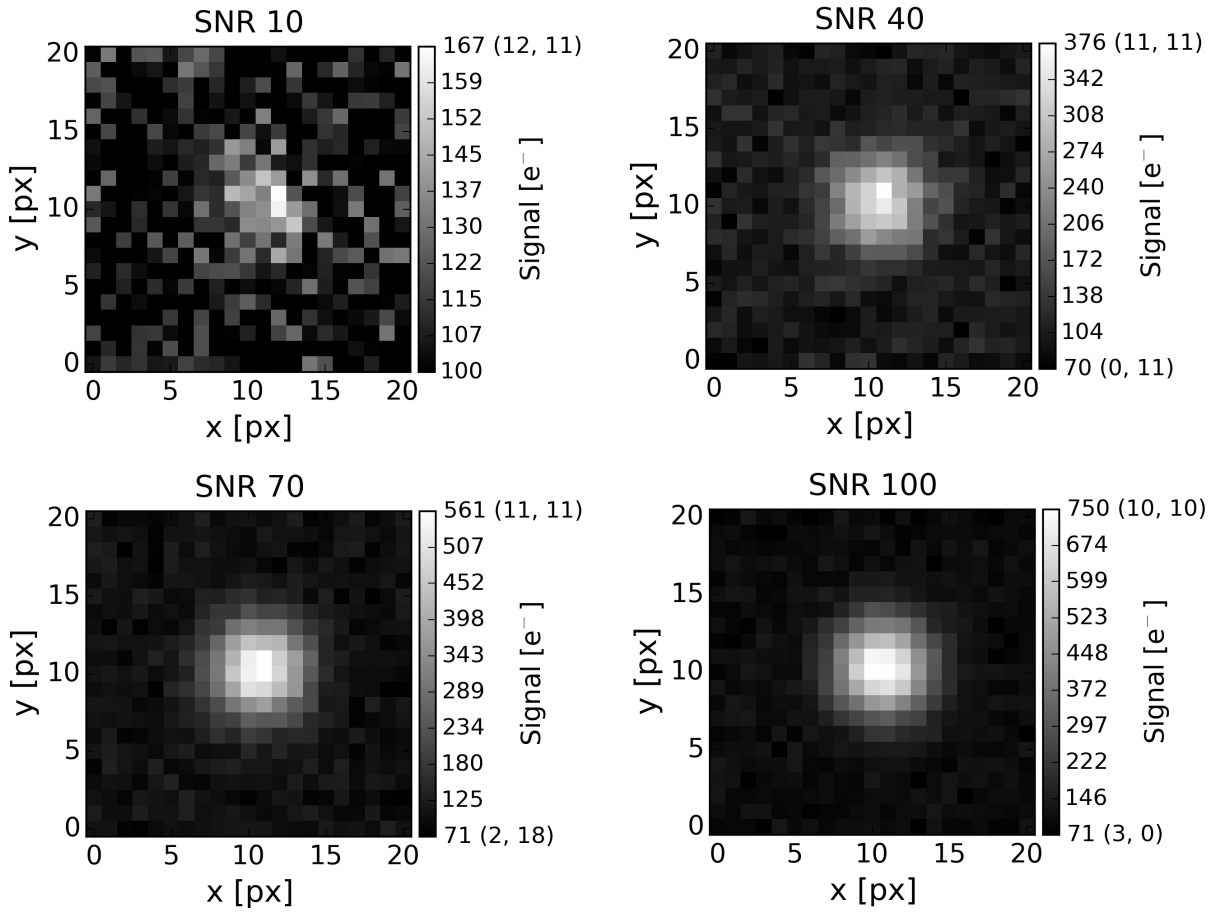

Figure 3.2.: Circular Gaussian spots with a FWHM radius of five pixels for different SNRs. The actual value of the SNR is labelled on the top of each image. Images were generated with *StarSim*.

3.1. Cramer-Rao Bounds

A general limit for optical position estimation methods on CCDs is provided in this section. This limit is applied in later introduced analyses on the performance of the centroiding methods. The optimal centroiding performance is limited by the CCD's read out noise, the shot noise, the sampling rate due to the pixel size as well as by the background. A mathematical description including these characteristics was initially introduced by Winick (1986) and is summarised below. The original work deals with the topic of object localisation on CCDs. However, it was discussed and adapted by several other authors (Chen, 1987; Wernet & Pline, 1993; Chen & Rao, 2009).

A circular Gaussian spot can be described by

$$S(x, y, \epsilon_x, \epsilon_y) = \frac{1}{2\pi\sigma_S^2} \exp\left[\frac{-(x-\epsilon_x)^2}{2\sigma_S^2}\right] \exp\left[\frac{-(y-\epsilon_y)^2}{2\sigma_S^2}\right] \tag{3.3}$$

where ϵ_x and ϵ_y represent the centre of the spot. Equation (3.3) can be split up into a combination of two one-dimensional Gaussians

$$S(x, y, \epsilon_x, \epsilon_y) = S(x, \epsilon_x)S(y, \epsilon_y) \tag{3.4}$$

where

$$S(x, \epsilon_x) = \frac{1}{\sqrt{2\pi\sigma_S^2}} \exp\left[\frac{-(x-\epsilon_x)^2}{2\sigma_S^2}\right]$$
$$S(y, \epsilon_y) = \frac{1}{\sqrt{2\pi\sigma_S^2}} \exp\left[\frac{-(y-\epsilon_y)^2}{2\sigma_S^2}\right]$$

Winick (1986) derived the following Cramer-Rao lower bound, which limits the variance of the unbiased position estimator $\hat{\epsilon}_x$.

$$E[(\hat{\epsilon}_x - \epsilon_x)^2] \geq \left[\lambda_S \left[\sum_{ij} \frac{[g_i'(\epsilon_x)g_j(\epsilon_y)]^2}{g_i(\epsilon_x)g_j(\epsilon_y) + \lambda_N/\lambda_S} - \frac{\left[\sum_{ij} \frac{g_i'(\epsilon_x)g_j(\epsilon_y)g_i(\epsilon_x)g_j'(\epsilon_y)}{g_i(\epsilon_x)g_j(\epsilon_y) + \lambda_N/\lambda_S}\right]^2}{\sum_{ij} \frac{g_i(\epsilon_x)g_j'(\epsilon_y)}{g_i(\epsilon_x)g_j(\epsilon_y) + \lambda_N/\lambda_S}}\right]\right]^{-1} \tag{3.5}$$

with

$$g_i(\epsilon_x) = \int_{x_i - \frac{\Delta x}{2}}^{x_i + \frac{\Delta x}{2}} S(x, \epsilon_x)\,\mathrm{d}x \qquad g_i'(\epsilon_x) = \tfrac{\partial}{\partial \epsilon_x} g_i(\epsilon_x)$$
$$g_j(\epsilon_y) = \int_{y_i - \frac{\Delta x}{2}}^{y_i + \frac{\Delta x}{2}} S(y, \epsilon_y)\,\mathrm{d}y \qquad g_j'(\epsilon_y) = \tfrac{\partial}{\partial \epsilon_y} g_i(\epsilon_y) \tag{3.6}$$

λ_S is the number of photons detected by the CCD during the integration, λ_N represents the average number of electrons per pixel produced by the dark current including read out noise. Δx is the physical size of a pixel and is set to one in order to get results in unit pixel. The functions g_i and g_j are used to discretise the function $S(x, y, \epsilon_x, \epsilon_y)$. Equation (3.5) represents the minimum mean squared error of for $\hat{\epsilon}_x$. Since the Gaussian spot is considered to be circular, the estimated errors in x and y dimension are assumed to be equal. Therefore, the result can be expressed as centroid estimation error by

$$\mathrm{CEE} = \sqrt{2} \cdot \mathrm{E}[(\hat{\epsilon}_\mathrm{x} - \epsilon_\mathrm{x})^2] \tag{3.7}$$

Figure 3.3 shows the dependency of the lower bound of the centroid estimation error on the size of the point spread function for different signal-to-noise ratios. Large image spots lead to

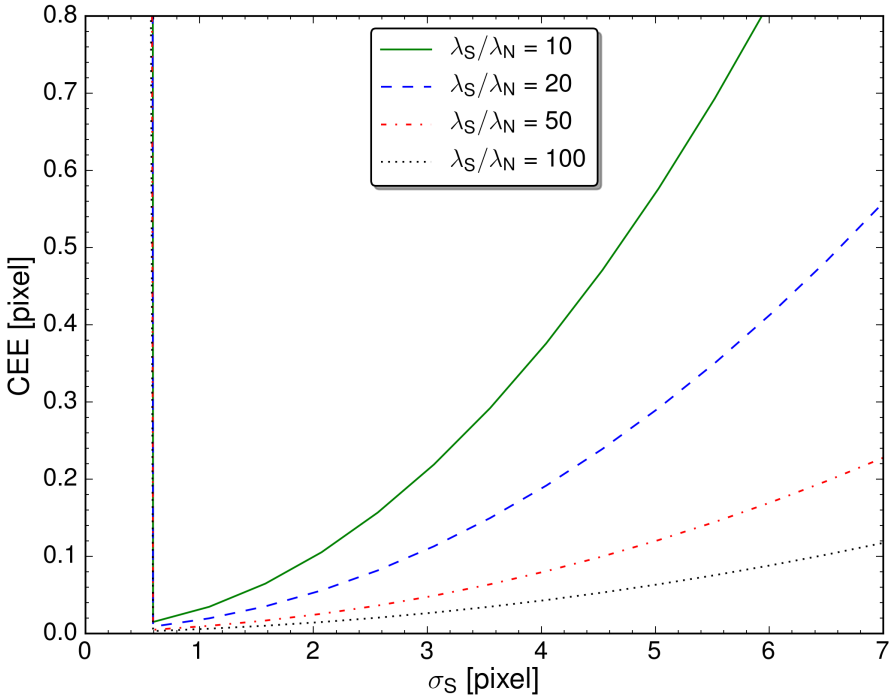

Figure 3.3.: The impact of the spot size on the centroid estimation error corresponding to the Cramer-Rao bound for different signal-to-noise ratios. The centroid was placed at $(\epsilon_x,\epsilon_y) = (20, 20)$ on a 40×40 pixel array. λ_S was changed to obtain different signal-noise ratios and λ_N was kept constant.

a lower SNR per pixel, thus the centroid estimation error increases. If the spread of the spot falls below half the size of a pixel, accurate centroid estimation on intra-pixel scale is not possible. The magnitude of the centroid estimation error depends on the actual value of λ_S rather than on the noise ratio λ_S/λ_N. Figure 3.4 depicts the minimal centroid estimation error for different signal-to-noise ratios. It emphasizes the dependency of the centroid estimation error on the SNR as well as on the spot size. Background signal is not included. Figure 3.5 depicts the centroid estimation error for different intra-pixel positions. The sinusoidal characteristic of the relative CEE is related to the integration limits in Equation (3.6), as they change if the spot is moved along the detector's axis. It can be seen that this effect, which occurs between pixel edge and pixel centre, is only of order 10^{-9} and therefore it is negligible for the centroiding performance limits. This does not imply that different intra-pixel positions do not change the centroid quality in general, as the intrinsic characteristics of a centroiding method may lead to better performance if the spot's peak is centred in a pixel. In addition, intra-pixel flat fields were not taken into account, which indeed have a non-negligible effect on the centroiding performance (see Section 2.7 and Section 3.5.5). The background is not included in Equation (3.5), but it also affects the lower limits for the localisation. The contribution of the background can be eliminated by thresholding methods in most cases. An adaptation of Equation (3.5) which includes the background is provided by Chen & Rao (2009).

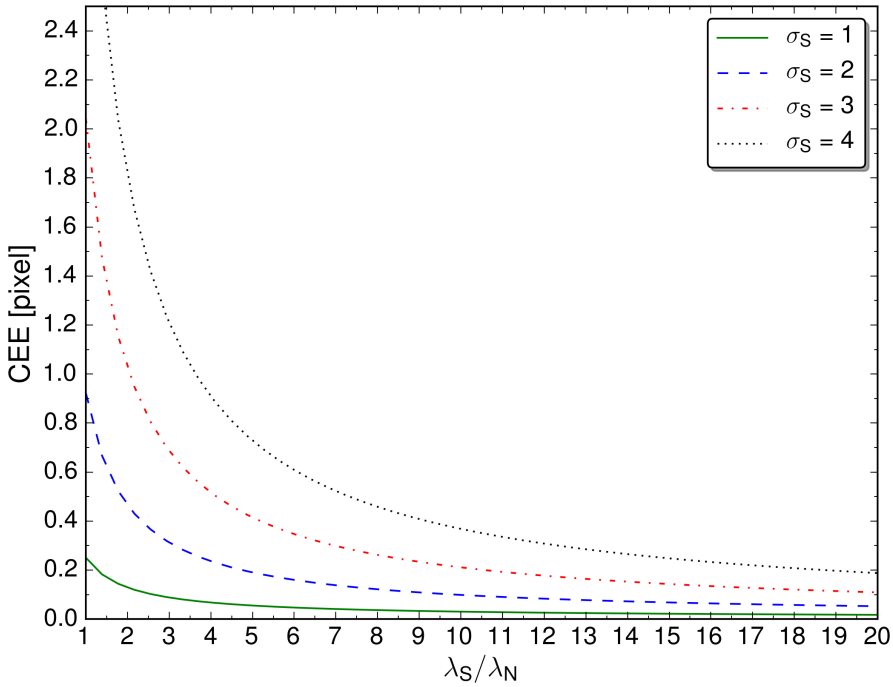

Figure 3.4.: The impact of the signal-to-noise-ratio on the centroid estimation error corresponding to the Cramer-Rao bound for different spot sizes. Poisson noise is not included in λ_S/λ_N. The centroid was placed at (ϵ_x, ϵ_y) = (20, 20) on a 40×40 pixel array. λ_S was changed to obtain different signal-noise ratios and λ_N was kept constant.

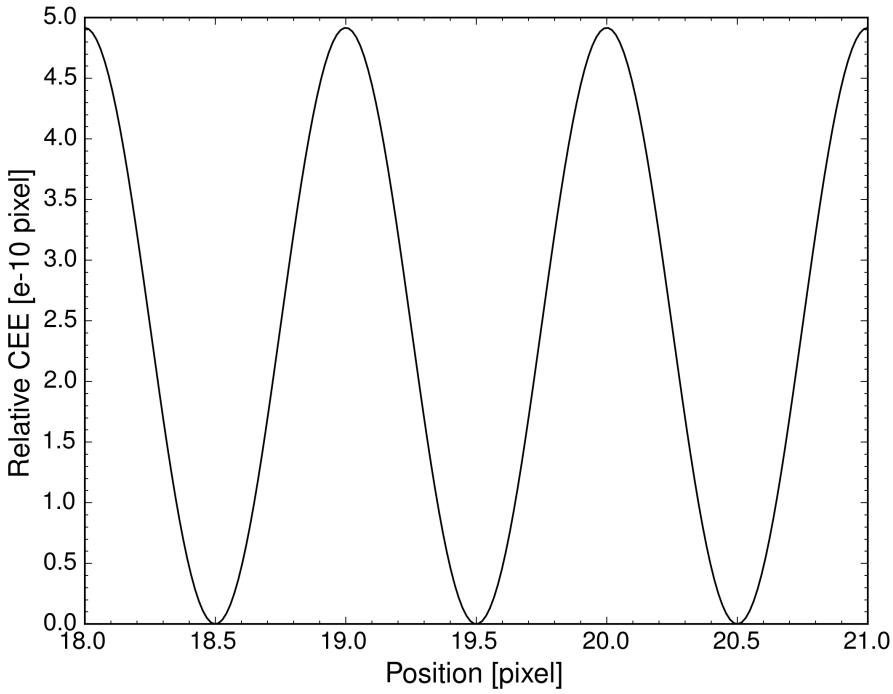

Figure 3.5.: The relative centroid estimation error corresponding to the Cramer-Rao bound for different intra-pixel positions for a spot size of $\sigma_S = 2$ pixel and a signal-to-noise ratio of $\lambda_S/\lambda_N = 10$. The centroid was initially placed at (ϵ_x, ϵ_y) = (18, 20) on a 40 × 40 pixel array and was moved to position (ϵ_x, ϵ_y) = (21, 20), subsequently.

3.2. Centre of Gravity Algorithms

The centre of gravity (CoG) or centre of mass (Uhm et al., 2008; Vyas et al., 2009b,a; Tremsin et al., 2003) represents the simplest method to determine the target position. In this section different variations of the centre of gravity method are discussed.

3.2.1. Standard Centre of Gravity (CoG)

The centre of gravity algorithm for object localisation is named by the eponymous principle in mechanics. In general, the centre of gravity determines the central location of the mass distribution of an object. The same principle can be used for computing the centroid of a stellar spot on a detector. Instead of weighting mass particles, the charges collected by each pixel are weighted by the pixel positions on the detector. Therefore, the CoG can be computed by

$$x_c = \frac{\sum_{i=0}^{N}\sum_{j=0}^{M} I_{i,j}\ x_i}{\sum_{i=0}^{N}\sum_{j=0}^{M} I_{i,j}}, \qquad y_c = \frac{\sum_{i=0}^{M}\sum_{j=0}^{N} I_{j,i}\ y_i}{\sum_{i=0}^{N}\sum_{j=0}^{M} I_{i,j}}, \tag{3.8}$$

where N and M are the number of pixels in a single row and column respectively. The pixel coordinates are represented by (x_i, y_i) and the charge held by each pixel is defined by $I_{i,j}$. For determining the centroid, the total signal inside the region of interest has to be computed. This may be applied in on-board photometry without introducing extra computation time, if thresholds are applied in advance. This opportunity is further discussed in Section 4.1.

3.2.1.1. Implementation

The standard CoG may be implemented by executing the following steps,

1. Compute the total signal I by summing up all pixel values in the RoI.
2. Iterate over columns and multiply column index with collapsed row signal.
3. Iterate over rows and multiply row index with collapsed column signal.
4. Divide results of steps (2) and (3) by I in order to get x_c and y_c.

Implementations of the standard CoG are provided in the Appendix A.3.1 in languages C and Python.

3.2.1.2. Error Estimation

Error estimation for centre of gravity algorithms is a well-discussed topic (Poyneer et al., 2003; Lee, 2002; Jia et al., 2010; Baker & Moallem, 2007; van Assen et al., 2002). The following

estimation was derived by van Assen et al. (2002) and describes the variance of the estimated position error with respect to a Gaussian noise component.

$$\mathrm{var}(c(x,y)) \approx \left[\left(\frac{\sigma_N^2 \sum_{x,y} x^2}{(N\,\hat{\mu}\,\hat{x}_{CoG})^2} + \frac{\sigma_N^2}{N\hat{\mu}^2} \right) \hat{x}_{CoG}^2,\ \left(\frac{\sigma_N^2 \sum_{x,y} y^2}{(N\,\hat{\mu}\,\hat{y}_{CoG})^2} + \frac{\sigma_N^2}{N\hat{\mu}^2} \right) \hat{y}_{CoG}^2 \right] \tag{3.9}$$

where $c(x,y)$ is an unbiased estimator of the true centroid, $(\hat{x}_{CoG}, \hat{y}_{CoG})$ is the estimated centroid provided by the CoG algorithm, $\hat{\mu}$ is the estimated average signal in the region of interest, N is the number of pixels and σ_N represents the total Gaussian noise per pixel. I estimated σ_N with the simulated read noise and dark current in later analyses. In the derivation of Equation (3.9) a constant pixel-weighting for CoG computation was considered, but van Assen et al. (2002) did not take Gaussian weighting functions into account. Thus, Equation (3.9) is only applicable for error estimation of CoG and IWC. Error estimation for the later introduced WCoG and IWCoG is discussed in Section 3.2.3.2.

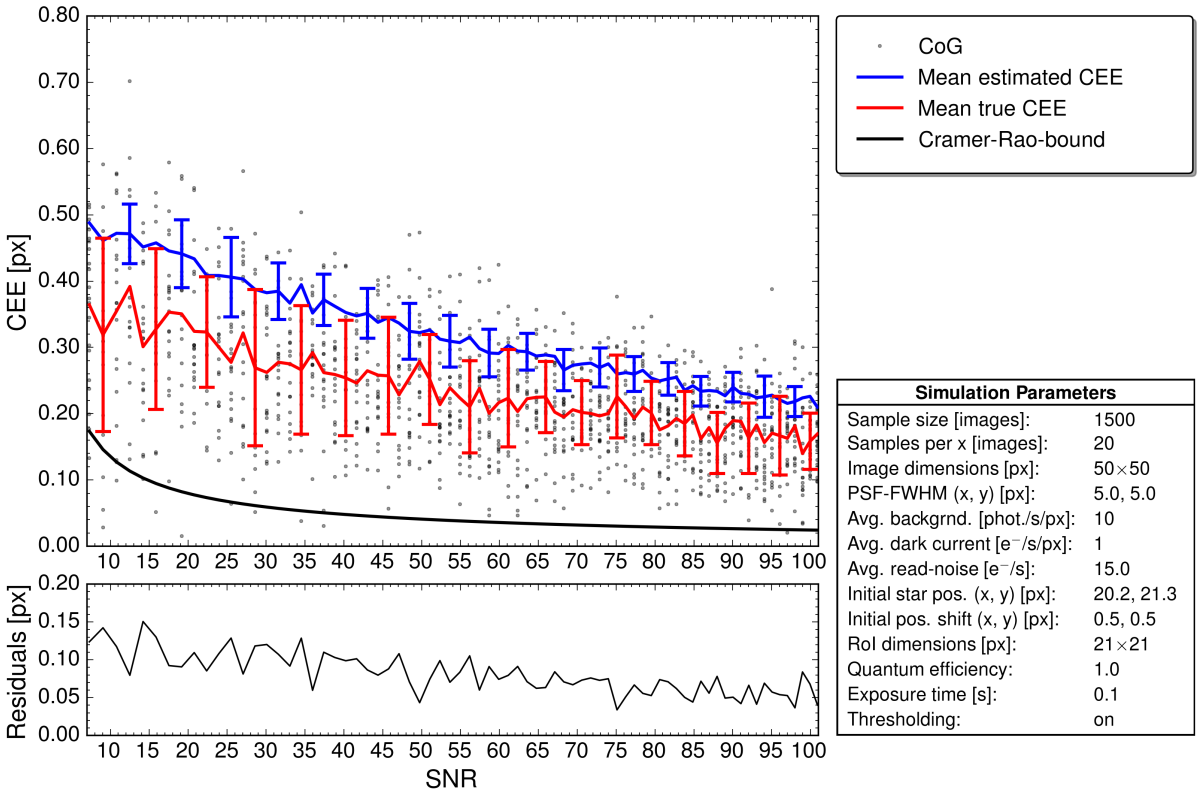

Figure 3.6.: The dependency of the CoG algorithm on the signal-to-noise ratio. A Monte-Carlo simulation carried out by *StarSim* produced a sample of 1 500 images (see Figure 3.2). The transparent black dots indicate the computed true CEE per image. The blue line indicates the estimated mean CEE based on error estimation described in Section 3.2.1.2. Error bars represent the standard deviation. The residuals depict the difference between mean estimated CEE and mean true CEE.

3.2.1.3. Performance

Monte-Carlo simulations were carried out in order to produce large samples of images. The performance of CoG was tested for different SNRs. Figure 3.6 reveals that the SNR plays an important role for the CoG method. An increase of the SNR by a factor of ten halved the true CEE as well as its spread. Although some results are close to the optimum, the mean true CEE does not converge against the Cramero-Rao bound in this simulation. The estimated CEE was computed with Equation (3.9). The systematic difference between mean estimated and mean true CEE of about 0.1 px is revealed in the residuals. This discrepancy is caused by a conservative noise estimation of σ_N^2 in Equation (3.9). Furthermore, it can be seen that the spread size of the mean estimated CEE is too small for low SNRs. This is caused by the missing photon noise component in the error estimation, as van Assen et al. (2002) only considered Gaussian white noise.

3.2.1.4. Summary

The performance of the CoG was tested and compared to other methods under various conditions (see Section 3.5). It turned out that the CoG is highly sensitive to any kind of noise. Therefore, it should not be applied if centroid information is required with high accuracy in the presence of a low SNR. In fact, Figure 3.6 depicts that the performance of CoG is far below the optimum, which is defined by the Cramer-Rao bound. Although, the CoG does not require an initial centroid estimation for its computation, this kind of estimation is required in order to define the RoI within the original image (see Figure 3.22). A few things must be considered before selecting such region. CoG is only capable of providing accurate results if the RoI contains the entire stellar spot. Furthermore, the central region of the spot should be situated at the centre of the RoI. Therefore, it is crucial that the dimensions of the RoI are odd integers. Another important fact is that the RoI should not contain a second star, which might be impossible if a crowded field is observed. In the presence of a second star, the centroid estimation of CoG is located between both stars, but it will be closer to the brighter star in general. Thresholding is also an inevitable process too keep the CEE low, in the presence of background signal (see Figure 3.23). In principle any signal that is not originating from the target distorts the centroid estimation provided by CoG if it is not thresholded. Thus, cosmic ray hits such as those illustrated in Figure 3.24 may cause large CEEs, if centroids are computed via CoG. In comparison to other algorithms the standard CoG is a rather simple method featuring a low processing time (see Figure 3.28). I recommend to calibrate images with dark frames as well as flat fields prior to the application of CoG. In particular, the reduction of a sensitivity gradient, such as the one illustrated in Figure 2.1, is indispensable.

3.2.2. Weighted Centre of Gravity (WCoG and IWC)

The weighted centre of gravity (WCoG) can be computed by extending the standard CoG by a weighting function, $W_{i,j}$ (Nicolle et al., 2004; Vyas et al., 2010; Vyas & Vohnsen, 2013). This leads to a modified version of Equation (3.8).

$$x_c = \frac{\sum_{i=0}^{N}\sum_{j=0}^{M} I_{i,j}\, W_{i,j}\, x_i}{\sum_{i=0}^{N}\sum_{j=0}^{M} I_{i,j}}, \qquad y_c = \frac{\sum_{i=0}^{M}\sum_{j=0}^{N} I_{j,i}\, W_{i,j}\, y_i}{\sum_{i=0}^{N}\sum_{j=0}^{M} I_{i,j}} \tag{3.10}$$

In general, it's best if the weighting function resembles the shape of the target spot. Thus, the point spread function represents an excellent weighting function. In most cases, the spot features a Gaussian shape and therefore an appropriate weighting function is given by Equation (2.1). In order to create such a weighting function, an estimation of the target position is required. This *initial* centroid estimation may be supplied whether by star finding algorithms (pre science) or by previous executions of the fine-guidance task during science mode. A Gaussian weighting function may not be applicable in the case of a highly distorted PSF, such as it is the case in the CHEOPS mission (see Section 4.2.3). A special case of the WCoG is the intensity-weighted centre of gravity (IWC), where the intensity function, $I_{i,j}$, is applied as weighting function (Vyas & Vohnsen, 2013). For this particular case, no initial centroid estimation is required. Implementations of both versions (WCoG and IWC) are provided in the Appendix A.3.2. A two-dimensional Gaussian function was applied to generate the discrete weighting function. Alternatively, a weighting 'template' (e.g. PSF image) may be provided and reused for the sake of lower computation times. In this case, the initial centroid estimation is used to position the template inside the coordinate system of the detector. Multiple versions of PSF-templates are required to cover intra-pixel locations. The amount of pre-computed templates depends on the desired centroid accuracy. Prior to the execution of the centroiding task, the template according to the given centroid estimation has to be chosen. One possibility to obtain such templates is to downscale high resolution PSF-models to the detector resolution.

3.2.2.1. WCoG - Implementation

Implementations of WCoG are provided in the Appendix A.3.2 in languages C and Python. The WCoG may be implemented by executing the following steps.

1. Obtain an initial centroid estimation x_0, y_0 (e.g. from previous centroiding runs).
2. Compute the weighting function $W_{i,j}$ with respect to the initial centroid estimation. Alternatively, select a pre-computed template for W, which must be of same size as I.
3. Iterate over all pixels inside the RoI and apply the weighting $I_{i,j} := W_{i,j} \cdot I_{i,j}$.
4. Proceed with the standard CoG procedure as described in Section 3.2.1.1.

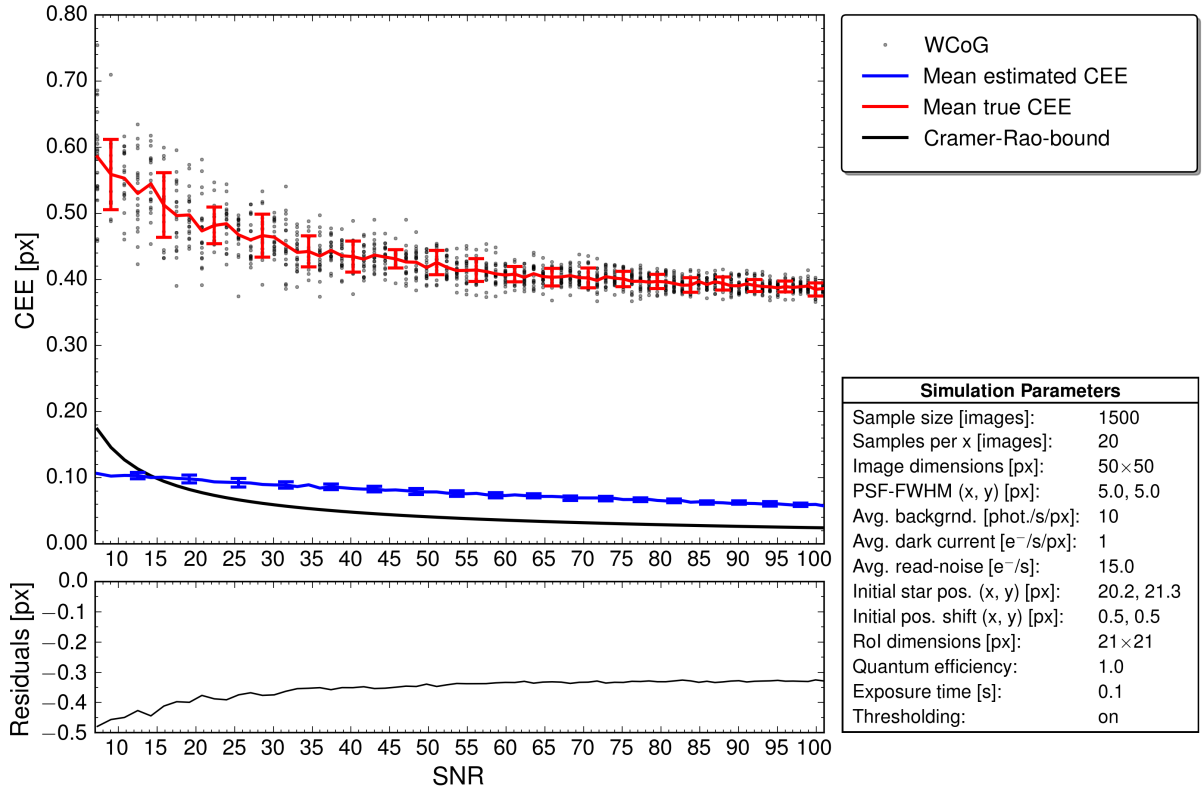

Figure 3.7.: The dependency of the WCoG algorithm on the signal-to-noise ratio. A Monte-Carlo simulation carried out by *StarSim* produced a sample of 1 500 images (see Figure 3.2). The transparent black dots indicate the computed true CEE for each image. The blue line depicts the estimated mean CEE based on error estimation described in Section 3.2.3.2. Error bars represent the standard deviation. The residuals depict the difference between mean estimated CEE and mean true CEE.

3.2.2.2. WCoG - Performance

Monte-Carlo simulations were carried out in order to produce a large sample of images. The performance of WCoG was tested for different SNRs. Equation (2.1) represents the normalised two-dimensional Gaussian function with a FWHM at five pixels in x and y. This function was applied as weighting function in all simulations presented in this chapter. The initial centroid estimation required for the weighting function was displaced by 0.5 pixel in x and y, which corresponds to a total CEE of 0.71 pixel as start condition. The magnitude of the computed true CEE in Figure 3.7 is a conspicuous feature of the WCoG algorithm. Error saturation occurs approximately at a CEE of 0.4 pixel, which is far away from the Cramer-Rao bound. This implies that even small displacements below one pixel in the initial centroid estimation are sufficient to distort results of WCoG. This particular problem may be addressed by performing multiple iterations of WCoG and is discussed further in Section 3.2.3 as a distinct centroiding method (IWCoG). Raising the SNR by a factor of ten reduced the mean error by only thirty percent. At the same moment, the spread of the true error values was narrowed down to one fourth. The mean estimated CEE illustrated in Figure 3.7 is based on the error estimation

presented in Section 3.2.3.2. It can be seen that this kind of error estimation is not applicable for WCoG if start conditions differ from the true centroid.

3.2.2.3. WCoG - Summary

The performance of the WCoG was tested and compared to other methods under various conditions (see Section 3.5). A two-dimensional Gaussian function served as weighting function in all simulations. WCoG is much less sensitive to the presence of noise than the standard CoG, due to the introduction of a weighting function. This weighting function acts like a filter that truncates all signal located far away form the expected centroid position. However, this additionally required centroid estimation represents a critical new error source. Even small deviations from the true centroid, cause a non-convergence behaviour of WCoG, which is clearly illustrated in Figure 3.7. Therefore, I do not recommend its application if high precision centroid estimations are required. Nevertheless, by performing multiple iterations of WCoG remarkable results can be achieved. This is further discussed in Section 3.2.3. In general, WCoG is capable of reaching the Cramer-Rao bound for SNRs below 100 if an optimal initial centroid estimation is provided.
Thresholding is recommended for WCoG, although the Gaussian weighting function reduced most of the background signal (see Figure 3.23). A significant advantage of WCoG compared to the standard CoG is the fact that the RoI needs not be centred on the target spot. However, this benefit is compensated by the newly introduced dependence on the position offset (see Figure 3.22). Nevertheless, WCoG is also applicable in the presence of a second star inside the RoI as long as the position offset is close to the target. Furthermore results were not affected by cosmic ray hits that occurred outside the target spot, as the weighting function eliminated their signal. This is only true if the weighting function is not overlapping with the impact location. If the weighting templates are not precomputed, the processing time of WCoG equals multiple computations of CoG (see Figure 3.28). I recommend to calibrate images with dark frames as well as flat fields prior to the application of WCoG, although the impact of variations in pixel sensitivities is much lower in comparison to the standard CoG (see Figure 2.1). In the end, I would not recommend to implement WCoG in most cases, due to the extreme dependency on the position offset. A different implementation of WCoG that applies a Gaussian weighting function, which is bigger than the actual target spot, may correct this problem. Nevertheless, this would increase the impact of background, cosmic ray hits and additional stars inside the RoI. Thus, such an adaptation of WCoG is not taken into further consideration.

3.2.2.4. IWC - Implementation

The IWC may be implemented by executing the following steps.

1. Iterate over all pixels and apply the intensity-weighting by computing $I_{i,j} := I_{i,j} \cdot I_{i,j}$.
2. Proceed with standard CoG procedure as described in Section 3.2.1.1.

Implementations of IWC are provided in the Appendix A.3.2 in languages C and Python.

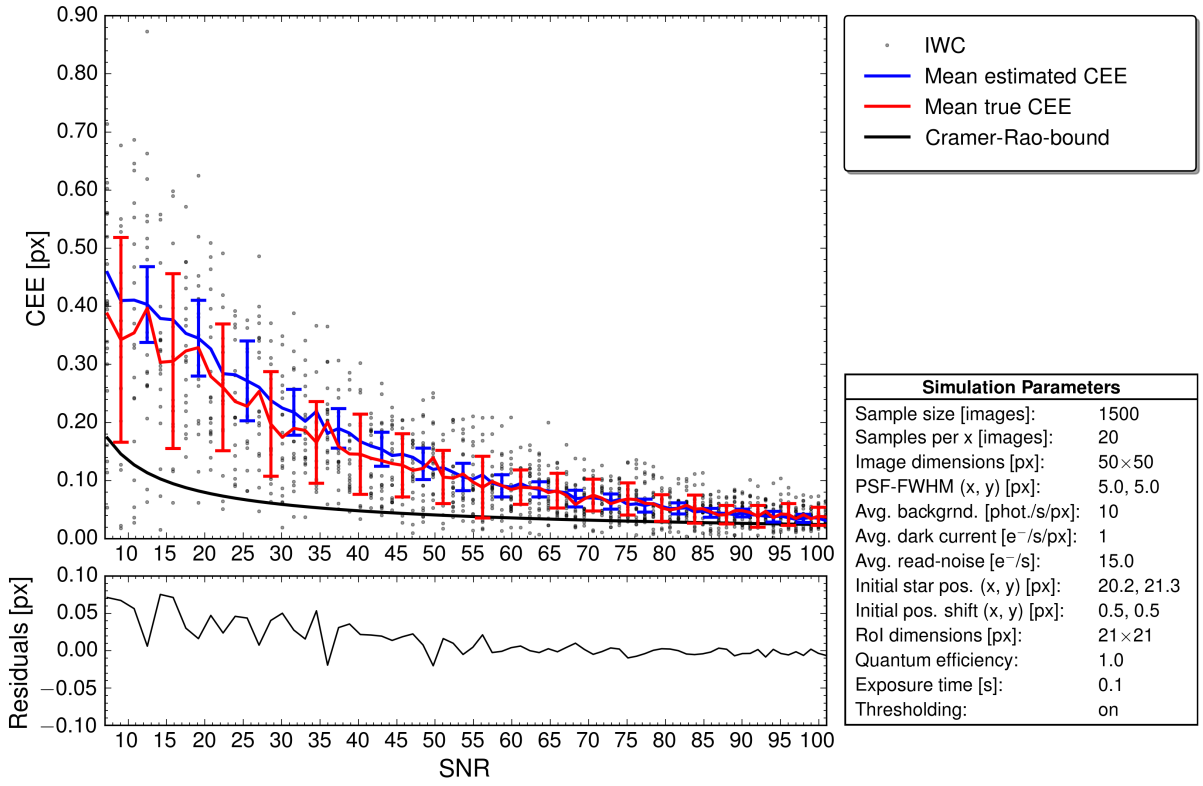

Figure 3.8.: The dependency of the IWC algorithm on the signal-to-noise ratio. A Monte-Carlo simulation carried out by *StarSim* produced a sample of 1 500 images (see Figure 3.2). The transparent black dots indicate the computed true CEE for each image. The blue line depicts the estimated mean CEE based on error estimation described in Section 3.2.1.2. Error bars represent the standard deviation. The residuals depict the difference between mean estimated CEE and mean true CEE.

3.2.2.5. IWC - Performance

Monte-Carlo simulations were carried out in order to produce samples containing a large number of images. The performance of IWC was tested for different SNRs. Figure 3.8 reveals the high influence of the SNR on the performance of IWC. An increase of the SNR by a factor of ten lowered the mean true CEE and its spread roughly by a factor of six. It is also noticeable that the mean true CEE is close to the Cramer-Rao bound for slightly higher SNRs (>100). However, the spread of the true errors at low SNRs is high compared to the regular CoG. This is caused by the application of the incoming signal as weighting function, since the whole signal is

squared along with its noise components prior to CoG computation. The residuals in the lower panel illustrate the remarkable quality of the error estimation provided in Section 3.2.1.2. The Gaussian noise component, σ_N, was estimated for the error computation after the weighting function was applied. The difference in the spread size of the mean estimated CEE compared to the mean true CEE is again caused by the missing photon noise component in Equation (3.9). The performance of the IWC changes extremely if non-optimal thresholds were applied. This feature is discussed in Section 3.5.3 in more detail.

3.2.2.6. IWC - Summary

The performance of the IWC was tested and compared to other methods under various conditions (see Section 3.5). In the SNR-analysis the performance was much better compared to the standard CoG. In fact, the centroid estimation was already close to the Cramer-Rao bound at a SNR of 100, as the CEE declined much faster for increased SNRs. Similar to the standard CoG, IWC does not require an initial centroid estimation for its computation. However, it is required for selecting the RoI, which should be centred at the stellar peak. Displacement of the RoI distorts results of IWC, but the introduced error is generally lower compared to standard CoG (see Figure 3.22). The impact of background signal is weaker, as well (see Figure 3.23). However, the application of thresholds is indispensable in order to keep the error low. Furthermore, centroids computed by IWC get distorted if a second object is located inside the RoI. In this case, the computed centroids are closer to the brighter object. Great care also must be taken if cosmic ray hits are expected inside the RoI, as the presence of a glitch may lead to unusable results (see Section 3.5.4). This is a side-effect of using the detected signal as weighting function, because the signal induced by the cosmic ray hit is squared. The performance of the standard CoG depends on the location of the peak position on intra-pixel scale. For IWC, such variations in the performance, between centroids located at the pixel's edge and the pixel's centre, only matter for very low SNRs. The reduction of a sensitivity gradient, such as illustrated in Figure 2.1, is highly recommend (see Section 3.5.5). Anyway, I recommend to calibrate images with dark frames and flat fields prior to the computation of IWC. One of IWC's main benefits is the remarkably low processing time, which is only slightly higher than for the standard CoG.

3.2.3. Iteratively Weighted Centre of Gravity (IWCoG)

Centroid estimations provided by WCoG strongly depend on the distance between the required initial centroid estimation to the true centroid. In order to reduce this effect, multiple consecutive computations of WCoG can be performed, in which the result of each iteration is used for centroid estimation in the subsequent computation. This procedure is called iteratively weighted centre of gravity (IWCoG) (Baker & Moallem, 2007; Vyas et al., 2010; Vyas & Vohnsen, 2013). In fact, by performing multiple iterations, the weighting function W is shifted towards the target position until the point of maximum overlap is reached. The number of required iterations mainly depends on the quality of the provided centroid estimation as well as on the target signal. The change of the CEE for multiple iterations is illustrated in Figure 3.9 for three different initial centroid estimations. The green squares, red circles and blue triangles represent an initial distance of 10, 8 and 5 pixels to the true centroid, respectively. By comparing convergence behaviour of shift-10 and shift-5, it is shown that starting twice as far from the true centroid doubled the number of iterations that were required to reach the same accuracy. This is only true if the same weighting function is applied and if it overlaps with the Gaussian spot in both cases. Figure 3.10 depicts the IWCoG's dependency on the target signal. Iterations were carried out until the CEE was below 0.1 pixels. A fixed shift of 10 pixels was applied to the start positions. Results show that for extremely faint star signals, a much higher number of iterations is required to retrieve the true centroid position. Additionally, the level of the background signal has a slight impact on the number of iterations as well. Due to this diversity of parameters the optimal number of IWCoG iterations is mission specific. I simulated worst-case scenarios for the EChO mission in order to define the number of iterations (see Section 4.1.4). For these simulations, the faintest star in the target list was chosen. Furthermore, the initial centroid estimations provided by the star trackers as well as the jitter of the spacecraft were considered.

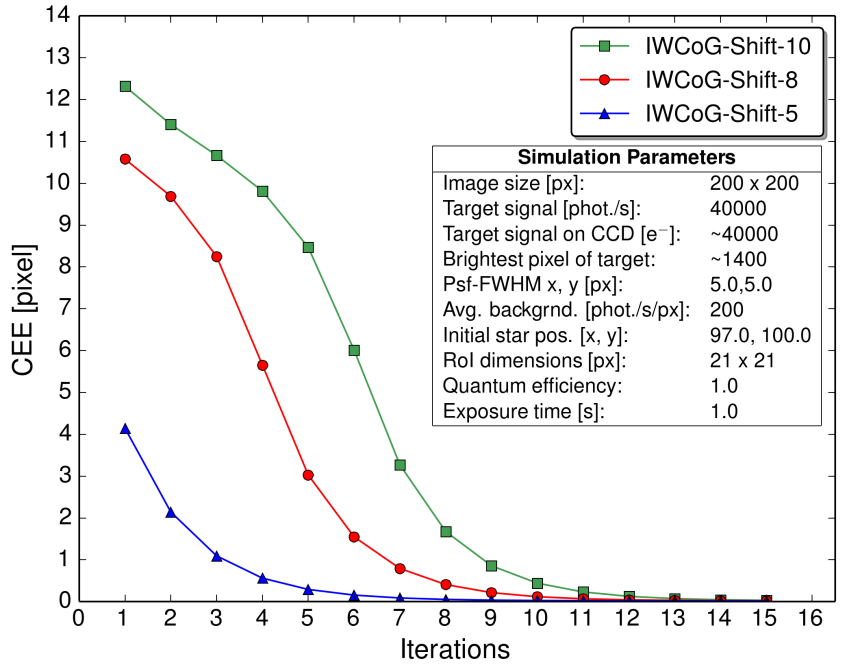

Figure 3.9.: Number of IWCoG iterations required to retrieve the true centroid for different start positions. The different lines represent different position estimations. Green squares, red circles and blue triangles indicate an initial shift to the true centroid of 10, 8 and 5 pixels. The simulated point spread function is illustrated in 2.3. Intensity of target and noise was constant in all three simulations.

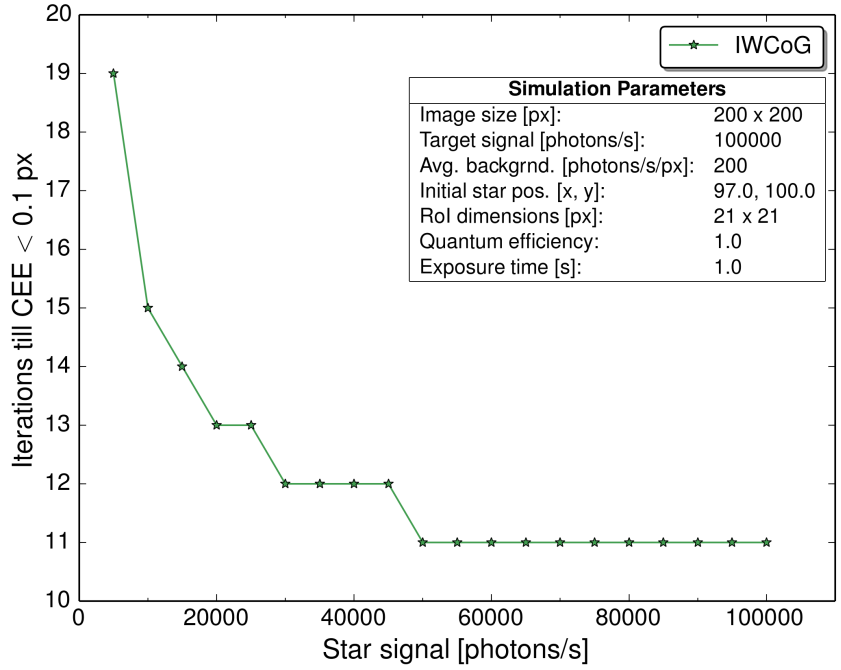

Figure 3.10.: Number of iterations necessary to reach CEEs below 0.1 pixels for different star signals. The shift in the initial centroid estimation was set to 10 pixels. The applied point spread function is illustrated in 2.3.

3.2.3.1. Implementation

The WCoG may be implemented by executing the following steps,

1. Obtain an initial centroid estimation x_0, y_0 (e.g. from previously executed centroiding runs).
2. Compute the weighting function $W_{i,j}$ with respect to x_0 and y_0. Alternatively, select a pre-computed template for W. Note that W must be of same size as I.
3. Iterate over all pixels and apply the weighting by $I_{i,j} = W_{i,j} \cdot I_{i,j}$.
4. Proceed with standard CoG procedure as described in Section 3.2.1.1.
5. Stop the execution if the maximum number of iterations is reached. Otherwise continue with step (6) or go to step (1).
6. *Optional:* Compare the values of x_c and y_c from current and last iteration. Stop the execution if the differences are lower than the predefined accuracy limits, otherwise go to step (1).

Implementations of IWCoG are provided in the Appendix A.3.3 in languages C and Python.

3.2.3.2. Error Estimation

Error estimation for WCoG and IWCoG is discussed by (Nicolle et al., 2004; Baker & Moallem, 2007). They assumed a symmetric Gaussian PSF for the weighting function, where $\sigma_x = \sigma_y$. Nicolle et al. (2004) derived the expected centroid estimation error for photon noise, σ_{ph}, and detector noise, σ_{det}, respectively. Therefore, the combined error can be written as

$$\mathrm{var}(\hat{x}_{WCoG}) = \mathrm{var}(\hat{y}_{WCoG}) = \sigma_{ph}^2 + \sigma_{det}^2 \tag{3.11}$$

with

$$\sigma_{ph}^2 = \frac{\pi^2}{2[\ln(2)]^2 \mathcal{N}_{ph}} \left(\frac{N_T}{N_D}\right)^2 \left(\frac{N_T^2 + N_w^2}{2N_T^2 + N_w^2}\right)^2, \tag{3.12}$$

$$\sigma_{det}^2 = \frac{\pi^3}{32[\ln(2)]^2} \left(\frac{\sigma_N}{\mathcal{N}_{ph}}\right)^2 \left(\frac{N_T^2 + N_w^2}{N_D)^2}\right)^2 \tag{3.13}$$

where $\mathcal{N}_{ph}$ is the total signal inside the RoI after thresholding, N_T is the FWHM of the instruments point spread function in pixels, N_w is the FWHM of the weighting function in pixels, N_D is the FWHM of the diffraction limited point spread function and σ_N represents the standard deviation of the Gaussian noise per pixel. In all simulations, I have set $N_D = 2$ for Nyquist sampling as well as $N_T = N_w$, since best results were obtained by using the PSF as weighting function. Equations (3.11) – (3.13) are valid for square-sized detectors only. The estimated errors for x and y dimension are equal, due to the assumed symmetry of the PSF and weight-

ing function. Note, that these estimators do not consider the initial centroid estimation that is required for the computation of WCoG and IWCoG. As a consequence, the estimation is only valid if the centre of the weighting function overlaps with the true centroid position, which is the case after a few iterations of IWCoG. If this error estimation is applied to WCoG, it should be dealt with great caution, as it only provides lower error boundaries which do not have to resemble the true error range in most cases (see Figure 3.7).

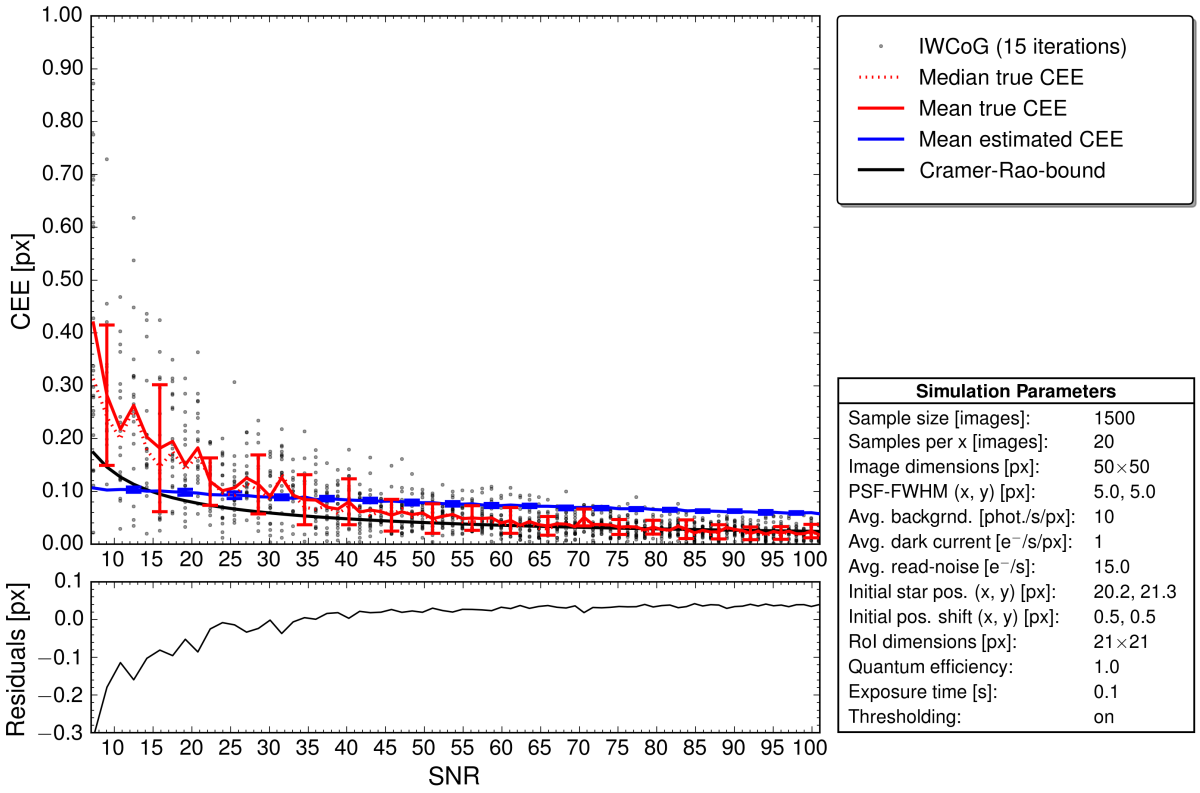

Figure 3.11.: The dependency of the IWCoG algorithm on the signal-to-noise ratio. A Monte-Carlo simulation carried out by *StarSim* produced a sample of 1 500 images (see Figure 3.2). 15 iterations of IWCoG were performed for each simulation. The transparent black dots indicate the computed true CEE per image. The blue line indicates the estimated mean CEE based on error estimation described in Section 3.2.3.2. The median values represent the most likely CEE. Error bars are symmetric as they represent the standard deviation of the mean CEE. The residuals depict the difference between mean estimated CEE and mean true CEE.

3.2.3.3. Performance

Monte-Carlo simulations were carried out in order to produce samples containing a large number of images. The performance of IWCoG was tested for different SNRs. Figure 3.11 reveals that performing multiple iterations of WCoG results in a much lower dependency on the SNR. Furthermore, it depicts that the impact of the initial centroid estimation, which significantly affected the WCoG algorithm, is removed entirely. Additionally, it can be seen that the IWCoG provides high-quality results with a small absolute CEE even for very low SNRs. An increase in the SNR only by a factor of five scaled down the CEE and its spread by nearly a factor of six. In general, the distribution of the true CEE is much smaller compared to other CoG algorithms. Furthermore, the IWCoG reached the optimal accuracy, limited by the Cramer-Rao bound, approximately at an SNR of 60. Here, the median true CEE is shown, as it differs from the mean true CEE for low SNRs. In this case the CEE follows a Poisson distribution rather than a normal distribution and the plotted average does not resemble the peak of the distribution any more. However, the most likely CEE is represented by the median. Error bars are symmetric as they illustrate the standard deviation of the mean true CEE. Error estimation was discussed in Section 3.2.3.2 and is represented as blue line. The residuals display a non-compliance of estimated error and true error for low SNRs. An explanation for the deviation of the error estimation which is based on Nicolle et al. (2004) requires further analysis. The error is also slightly overestimated at higher SNRs. However, as it is visualised in the residuals the magnitude of this overestimation is negligible (< 0.05 px). Note that, even if the error estimation did not resemble the true error perfectly, it is still an estimation of high quality as the accuracy is below 0.3 px.

3.2.3.4. Summary

The performance of the IWCoG was tested and compared to other methods under various conditions (see Section 3.5). A two-dimensional Gaussian weighting function was used like in the simulations for WCoG. The performance of IWCoG for different SNRs is outstanding. Even for low SNRs the centroid estimation was close to the Cramer-Rao bound. Eventually, this limit was reached at a SNR of about 60. After a certain number of iterations the CEE does not decrease any further. The number of iterations required to reach this error saturation depends on the start position as well as on the target's brightness. The main disadvantage of WCoG was the strong dependency on the start position. This issue was solved by performing multiple iterations. Therefore, IWCoG is capable of providing very accurate centroid estimations even for poor initial conditions. However, convergence against the true centroid only occurs as long as the weighting function (Gaussian spot) is overlapping with the target spot (see Figure 3.22). The target spot may not necessarily be centred in the RoI. Thresholding the background may lead to

faster convergence, but it is not a mandatory task to obtain low CEEs (see Figure 3.23). Identically to WCoG, centroid estimation is not affected by cosmic ray hits that occur outside the target spot, as long as they are not overlapping with the weighting function (see Section 3.5.4). If such glitches happen to be inside the target spot, the centroid estimation is distorted. IWCoG is also applicable in crowded field observations, as long as the star position is close to the target. In general, the processing time of IWCoG is simply the number of iterations multiplied by the average time required for the computation of WCoG. Thus, processing times may be too large depending on the hardware characteristics of the spacecraft. A possible approach for lower computation times is pre-computing a set of weighting functions. Calibration with dark frames is not necessarily required, if the occurrence of hot-pixels is low. The same applies for flat field calibration as long as variations in the pixel sensitivity are low. However, both procedures are suggested if high-precision centroiding is performed. In comparison to all other centroiding algorithms, IWCoG is among the most powerful methods in terms of accuracy and stability.

3.3. Correlation-Based Centroiding

The cross-correlation of a CCD image and a reference spot, that is often referred to as 'template' or 'pattern', may be used for centroid estimations as well (Vyas et al., 2010; Poyneer et al., 2003; Uhm et al., 2008; Thomas, 2004). This method is fairly well known for the purpose of adaptive optics with Shack Hartmann wavefront sensors. A possible application for space telescopes is discussed in this section. The correlation-based centroiding method is currently the only pattern matching technique that is used for centroiding tasks. The pattern should resemble the image spot as close as possible, similar to the weighting function applied in WCoG/IWCoG (see Section 3.2.2 and 3.2.3). Therefore, it is essential to obtain an accurate model of the PSF. In most cases, a two-dimensional Gaussian model (see Section 2.3) is applied. Once the cross-correlation between the image and the reference has been computed, the position of the correlation peak reveals the centroid location. The accuracy of this estimation is limited by the image resolution. One possible way to obtain intra-pixel centroid estimations is interpolation subsequent to the cross-correlation. This represents the state of the art implementation for centroiding via cross-correlation and is discussed in Section 3.3.3. Alternatively, I present a new way to obtain intra-pixel centroid estimations in Section 3.3.4 by upsampling the image resolution. If the reader is not familiar the with the concept of cross-correlation, I suggest to read the section below, as it describes the underlying principles briefly.

3.3.1. Correlation and Convolution

In general cross-correlation can be applied to describe the correlation between two signals. In the one-dimensional case it is often used do determine positions in spectral analysis, such as locating narrow spectral lines embedded in a large spectrum. This principle can easily be extended to solve similar two-dimensional problems like finding star positions in an image. The cross-correlation of two discrete two-dimensional functions, f and I, is defined as follows.

$$C(x,y) = (f \circ I)(x,y) = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} f(m,n)\, I^*(x+m, x+n) \tag{3.14}$$

$$\text{for } -(K-1) \leq x \leq M-1 \text{ and } -(L-1) \leq y \leq N-1$$

The above equation evaluates the cross-correlation at given x and y position. The dimension of f is $M \times N$ and the dimension of I is $K \times L$. Therefore the resulting matrix, C, is of size $(M+K-1) \times (N+L-1)$. For the purpose of centroiding, the computation of the complex conjugate, I^*, in Equation (3.14) is not relevant, as $I = I^*$ because I is an image that only holds real values. Function f is often referred to as filter, kernel or mask and holds information

about the expected stellar shape. C is the resulting correlation matrix, where the coefficients with highest values indicate the points of maximum correlation.
The processing of a convolution is quite similar. In fact, cross-correlation provides the same result as convolution if the filter is flipped in both dimensions before the correlation is carried out. The discrete two-dimensional convolution of f and I is defined as follows.

$$C^{\star}(x,y) = (f * I)(x,y) = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} f(m,n)\, I^{*}(x-m, y-n) \tag{3.15}$$
$$\text{for } -(K-1) \leq x \leq M-1 \text{ and } -(L-1) \leq y \leq N-1$$

$C^{\star}$ denotes the result of the convolution (not the complex conjugate). *If the filter is symmetric, convolution and correlation provide identical results*. Both are shift invariant and linear operations. However, a big difference is the fact that the convolution can be calculated with less computation time by applying the convolution theorem. This means, the convolution may be computed by point-wise multiplication of image and filter in the Fourier domain, which can be expressed as follows.

$$C^{\star} = \mathcal{F}\big\{\mathcal{F}\{f\} \cdot \mathcal{F}\{I\}\big\}^{-1} \tag{3.16}$$

Centroid estimations may be performed by Equation (3.16), by using a model of the PSF as symmetric filter. For the computation of Equation (3.16) a Fast-Fourier-Transform algorithm (FFT) may be applied. Although the overhead produced by such algorithms leads to additional computation time, the calculation via FFT is much faster compared to direct mode in most cases. Direct computation may only be faster for the case of very small image and filter sizes.

3.3.2. Application to Centroiding

A few things must be considered if cross-correlation is applied to obtain centroid estimations. Keeping the window size small is crucial, as bigger RoIs lead to much higher computation times (see Figure 3.28). Although this method was originally designed for wave-front sensing in adaptive optics, it's also applicable for centroiding tasks in space if sufficient CPU capacity and clock speed are available. One big advantage of the correlation method is that no initial centroid estimation is necessary to run the algorithm and find star positions. However, for the case of multiple stars inside the RoI it is necessary to estimate the centroid position in order to distinguish the target star from others. There are several options available for boundary conditions in two-dimensional image correlation. Two types are applicable for the purpose of centroiding. One possibility is to reflect the image array at the boundary and another option is to set boundary values constantly to zero. I suggest to use reflecting boundaries for handling small PSF sizes and zero-value boundaries otherwise.

The performance of correlation-based centroiding is limited by the image resolution. In fact, without any modification the obtained centroid information is limited to pixel scale, meaning that intra-pixel locations cannot be determined. Modifications that solve this particular problem are presented in the following sections.

3.3.3. Centroid Estimation by Interpolation

Poyneer et al. (2003) presented a way to obtain intra-pixel centroid estimations by interpolating between three points across the maximum of the correlation function. The correlation between the sub-aperture image (or RoI), $S(i, j)$, and a reference spot, $r(i, j)$, of Gaussian shape is computed. The reference spot should be smaller than the RoI in order to get best results. Here $r(i, j)$ is quadratic with size $N \times N$. The centroid estimation below is derived for the x-dimension and can be applied for the y-dimension analogous by flipping indices. The x-estimator is represented by

$$E(x_c^*) = \frac{0.5(m_1 - m_{-1})}{m_1 + m_{-1} - 2m_0} \tag{3.17}$$

where $\mathrm{m_k}$ are the means of the cross-correlation

$$m_k = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} r(i-k, j)\lambda(i + p_x, j + p_y) \tag{3.18}$$

with (p_x, p_y) being the peak location of the correlation function. The function $\lambda(i + p_x, j + p_y)$ represents the image, $S(i, j)$, reduced by the Gaussian noise component. Photon noise is still included in S in each pixel. The components m_1, m_0 and m_{-1} determine the points of interpolation, where m_0 is for the peak and m_1, m_{-1} for the adjacent areas. The characteristics of this method were analysed in Section 3.5 and are summarised in Section 3.3.3.4.

3.3.3.1. Implementation

The correlation-interpolation method may be implemented by executing the following steps,

1. Obtain a template of the point spread function and use it as reference spot r. The resolution of the spot must match the resolution of stars inside the RoI.
2. Compute the cross-correlation between RoI (S) and reference spot (r) via convolution in Fourier domain. Dimensions of S and r may differ.
3. Evaluate the peak location of the correlation function on pixel scale in order to determine m_1, m_0 and m_{-1} according to Equation (3.18).
4. Obtain the centroid estimator by interpolating with Equation (3.17).

Implementations of correlation-interpolation are provided in the Appendix A.3.4.

3.3.3.2. Error Estimation

The performance of the correlation method presented in 3.3.3 was analysed by Poyneer et al. (2003). For the case of zero shift, which means that m_0 is overlapping with the peak position, the variance due to photon noise can be expressed as

$$\mathrm{var}(x_c)_p = \frac{\sigma_1^2 - \sigma_{-1,1}^2}{8f(m_0 - m_1)^2} \tag{3.19}$$

where f represents the quantum efficiency and (σ_k^2,$\sigma_{k,l}^2$) are the variances and covariances of the cross-correlation expressed by

$$\sigma_k^2 = \sum_{i=0}^{N-1}\sum_{j=0}^{N-1} r^2(i-k,j)\sigma_S^2(i,j) \tag{3.20}$$

$$\sigma_{k,l}^2 = \sum_{i=0}^{N-1}\sum_{j=0}^{N-1} r(i-k,j)r(i-l,j)\sigma_S^2(i,j) \tag{3.21}$$

with $\sigma_S^2 = \lambda(i,j) + \sigma_N^2$ including the variance of the Gaussian noise σ_N^2.
The standard deviation associated to Equation (3.19) scales inversely with the SNR. The variance caused by the read-noise component can be expressed as

$$\mathrm{var}(x_c)_r = \sigma_N^2 \frac{\sum_{i=0}^{N-1}\sum_{j=0}^{N-1} r(i-1,j)[r(i-1,j) - r(i+1,j)]}{8(m_0 - m_1)^2} \tag{3.22}$$

Combining Equations (3.19) and (3.22) leads to the total error in the centroid estimation for the x-component. Yet again, the y-component may be computed analogous by flipping indices.

3.3.3.3. Performance

Monte-Carlo simulations were carried out in order to produce samples containing a large number of images. The performance of the correlation-interpolation method was tested for different SNRs and results are shown in Figure 3.12. The mean true CEE approached the Cramer-Rao bound at higher SNRs and results only slightly differ from this limit as noise was increased. Raising the SNR by a factor of ten resulted in a decrease of the CEE nearly by a factor of five. Additionally, the standard deviation of the mean true CEE is approximately lowered by a factor six. The residuals reveal that the error estimation described in Section 3.3.3.2 resembles the true error almost perfectly. The only considerable deviation is the slight underestimation of the error at SNRs below ten. Although the plot reveals a slight dependency on the SNR, the correlation-interpolation method is considered as a noise-insensitive method in comparison to other centroiding techniques.

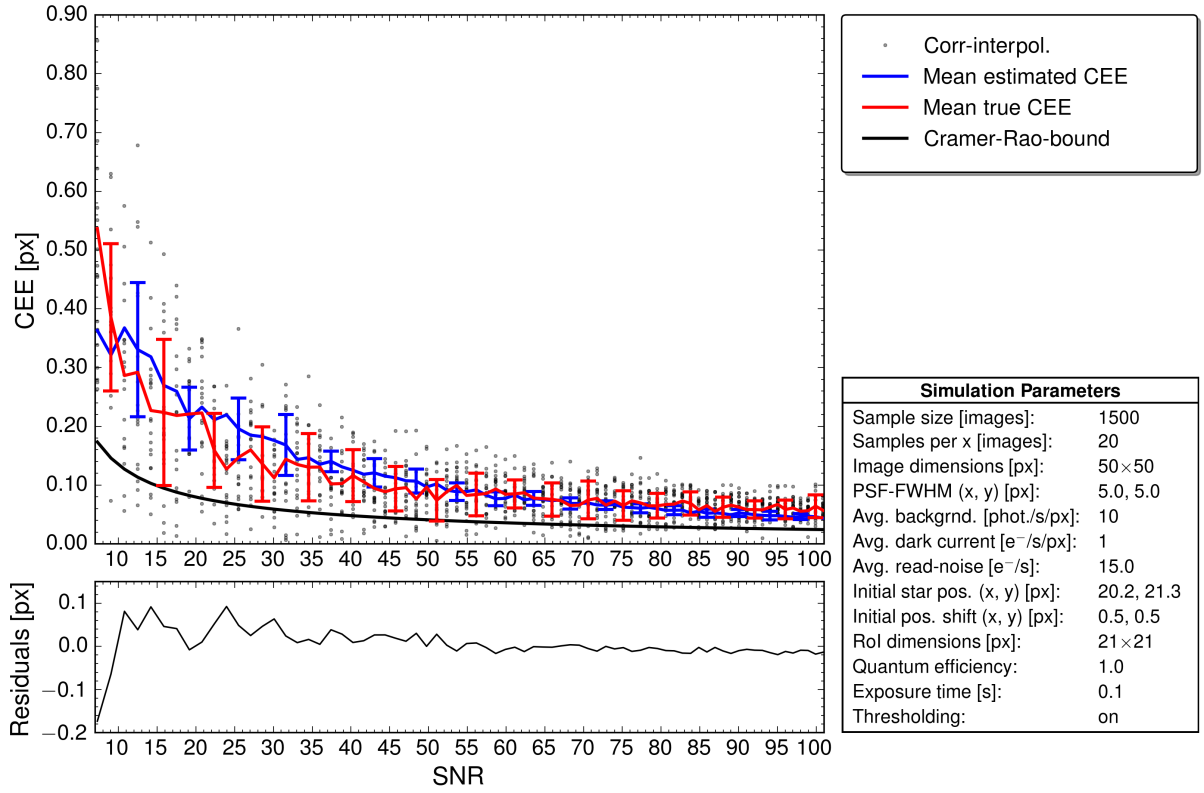

Figure 3.12.: The dependency of the correlation-interpolation method on the signal-to-noise ratio. A Monte-Carlo simulation carried out by *StarSim* produced a sample of 1 500 images (see Figure 3.2). The transparent black dots indicate the computed true CEE per image. The blue line indicates the estimated mean CEE based on the error estimation described in Section 3.3.3.2. Error bars represent the standard deviation. The residuals depict the difference between mean estimated CEE and mean true CEE.

3.3.3.4. Summary

The performance of the correlation-interpolation method was tested and compared to other methods under various conditions (see Section 3.5). In contradiction to the CoG algorithms, the performance of correlation methods is not affected by an oversized RoI. Furthermore, the computation does not require an initial centroid estimation. However, such estimation is required to define the RoI. In addition, it is necessary in the presence of a second star inside the RoI, because two correlation peaks exist. The correlation-interpolation method features a low dependency on the SNR. Although centroid estimations are very accurate for very low SNRs, the Cramer-Rao bound is not reached entirely below SNRs of 100. Thresholding the background is less important than for most CoG types, yet it is recommended (see Figure 3.23). In principle, the correlation-interpolation method is invariant to displacements of the RoI as well as initial conditions. In fact, the target spot must be embedded in the RoI entirely. Apart from that, the spot can be shifted anywhere in the RoI without any loss in performance (see Figure 3.22). Furthermore, centroid estimation is not affected by cosmic ray hits that occur outside the target spot (see Section 3.5.4). However, impacts within the spot lead to distorted centroid estimation. The

processing time of correlation-interpolation is much lower compared to the the later introduced correlation-upsampling method (see Figure 3.5.6). The computation time increases with larger PSF sizes, as bigger templates are required. In simulations where a symmetrical Gaussian PSF with a FWHM at five pixel was used, the processing time was about the same as for WCoG and LMA. Similar to the CoG, the correlation-interpolation method performs worse if the spot's peak is situated at the pixel's edge (see Figure 3.5.5). I recommend to perform image calibration with dark frames as well as flat fields. In particular sensitivity gradients should be reduced. In comparison to all other centroiding algorithms, correlation-interpolation is among the most powerful methods in terms of accuracy and stability and features reasonable processing times for space applications.

3.3.4. A New Method: Centroid Estimation by Upsampling

In this section, I present a new method of getting centroid estimators on intra-pixel scale via cross-correlation, as an alternative to the parabolic interpolation method presented in Section 3.3.3. It has been stated earlier that the peak of the cross-correlation between image and reference spot represents a low-accuracy centroid estimation. In terms of pattern-matching, the image is cross-correlated with a model of the point spread function in order to locate the points of maximum overlap. These points indicate integer pixel positions of cross-correlation peaks inside the image. At this point one can interpolate in order to acquire the intra-pixel positions of a peak (as done in Section 3.3.3), or perform upsampling of the image and the template prior to the correlation. In particular, upsampling means upscaling the image resolution by a certain factor, which I refer to as upsampling factor. Figure 3.13 shows three different cases of cross-correlation with a reference spot. The first case shows the original image and a low-resolution PSF-model as reference spot. For the second and third case the image was upsampled and convolved with a high-resolution PSF model leading to more precise centroid estimation on intra-pixel scale. The third one is a special case, as spline interpolation of order one has been performed additionally to upsampling the image. This kind of additional interpolation produced more stable results in some cases. However, this step does not improve the accuracy and it will not further be discussed in this thesis. It is crucial for the success of this method that the dimensions of the reference spot are of odd size. In addition, the spot must be perfectly centred, like it is illustrated in the upper right panel of Figure 3.13.

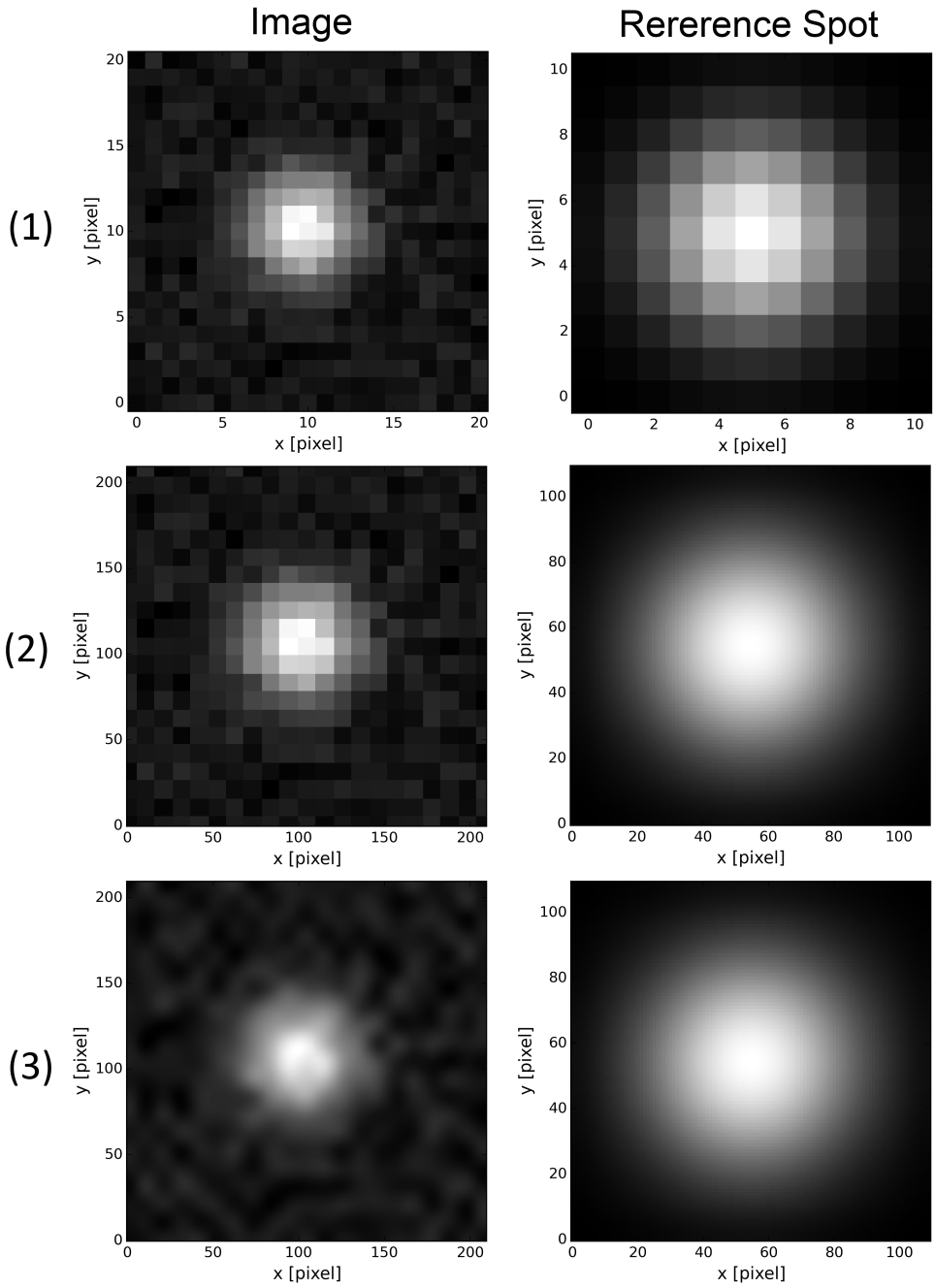

Figure 3.13.: Three different cases of cross-correlation between image and reference spot. Panel (1) depicts the original image and a circular Gaussian reference spot with a FWHM at five pixel. Panel (2) shows the image upsampled by a factor of ten as well as a PSF model with ten times increased resolution as reference. Note the change in the scale of the axes. Panel (3) is a special case of (2), where spline interpolation of order one has been performed additionally to upsampling. All images were generated with *StarSim*.

3.3.4.1. The Upsampling Factor

Figure 3.14 shows the CEE for different upsampling rates. It can be seen that the choice for the optimal rate that is necessary to achieve a certain accuracy, is affected by the prevailing SNR (see Equation (3.2)). In addition, Figure 3.14 reveals that the upsampling-correlation method provides results of high quality, even for very low SNRs. In the absence of any noise

sources, the mean true CEE converges against the Cramer-Rao bound for higher upsampling factors. In the case of a very low SNR (red circles), at a certain point the CEE does not improve any further for increased upsampling factors, as error saturation is reached. For the low-SNR simulation illustrated in Figure 3.14 an upsampling factor greater than four does not improve the CEE. Instead, the CEE levels off at 0.2 pixels. Unfortunately, the type of noise source plays an important role for this phenomenon. Therefore, the total SNR does not directly correlate with the required upsampling factor. In fact, an increased read-noise stronger influences the performance of the upsampling-correlation method compared to enhanced background noise, as the background noise is of uniform nature. In principle, both correlation-based centroiding methods are not affected by any noise situated outside the target spot. The optimal upsampling factor is mission specific and depends on the *desired CEE* as well as on prevailing *noise-levels*.

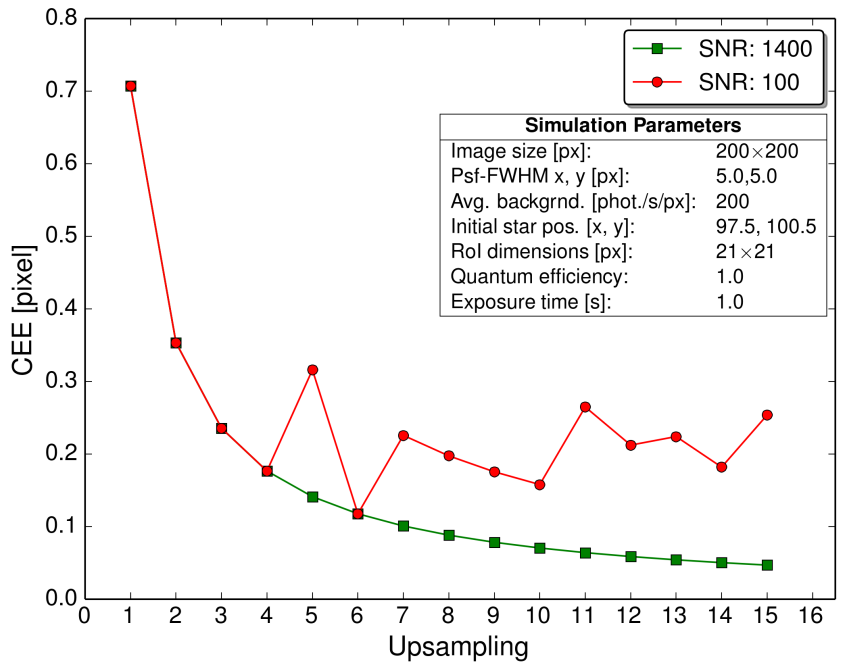

Figure 3.14.: The centroid estimation error for different upsampling factors. Red circles indicate a low SNR of 100 and green squares represent simulations with a SNR of 1 400.

Increased upsampling factors lead to much higher computation times. For instance, an upsampling factor of 10 for a 60×60 px RoI, results in a 600×600 px array. Despite the additional computational cost of the newly introduced upsampling process, the computation time for the FFT is at least increased by a factor of 150, due to the larger image size. This estimation is based on the computational complexity of practical FFT algorithms. Such algorithms perform about $O(N \log_2 N)$ floating-point operations, where N represents the number of pixels.
The most significant disadvantage of the upsampling-correlation method is depicted in Figure 3.15. It represents the escalation in computation time for increased upsampling rates, which are essential for accurate centroid estimations (see Figure 3.14). A comparison with computa-

tion times of other centroiding techniques is given in Figure 3.28. The substantial amount of computation time is the main reason why the correlation-upsampling method is currently not applicable for space applications. However, it might be applicable in the near future for space telescopes with increased CPU capabilities, as it is among the group of the most powerful and reliable centroiding techniques.

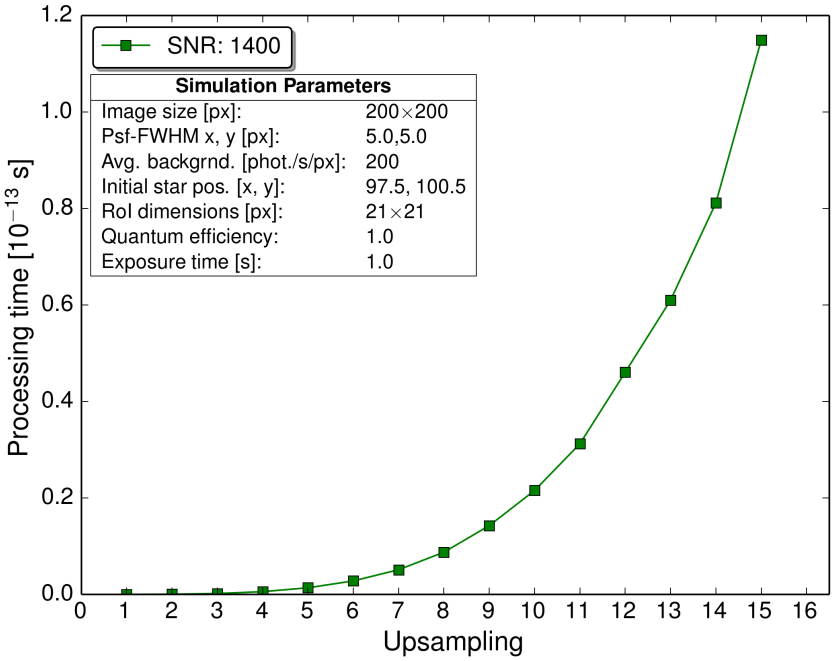

Figure 3.15.: The computation time of the upsampling-correlation method for different upsampling rates. Measurements were taken on a PC (CPU: Intel Core i7-3630QM, 4×2.4 GHz) for the Python implementation.

3.3.4.2. Implementation

The correlation-upsampling method may be implemented by executing the following steps,

1. Obtain a template of the point spread function and use it as reference spot r. The resolution of the template is defined by the upsampling factor U.
2. Upsample the RoI S by the factor U.
3. Compute the cross-correlation between upsampled RoI (S) and upsampled reference spot (r) via convolution in Fourier domain. Dimensions of S and r may differ.
4. Evaluate the peak location (x_{peak}, y_{peak}) of the correlation function in order to obtain centroid estimators.
5. Obtain centroid estimator x_c by converting the peak location to the coordinate system of the original image by applying $x_c = (x_{peak} - 0.5 \cdot U - 0.5)/U$. Compute the estimator x_c analogously.

Implementations of correlation-upsampling are provided in the Appendix A.3.4.

3.3.4.3. Error Estimation

In addition to the error estimation presented in Section (3.3.3.2), a lower boundary for the error estimation is provided by the upsampling factor U.

$$\mathrm{err}(x_c) = \mathrm{err}(y_c) \geq (2U)^{-1} \tag{3.23}$$

This limit is directly related to the image resolution. Without changing the resolution ($U = 1$), the correlation peak may be situated anywhere inside the pixel. In fact, if the peak is located at the border of the pixel (e.g. in x-dimension), the centroid estimator is shifted by 0.5 pixel. The error estimator in Equation (3.23) coincides with the results presented in Figure 3.14. It is clearly visible, that an upsampling factor of $U = 2$ leads to a two-dimensional CEE of about 0.355 pixel, which corresponds to a shift of 0.25 pixel in one dimension.

3.3.4.4. Performance

Monte-Carlo simulations were carried out in order to produce samples containing a large number of images. The performance of the correlation-upsampling method was tested for different SNRs. All simulations were performed with an upsampling factor of 10, which corresponds to a lower limit for the true CEE of 0.071 (see Section 3.3.4.3). Figure 3.16 reveals that this limit is reached at a SNR of 50. At this point the spread of the true CEE completely vanishes. It is a remarkable feature of this algorithm that there is absolutely no spread, once the optimal performance limit is reached. Furthermore, the distribution of the mean true CEE is quite low for small SNRs and the error is close to the Cramer-Rao bound (< 0.2 px). This method produces most constant results and therefore it is considered as the most noise-insensitive algorithm presented in this thesis. Similar to all other algorithms twenty images were generated per SNR value. As a consequence of the noise insensitivity and due to the discreteness of correlation-upsampling, the centroid estimations overlap. Therefore the appearance of the black data points in the graph is more compact compared to other algorithms. Overlapping results appear as dark data points, whereas single non-overlapping CEEs are transparent. The residuals show that using the lower-bound error estimation discussed in Section (3.23) resembles the true CEE completely once a certain SNR is reached.

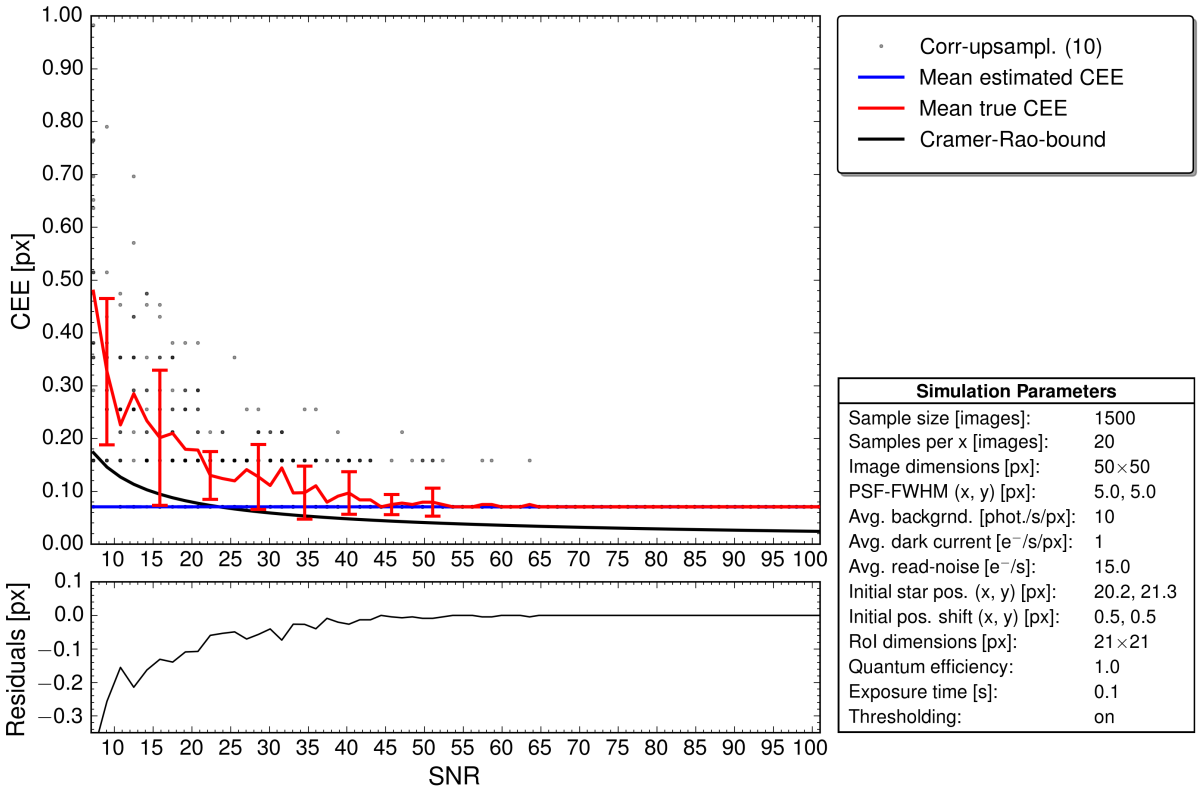

Figure 3.16.: The dependency of the correlation-upsampling method on the signal-to-noise ratio. A Monte-Carlo simulation carried out by *StarSim* produced a sample of 1 500 images (see Figure 3.2). The transparent black dots indicate the computed true CEE per image. The blue line indicates the estimated mean CEE based on error estimation described in Section 3.3.4.3. Error bars represent the standard deviation. The residuals depict the difference between mean estimated CEE and mean true CEE.

3.3.4.5. Summary

The performance of correlation-upsampling was tested and compared to other methods under various conditions (see Section 3.5). The accuracy of this algorithm is limited by the upsampling factor in Equation (3.23). Simulations with different SNR-values revealed that the performance is less stable for low SNRs, whereas for SNRs above 60 correlation-upsampling provided constant values for each simulated image. Results were close to the Cramer-Rao bound, but the CEE saturated at 0.071 px due to the upsampling factor of 10 (see Equation (3.23)). Similar to correlation-interpolation, no initial centroid estimation is required, except for selecting the RoI and for distinguishing the target spot from others in a crowded field. Yet, there is a slight difference to correlation-interpolation if the target spot is situated close to the RoI border (see Figure 3.22). In this case, the errors are slightly higher for correlation-upsampling and therefore it is not entirely invariant to RoI displacement (see Figure 3.22). Thresholding the background does not affect the performance at all and is therefore not recommended (see Figure 3.23). The method is also invariant to cosmic ray hits that occur outside the target spot (see Section 3.5.4). Impacts within the spot lead to distorted centroid estimation as it is the case for any other method. However, this is only true for glitches with high energies. In fact, correlation-upsampling is invariant to weak cosmic ray hits (with signals far below the brightest pixel of the target). The processing time of correlation-upsampling is extraordinarily high, thus it may not be applicable on current space hardware (see Figure 3.5.6). The computation time mainly depends on the upsampling factor and on the size of the target spot. Simulations showed that computations carried out with an upsampling factor of 10 for a RoI size of 20×20 pixels take about 300 times longer than the standard CoG. This represents the main disadvantage of correlation-upsampling. However, the processing time might be negligible in future, due to increased hardware capabilities. In contradiction to the previously introduced correlation method, the upsampling method is invariant to intra-pixel positions of the stellar peak (see Figure 3.5.5). Anyhow, the presence of a strong sensitivity gradient distorts centroid estimations like for any other method. In such cases, I recommend flat field calibration. Calibrations with dark frames can be neglected if the amount of hot-pixels is low. Correlation-upsampling belongs to the most powerful centroiding methods in terms of stability. However, high-accuracy centroid estimations are generally possible, but they require long processing times.

3.4. Direct Fitting Strategies

Another attempt to obtain centroid estimations is direct fitting of a pre-defined model (Tremsin et al., 2003; Delabie et al., 2014; Thompson et al., 2002). In this chapter, I describe ways to fit one-dimensional and two-dimensional Gaussian models. In most cases Gaussian models represent the PSF with sufficient accuracy. However, there are exceptional cases such as the CHEOPS mission, where custom PSF-models are required. This particular case is discussed in Section 4.2.3. It is important to mention that none of the presented algorithms are restricted to Gaussian models only. For direct fitting it is necessary to estimate the centroid position in advance, in order to provide initial parameter estimations. This input may be provided by star trackers or by combining centroiding techniques (hybrid-technique).

3.4.1. Gaussian Three-Point Fit (G3P)

A simple fitting technique is described by Tremsin et al. (2003) and summarised below. A one-dimensional Gaussian model can be described as follows:

$$I(x) = \frac{A}{\sigma\sqrt{2\pi}} \exp\left[-\left(\frac{(x-x_0)^2}{2\sigma^2}\right)\right] \tag{3.24}$$

where A is the amplitude, x_0 is the central peak position and σ is the standard deviation. By applying $\ln$ on both sides of Equation (3.24), it can be rewritten as

$$\ln I(x) = -\left[x_0^2 - 2x_0x + x^2 + 2\sigma^2 \ln\frac{\sigma\sqrt{2\pi}}{A}\right] / \, 2\sigma^2 \tag{3.25}$$

Equation (3.25) represents a quadratic function where x_0 denotes the maximum. This is the only parameter holding centroid information. By fitting a parabola into three equidistant points $I_{i-1}(x_{i-1})$, $I_i(x_i)$ and $I_{i+1}(x_{i+1})$, the maximum x_0 can be expressed as

$$x_0 = P\frac{\ln I_{i+1} - \ln I_{i-1}}{2[2\ln I_i - \ln I_{i-1} - \ln I_{i+1}]} + iP \tag{3.26}$$

where P denotes the fixed distance between the three points. For our application this distance represents the size of a pixel and therefore P is set to one.
In order to obtain the centroid, the fit has to be applied in y-dimension as well. Therefore, five points are required in total, where the central point I_i is the same for both dimensions. The selection of these points is the most crucial step for this technique, due to fact that the centroid cannot be obtained if it is not located within the selected points. Therefore, this method strongly relies on the initial centroid estimation. Instead of choosing the brightest pixel, as suggested by Tremsin et al. (2003), I recommend to use a noise and glitch insensitive strategy, such as

correlation-based centroiding for estimating the initial centroid. In fact, correlation can be used to determine a reliable centroid position with integer accuracy (without upsampling) and subsequently, any kind of fit could be used to increase the accuracy to intra-pixel scale.

3.4.1.1. Implementation

The centroiding method G3P includes two one-dimensional fits of the PSF's central peak. Each fit is applied to obtain the centroid estimation for one dimension (x and y). This technique may be implemented by executing the following steps,

1. Obtain an initial centroid estimation x_0, y_0 (e.g. from previously executed centroiding runs).
2. Use x_0 and y_0 to select the three pixels I_{i-1}, I_i and I_{i+1} for both dimensions. Since I_i represents the central pixel in both cases, a total amount of five pixels must be selected.
3. Ensure that none of the selected pixels holds zero values. Otherwise replace zero values with 10^{-8}.
4. Perform the centroid estimation by Equation (3.26) with $P = 1$.

Implementations of G3P are provided in the Appendix A.3.5 in languages C and Python.

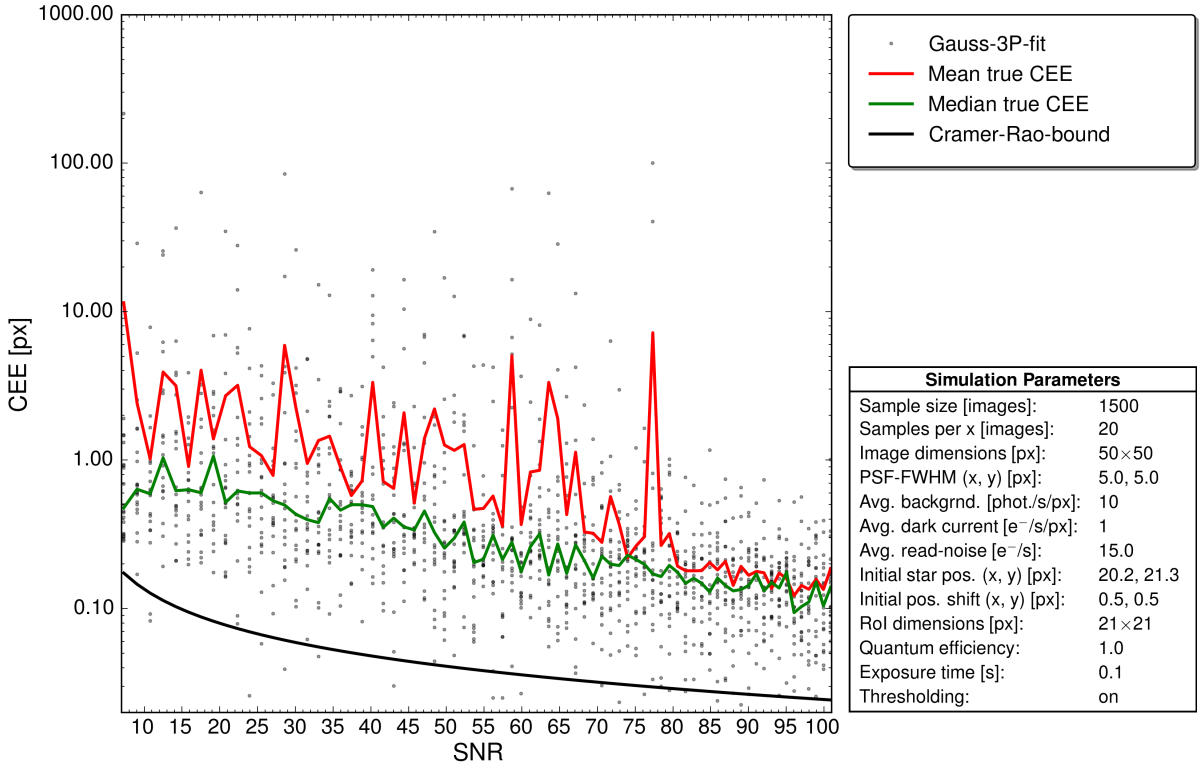

Figure 3.17.: The dependency of the Gaussian 3-point fitting method on the signal-to-noise ratio. A Monte-Carlo simulation carried out by *StarSim* produced a sample of 1 500 images (see Figure 3.2). The transparent black dots indicate the computed true CEE per image. The blue and green line indicate the mean and median true CEE respectively.

3.4.1.2. Performance

Monte-Carlo simulations were carried out in order to produce samples containing a large number of images. The performance of the three point fitting technique was tested for different SNRs. Figure 3.17 depicts the dependency of the fit described by Equation (3.26) on the SNR. The choice of the five points that are used for the fit depends on the initial centroid estimation which had been displaced constantly by 0.5 pixel in x and y in this simulation. Note that the y-axis is of logarithmic scale. The green line shows the median true CEE which is mostly below one pixel. However, in some cases the noise component caused extraordinarily high CEEs, as only three data points are used for the fit. Results beyond the size of the RoI (21×21 px) were not clipped in order to illustrate the possible extreme behaviour of this fit. For the investigated SNRs, the method showed a clear offset to the Cramer-Rao bound. The mentioned extreme behaviour did not occur at high SNRs (> 90). This is also verified by the fact that the median values conform to the average values for such high SNRs. Due to its bad performance, this method was not included in the comparison between the algorithms illustrated in Figure 3.20.

3.4.1.3. Summary

The performance of G3P was tested and compared to other methods under various conditions (see Section 3.5). In simulations with different SNRs, G3P performed worse than any other method. Large errors occurred randomly for low SNRs (<80), thus G3P is considered to be very unstable. For larger SNRs the performance increased, but the mean CEE did not converge against the Cramer-Rao bound. The initial centroid estimation is crucial for this method, as it determines the pixels that are included in the fit. G3P is invariant to a displacement of the RoI, which means that the target spot can be located anywhere inside the RoI as long as the initial centroid estimation is accurate (see Figure 3.22). Furthermore, it is invariant to the presence of additional objects inside the RoI. Thresholding the background is not necessarily required (see Figure 3.23). The method is invariant to cosmic-ray hits as long as the glitch does not occur in one of the five pixels required for the fit (see Section 3.5.4). Otherwise the provided centroid estimation must be discarded. The introduction of random pixel sensitivities lowered the performance stronger than for any other method (see Section 3.5.5). In general, dark current and pixel sensitivity variations should be reduced via calibration processes, otherwise G3P may not provide reasonable results. The shape of the PSF is essential for successful centroid estimations with G3P, as it works best for radial symmetric Gaussian PSFs. In particular, G3P should not be applied for PSFs without a central peak in the spot. The computation time was very low as only five data points were included in the fit in any case (see Figure 3.5.6). I do not recommend to use this method for centroiding as it requires initial centroid estimation with errors at least below two pixels. Initial centroid estimations should not be performed by selecting the brightest

pixels, as cosmic ray hits and hot-pixels may cause strong distortions. The variance of the mean CEE was significantly larger compared to other methods in all simulations, therefore G3P is considered to be highly unstable.

3.4.2. Two-Dimensional Non-Linear Least-Squares Minimisation

In order to take advantage of all pixels containing stellar signal, two-dimensional models may be applied to fit the image data. The following equation represents a two-dimensional Gaussian function

$$f(x,y) = A \exp\left[-\left(\frac{(x-x_0)^2}{2\sigma_x^2} + \frac{(y-y_0)^2}{2\sigma_y^2}\right)\right] \tag{3.27}$$

where the parameters x_0 and y_0 hold the centroid information. The technique described below allows to fit the other parameters as well. However, σ_x and σ_y are determined by the instrument's PSF, thus they are well-known. Therefore, they can be treated as constants. The amplitude, A, is proportional to the incident star signal, therefore it is also known prior to the observation in many cases. In general, least-squares minimisation is performed to minimise the function

$$S(\vec{a}) = \sum_{i=1}^{n} r_i(\vec{a})^2 \tag{3.28}$$

with the residuals $r_i(\vec{a}) = y_i - f(x_i, \vec{a})$, where x_i and y_i are measured quantities and $\vec{a}$ holds the parameters of the non-linear model, $f(x_i, \vec{a})$. This allows to optimise the parameter set $\vec{a}$. Equation (3.28) is not restricted to one-dimensional problems. For instance, r_i can be expressed for two-dimensional function as $r_i(\vec{a}) = z_i - f(x_i, y_i, \vec{a})$. There exist numerous methods for solving this particular problem. However, I used a well-established method which is described in the next section.

3.4.2.1. Levenberg-Marquardt Algorithm (LMA)

The Levenberg-Marquardt algorithm (LMA) is a common tool for least-squares minimisations of non-linear parameters. The original method of Levenberg (1944) was adapted over the years by various authors (Marquardt, 1963; Jacobs, 1977; Press et al., 2007). An optimised implementation in C and Fortran is offered by the Minpack project (Moré et al., 1984), which evolved to a standard library for fitting of non-linear functions over the years. I used Minpack's routines to solve the two-dimensional least-squares problem in order to estimate the centroid. Python and C source code is provided in the Appendix A.3.6. It should be mentioned that Python's standard fitting routines use Minpack behind the scenes as well.

There exist several methods to minimise Equation (3.28), such as the Gauss-Newton algorithm or the method of gradient descent. The LMA combines and utilises the advantages of these two methods. However, it still remains a plain downhill search. This represents the biggest disadvantage of the LMA, as it converges to nearby local minima rather than prioritising the global minimum. Therefore the choice of start parameters is very decisive in terms of finding the solution. On the other hand, the LMA is considered to be very robust, as it converges in most cases also if the initial parameter estimation strongly differs from their final values. The following description of the LMA is based on Nocedal & Wright (2006).
The LMA performs multiple iterations with the goal to optimise the parameter set $\vec{a}$. This is done by modifying the parameters in each iteration by applying $\vec{a} = \vec{a}+\vec{\delta}$. In order to determine the best modification the following linearisation is applied.

$$f(x_i, \vec{a} + \vec{\delta}) \approx f(x_i, \vec{a}) + J_i\vec{\delta} \tag{3.29}$$

The component J_i represents the gradient of the model function with respect to the parameter vector.

$$J_i = \frac{\partial f(x_i, \vec{a})}{\partial \vec{a}} \tag{3.30}$$

Applying this to Equation (3.28) and using vector notation leads to

$$S(\vec{a} + \vec{\delta}) \approx ||\vec{y} - \vec{f}(\vec{a}) - J\vec{\delta}||^2 = ||\vec{r}(\vec{a}) - J\vec{\delta}||^2 \tag{3.31}$$

where J represents the Jacobian matrix and $\vec{r}$ are the residuals. Computing $\frac{\partial S(\vec{a}+\vec{\delta})}{\partial \vec{a}} \equiv 0$ gives

$$(J^T J)\vec{\delta} = J^T\vec{r}(\vec{a}) \tag{3.32}$$

By solving this set of linear equations, δ can be determined. Levenberg introduced the damping parameter, λ, into this equation, which led to

$$(J^T J + \lambda I)\vec{\delta} = J^T\vec{r}(\vec{a}) \tag{3.33}$$

where I represents the unit matrix. Due to this modification, the method is alternatively called damped least-squares minimisation. Later on Marquardt adapted the equation by scaling the gradient's components which lead to faster convergence. Eventually, this led to the LMA's final form.

$$\left[J^T J + \lambda \operatorname{diag}(J^T J)\right] \vec{\delta} = J^T\vec{r}(\vec{a}) \tag{3.34}$$

3.4.2.2. Implementation

An implementation where the computation of matrix multiplication $J^T J$ is not necessary (singular value decomposition) is provided by Nocedal & Wright (2006). Increasing λ means the LMA resembles the gradient descent method rather than the Gauss-Newton method, whereas decreasing λ leads to the exact opposite. The principle of the algorithm can be summarised by the following iteration steps (see also Press et al. (2007)),

1. Compute the error $S(\vec{a})$.
2. Solve the linear Equations (3.34) for $\vec{\delta}$.
3. Compute the new error $S(\vec{a} + \vec{\delta})$.
4. If the error has *increased*: reject the step and *increase* λ.
5. If the error has *decreased*: accept the step by setting $\vec{a} = \vec{a} + \vec{\delta}$ and *decrease* λ.
6. Stop iterating if the exit condition is met, otherwise go to (1).

Iterations should be stopped if the reduction of the computed error or the change in $\vec{\delta}$ fall below pre-defined limits. The best choice of λ depends on the initial scale of the problem. However, Press et al. (2007) suggested to set the damping parameter initially to $\lambda_0 = 0.001$ and to apply a factor of 10 for increasing/decreasing.
Additionally, the LMA can be described as a trust-region algorithm. This means that the parameter choice is restricted to $||\vec{\delta_i}|| \leq \Delta_i$, where Δ_i is the radius of the trust-region. Trust-region algorithms are not further discussed as they are beyond the scope of this thesis, but an elemental description is provided by Nocedal & Wright (2006). Implementations of the LMA are provided in the Appendix A.3.6 for C and Python.

3.4.2.3. Application to Centroiding

The amplitude, A, of the model function (3.27) scales with the stellar signal and can be treated as constant for each observation. Therefore, the only remaining fitting parameter is the centroid position. In general, it is easy to distinguish between stars and other features on the detector, due to the PSF structure. Each star represents a local minimum, therefore it is likely that the LMA converges against stars that are closer to the start position.
The computation time of the LMA scales linearly with the number of iterations performed. It was found that the convergence speed of the LMA may be very slow, particularly if the best fit is located at a narrow maximum, such as the case for small point spread functions. This problem is illustrated by Figure 3.18 in which the first two hundred iterations are displayed. A circular Gaussian PSF with a spread of $\sigma \approx 2$ px was applied. The initial centroid estimation was displaced by only two pixels in x and y-dimensions. Although the initial CEE is very low, the

computation time is extraordinarily high, due to the slow convergence speed. However, there exist improvements to the LMA which address this particular problem (Transtrum & Sethna, 2012). For space application the computation time is a crucial topic. Therefore, it is interesting to know the maximum number of required iterations. I highly recommend to determine the number of iterations for each mission individually by testing the algorithm with an accurate PSF-model. Contrary to the IWCoG the convergence speed is not influenced by the stellar signal.

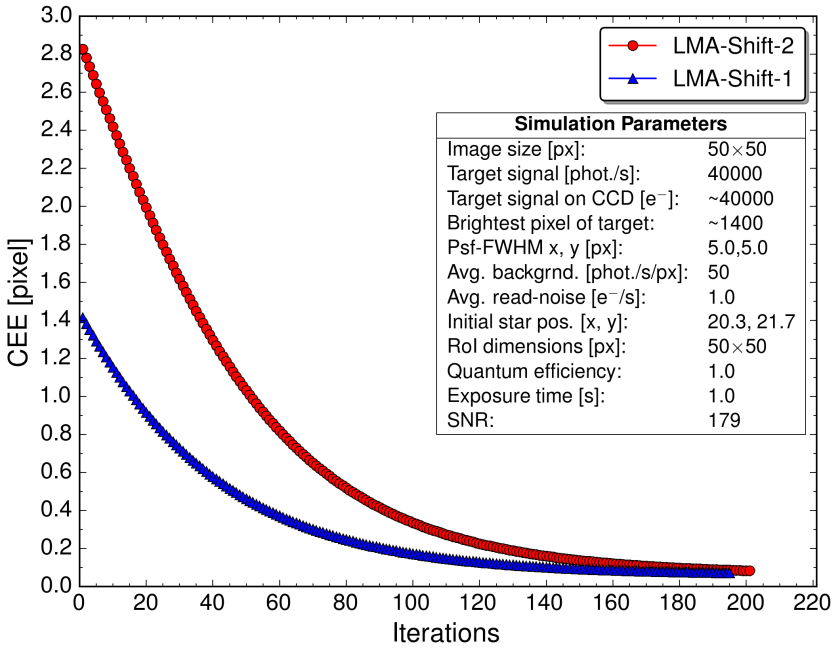

Figure 3.18.: Convergence speed of the Levenberg-Marquardt algorithm for a small PSF. Red circles and blue triangles indicate displaced centroid estimation by two and one pixel, respectively. Simulated images did not contain relevant noise sources.

3.4.2.4. Error Estimation

Error estimation of the fit parameters, $\vec{a}$, can be obtained via the covariance matrix $(\mathrm{J^T J})^{-1}$. This matrix has to be scaled with the reduced chi-square

$$\chi^2 = \sum_{i=1}^{N} \frac{r_i(\vec{a})^2}{N-n} \tag{3.35}$$

where N represents the number of data points, $\mathrm{r_i}$ are the residuals and n is the number of parameters (elements in $\vec{a}$). The standard error of the fitted parameters can be computed by taking the square root of the diagonal elements.

$$\vec{\sigma}_a = \sqrt{\mathrm{diag}[(J^T J)^{-1} \cdot \chi^2]} \tag{3.36}$$

This kind of error estimation simply represents the error of the fit and does not even require additional computation time. Compared to other error estimations it is more comfortable as it does not require any kind of noise estimation.

3.4.2.5. Performance

Monte-Carlo simulations were carried out in order to produce samples containing a large number of images. The performance of the LMA for different SNRs is almost equivalent to the performance of the IWCoG. This is further illustrated in Figure 3.20. An increase in the SNR by a factor of five lowered the mean true CEE and its spread by a factor of five to six. The LMA starts off close to the Cramer-Rao bound and reaches this limit approximately at a SNR of 60. The error estimation shown in Figure 3.19 is described in Section 3.4.2.4. Although this error estimation is simply based on the error of the fit without considering any noise estimations, it represents the true error with sufficient accuracy. However, the residuals reveal a constant overestimation of the CEE of about 0.05 px for SNRs above 20.

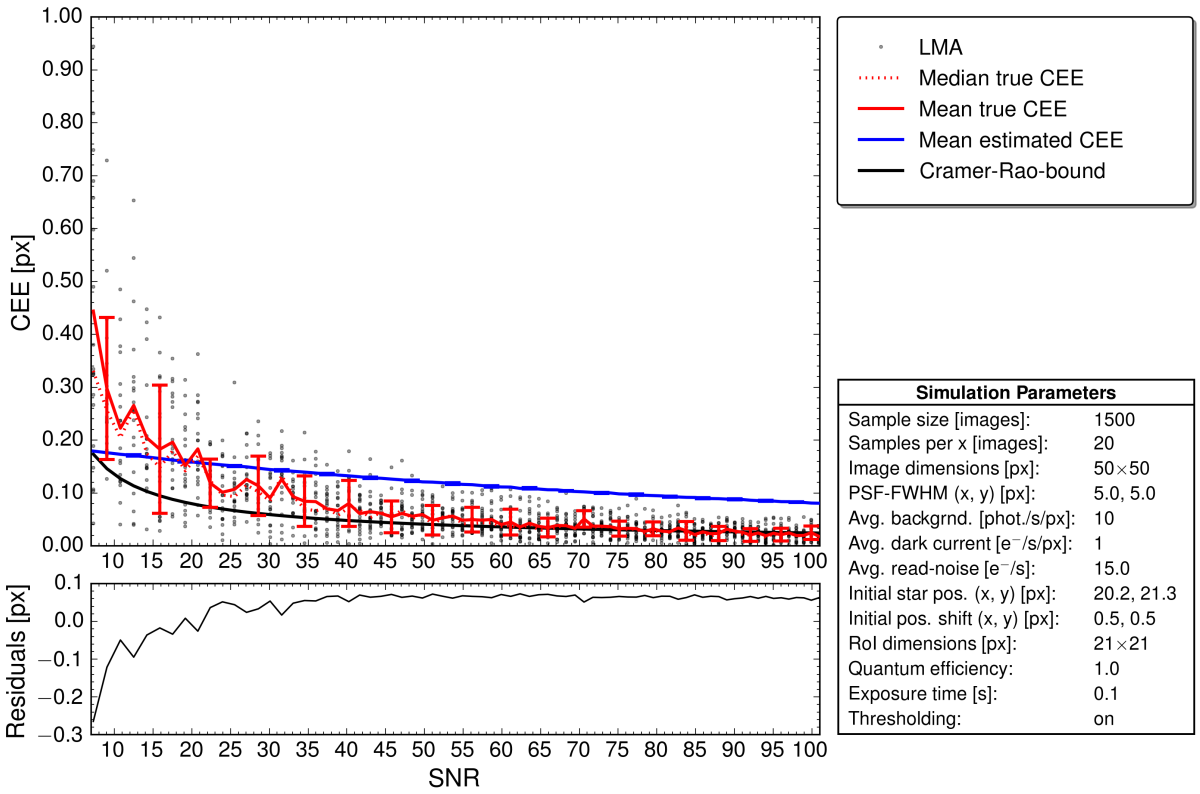

Figure 3.19.: The dependency of the LMA on the signal-to-noise ratio. A Monte-Carlo simulation carried out by *StarSim* produced a sample of 1 500 images (see Figure 3.2). The transparent black dots indicate the computed true CEE per image. The blue line indicates the estimated mean CEE based on error estimation described in Section 3.4.2.4. The median values represent the most likely CEE. Error bars are symmetric as they represent the standard deviation of the mean CEE. The residuals depict the difference between mean estimated CEE and mean true CEE.

3.4.2.6. Summary

The performance of the LMA was tested and compared to other methods under various conditions (see Section 3.5). The two-dimensional Gaussian model described by Equation (3.27) was applied to determine a centroid estimation. The centroiding performance of LMA is almost equal to the performance of IWCoG for different SNRs. Even for low SNRs the centroid estimation was close to the Cramer-Rao bound. Eventually, this limit was reached at a SNR of about 60. An initial centroid estimation is required for fitting the model into the RoI. At an offset of about $5\sigma_{\mathrm{PSF}}$, the LMA is not applicable, as no solution can be provided (see Figure 3.22). For any start positions closer to the true centroid the LMA is invariant and provides optimal results. However, the convergence speed increases for larger offsets as more iterations are required. The LMA is totally invariant to uniform background signal. Therefore thresholding is not required (see Figure 3.23). The presence of cosmic ray hits only affects the performance if they occur inside the stellar spot (see Section 3.5.4). For observations in a crowded field the initial conditions must include star positions close to the target. The computation time depends on the fit-model, the initial conditions and on the size of the RoI (see Section 3.5.6). Calibration with dark frames and flat fields are not necessarily required, as the influence of such image components on the astrometric error is relatively low. The LMA belongs to the most powerful methods in terms of accuracy and stability. In fact, LMA and IWCoG were slightly closer to the Cramer-Rao bound than the correlation-based centroiding techniques in most simulations.

3.5. Comparison of Algorithms

The impact of the SNR on the performance was presented for each algorithm separately in the previous sections. In this section the performance of the centroiding methods is compared in various circumstances such as cosmic ray hits, displacement of the RoI and others. Additionally, a comparison of the computation times was carried out, since it is a crucial subject for space applications.

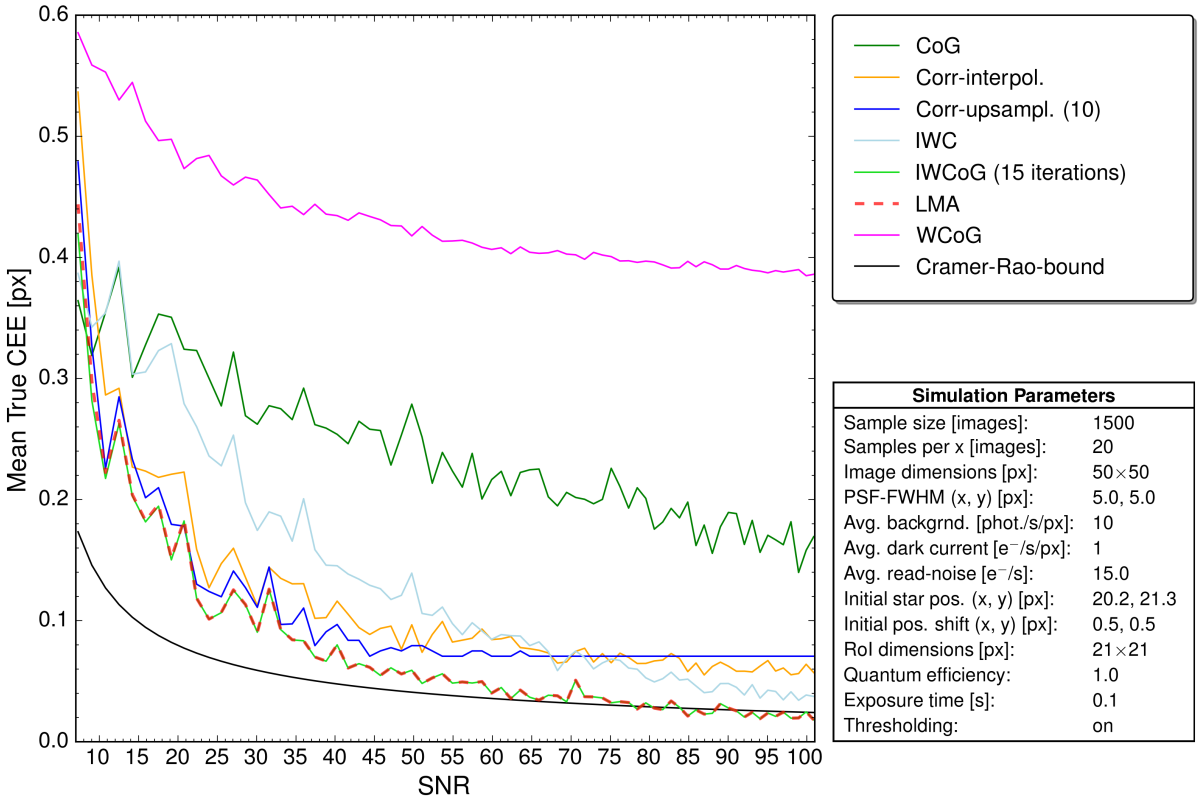

Figure 3.20.: A comparison of the dependency on the SNR of various centroiding methods. A Monte-Carlo simulation carried out by *StarSim* produced a sample of 1 500 images (see Figure 3.2). More detailed results are provided in the particular section of each method, where scatter of the mean true CEE was investigated as well.

3.5.1. Dependency on Signal-to-Noise Ratio

Figure 3.20 depicts the mean true CEE for different SNRs. The SNR dependency was illustrated for each centroiding method respectively in the previous sections. The one-dimensional fit described in Section 3.4.1 is not included, as its mean true CEEs are several magnitudes higher than others. The bad performance of the WCoG algorithm is related to the initial CEE of 0.71 pixel (see Section 3.2.2.2). The only algorithms that reached the optimal limit (Cramer-Rao bound) within the investigated SNR range are LMA and IWCoG. On first sight, it looks like their results perfectly overlap. Such characteristic overlap only occurs if the same function is

used as fit-model in the LMA and for constructing the weighting function in IWCoG. However, there is small difference for SNRs below 10, where IWCoG performed slightly better. The comparison of the two correlation-based centroiding methods showed, that the upsampling method provided more accurate results than the interpolation method, until the error saturation occurred due to the chosen upsampling factor. Although IWC is a rather simple method that does not require multiple iterations or initial centroid estimation, the CEE of IWC were surprisingly low for high SNRs.

3.5.2. Dependency on Region of Interest and Start Position

The behaviour of the centroiding methods was investigated for different start positions (see Figure 3.22). The initial position estimation as well as the centre of the RoI was moved away from the stellar centre as illustrated in Figure 3.21. Therefore, Figure 3.22 depicts the dependency on two different components. Firstly, the *initial centroid estimation* that is required for some algorithms may affect their performance if the distance to the true centroid is large. In fact, such initial position estimation is necessary for all methods, as it is used to define the RoI. Therefore, the star may be *displaced* from the centre inside the RoI, which affects the performance of some methods. The simulated image included a target star with a fixed SNR of 100. Simulations introduced in the previous sections, revealed that the noise component at such a SNR is low enough to provide CEEs with very low scatter (e.g. see Figure 3.11). Therefore, no Monte-Carlo simulation was performed in this case. The simulations were performed with proper thresholds. Neglecting this process would lead to even steeper curves in Figure 3.22. Results are now discussed for each method respectively.

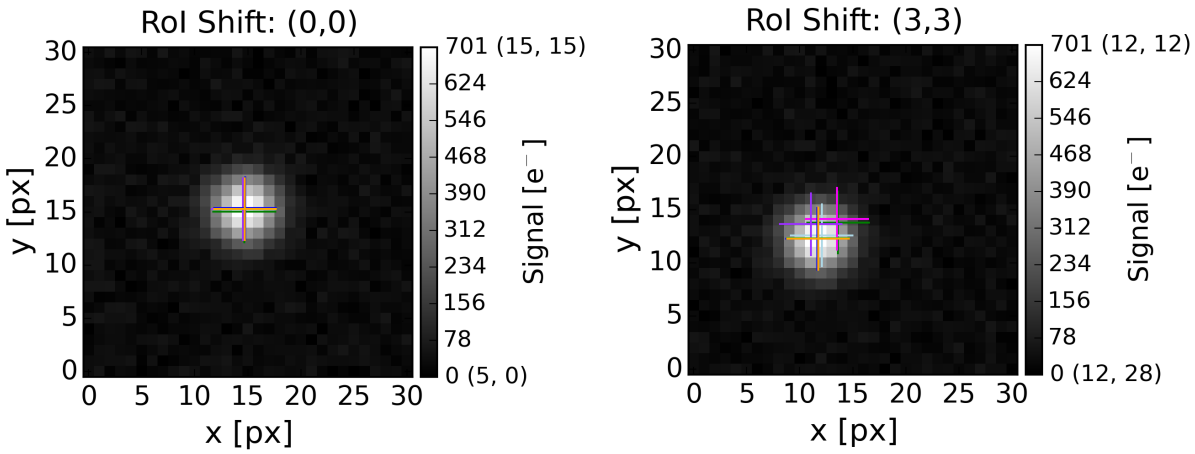

Figure 3.21.: Two simulated images of the same star with a SNR of 100. Crosses indicate results of the centroiding algorithms and the color codes are the same as in Figure 3.22. *Left*: The RoI is centred at the star. *Right*: The RoI was displaced by 3 px in x and y, which corresponds to a total initial CEE of 4.24 px. Images were generated with *StarSim*.

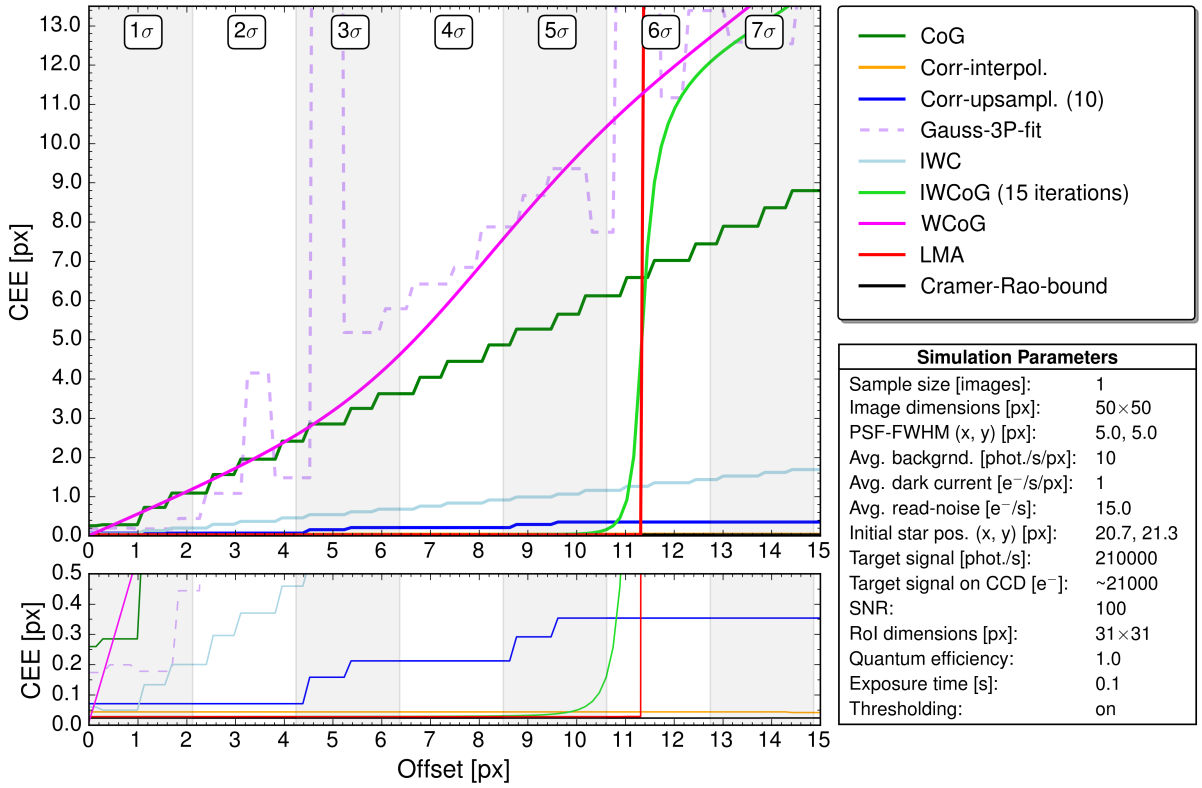

Figure 3.22.: The dependency of various centroiding algorithms on the initial position estimation (offset). This offset is a combination of the offset in x and y and represents the absolute distance to the true centroid. The images shown in Figure 3.21 illustrate the applied procedure. The lower panel simply represents a zoom on algorithms that perform well on position shifts. Shaded areas display the ranges of the standard deviation of the circular Gaussian spot.

CoG shows a stair-shaped profile. This step-wise increase in the CEE is related to the process of shifting the RoI away form the spot centre (see right panel of Figure 3.21). The shift was carried out on integer scale, which caused the characteristic stair-profile. Note that CoG does not require any initial centroid estimation for the computation, but the placement of the RoI strongly influences the computed centroid location. Best results are obtained if the star is centred inside the RoI.

WCoG depends even stronger on the initial offset. In contradiction to the standard CoG, the WCoG did not result in the characteristic stair profile. This is because WCoG applies a start position for evaluating the weighting function and the offset of the star position was gradually increased in steps of 0.1 px in x and y. This effect completely dominated the displacement of the RoI, which occurred on integer scale. In fact, this means that a displacement of the RoI-centre has no impact on the CEE, as long as the start position is accurate. Results were similar to the CoG as long as the weighting function overlapped with the stellar spot. WCoG produced worse results if the weighting function was placed at distances farther than 2σ away from spot centre.

IWC performed much better than CoG and WCoG with CEEs below two pixels during the entire simulation. However, a computation without thresholds would increase the slope of the CEE significantly. IWC also features a stair shaped profile, caused by the integer shift of the RoI. However, the zoomed plot on the lower panel of Figure 3.22 shows that the CEE is increased only by 0.1 px per step, which is much lower compared to the standard CoG.

IWCoG was performed with fifteen iterations and showed entirely no dependency on the start position up to a certain point. This characteristic point was about 5.4 σ away from the spot centre. This can be explained by looking at the weighting function. Once the weighting function is placed at a distance where it does not overlap with the stellar spot, the IWCoG stops converging against the true centroid even when the number of iterations is increased. This process is equal to the multiplication of two equally sized one-dimensional Gaussian profiles shifted to each other. If a read noise component is dominating the flanks of these Gaussians, the convergence limit is shifted. Performing less iterations would rise the CEE's slope for offsets below the convergence limit. This feature has been discussed in Section 3.2.3 (see Figure 3.9).

Gauss-3P-fit is a one dimensional fit using three points in x and y respectively. Therefore the CEE is lower than the offset only if the start position is located less than three pixels away from the spot centre. For higher values the magnitude of the CEE is mostly about the size of the offset. However, some offsets produced extreme values (e.g. at an offset of five pixels), which is one reason why this method is considered as highly unstable. Compared to the other algorithms this method again exhibits the worst performance like in the analysis for different SNRs.

LMA performed similar to IWCoG and featured the same convergence limit. Once the fit-model did no overlap with the stellar spot, the LMA did not provide any centroid information. However, up to this point the CEE was constantly low, even for high offsets.

Correlation-upsampling constantly provided results below 0.4 pixel. It featured an interesting stair profile illustrated in the lower panel of Figure 3.22. In contradiction to the other methods the stair profile is not related to the integer shift of the RoI. In fact, the star was moved towards the left edge of the RoI as offsets were increased (see Figure 3.21). This affected the results of the convolution, as some parts of the reference template were located outside the RoI. Boundary conditions were applied so that values beyond the RoI were constantly set to zero. Thus, results were less accurate at the border of the RoI. Alternatively, reflecting boundaries may be applied to correct this effect.
Correlation-interpolation showed no dependence on the start position. The stellar spot may be estimated anywhere inside the RoI, without affecting the high-precision performance of this method.

The two correlation methods, together with LMA and IWCoG performed best for imprecise initial centroid estimations. Therefore, these methods should be considered if centroiding is performed on spacecraft that exhibit a strong jitter.

3.5.3. Dependency on Thresholds

Figure 3.23 depicts the dependency of various centroiding methods on background thresholding. The mean true CEE was computed out of twenty images for each threshold level. The scatter of the true CEE was not plotted to avoid overloading the figure. However, the standard deviation of the computed true CEE was below 0.5 pixel for thresholds below 1000 for most methods. The only exceptions are the correlation-upsampling method, which did not feature any scatter and the Gauss-3P-fit, where the spread size was doubled. The initial centroid estimation was constantly set to one pixel in both dimensions during the entire simulation. All images were simulated with a SNR of 100 and they included a background signal of about 200 $e^{-}px^{-1}$. The threshold was gradually incremented to the highest pixel values (of about 1070) in steps of 10 e^{-}. Figure 3.23 reveals that the optimal threshold ranged from 300 to 500 e^{-}, as the CEE was equal to the Cramer-Rao bound for most methods in this threshold range. A minimum threshold value of 300 e^{-} was sufficient to remove the background, as well as bias and read-noise in regions outside the spot. For thresholds above 500 e^{-}, the outer parts of the stellar spots started to fade away, thus the CEE increased (over-thresholding). Results are now discussed for each method respectively.

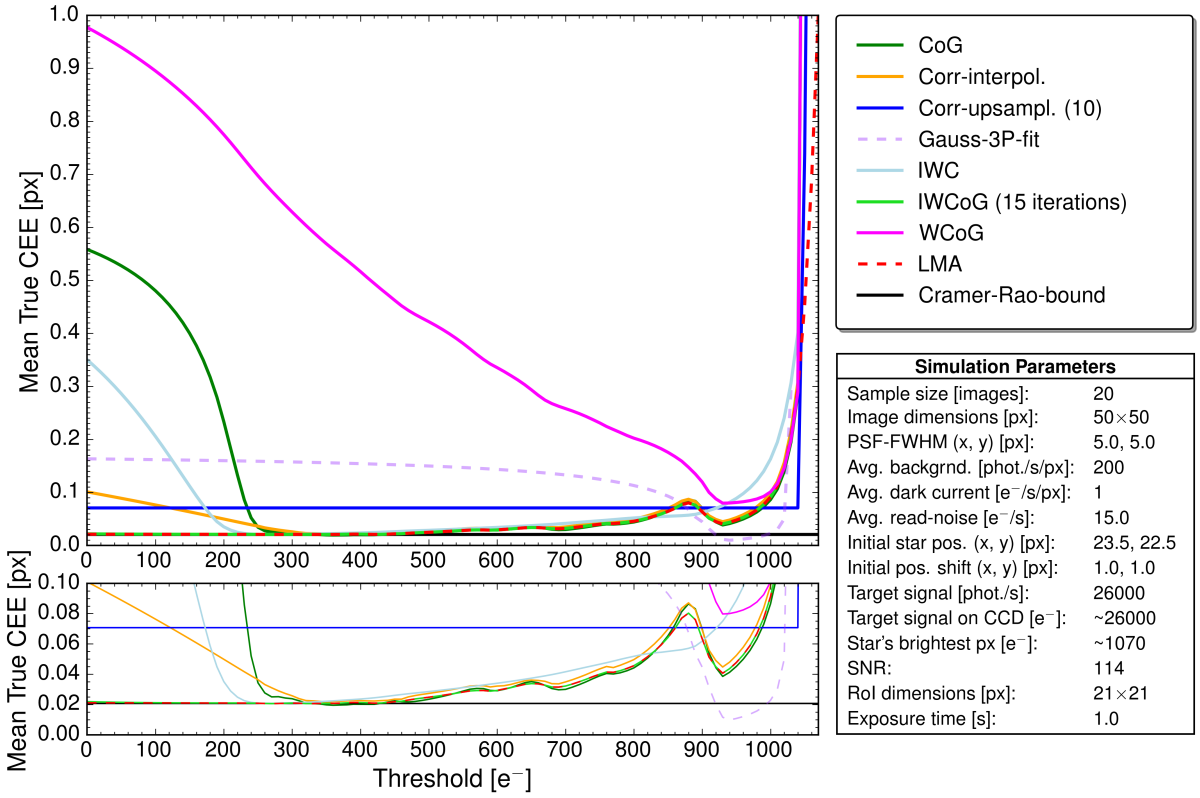

Figure 3.23.: The dependency of various centroiding algorithms on thresholds. The mean true CEE was computed out of twenty images for each threshold level. The lower panel represents a zoom on the Cramer-Rao bound.

CoG displayed a strong non-linear behaviour for different thresholds. The application of a threshold that removed half of the background, only improved the CEE by about fifteen percent. Thresholding the entire background led to a performance at the Cramer-Rao bound. Over-thresholding slightly increased the CEE up to thresholds below 1 000 e^-. For even higher thresholds the stellar signal was almost entirely removed which caused an escalation of the CEE.

WCoG showed almost linear dependency on thresholds. However, the computed CEE did not reach the Cramer Rao bound, due to the displacement of the start position. This has been further discussed in Section 3.5.2.

IWC featured a linear dependency on thresholds, as well. IWC reached the Cramer-Rao bound faster than other methods, as it is depicted in the lower panel of Figure 3.23. For further increased thresholds IWC behaved like the standard CoG.

IWCoG and **LMA** started off at the Cramer-Rao bound. Thus thresholding did not improve the centroid quality of these methods. However, over-thresholding affected both methods in the same way as the other CoG-based algorithms.

Gauss-3P-fit did not require thresholding either. The decrease of the CEE in the extreme over-thresholded region was only caused by a favourable star position for the three-point-fit. This does not occur in general.

Correlation-upsampling is the only method showing absolutely no dependency on any of the applied thresholds. It is therefore considered to be entirely invariant to uniform background signals as well as over-thresholding. The computed CEEs correspond to the lowest possible limit defined by the upsampling factor of ten (see Section 3.3.4).
Correlation-interpolation displayed a weak linear dependency on thresholds. I suggest to always apply thresholds for this method, as the background level could be orders of magnitude higher in real observations.

The correlation-upsampling method, together with LMA and IWCoG performed best in the presence of a uniform background signal. Thresholding the image is not a compulsory step for these methods.

3.5.4. Impact of Cosmic Ray Hits

The impact of cosmic ray hits on the centroiding methods was analysed for two different scenarios. Figure 3.24 illustrates sample images of these scenarios. The cosmic ray hits were distinguished by their location. In fact, it was distinguished whether they occurred outside (type I) or inside the stellar spot (type II). The origin of cosmic ray hits and the related glitches are discussed in Section 2.8. Note that the following analysis is restricted to the two glitch positions illustrated in Figure 3.24. An investigation including random glitch positions for various glitch frequencies is provided in the mission specific Chapter 4.2.4.2.
Figure 3.25 and 3.26 summarise the simulation results for the two glitch scenarios. In both scenarios, the glitch signal was gradually incremented up to 5 000 e^- in steps of 100 e^-. The initial centroid estimation was constantly displaced by 0.5 pixel in both dimensions. The mean true CEE was computed out of twenty images per glitch signal. The scatter was negligibly low for all methods, due to the high SNR of 170. Only a few methods were affected by glitches of type I, but all algorithms showed distortion in the second scenario. Both scenarios are now discussed for each method respectively.

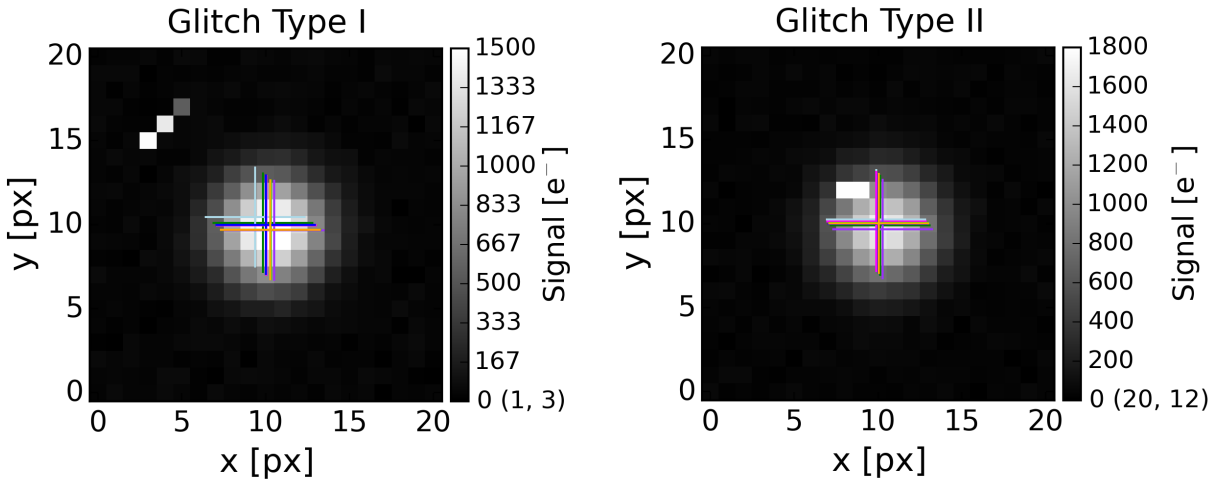

Figure 3.24.: Both sample images depict the same star with distinct cosmic ray hits. They show the thresholded region of interest. *Left*: cosmic ray hit outside the stellar spot (Type I). *Right*: cosmic ray hit inside the stellar spot (Type II). Crosses indicate results of the centroiding algorithms and the color codes are the same as in Figure 3.25. Images were generated with *StarSim*.

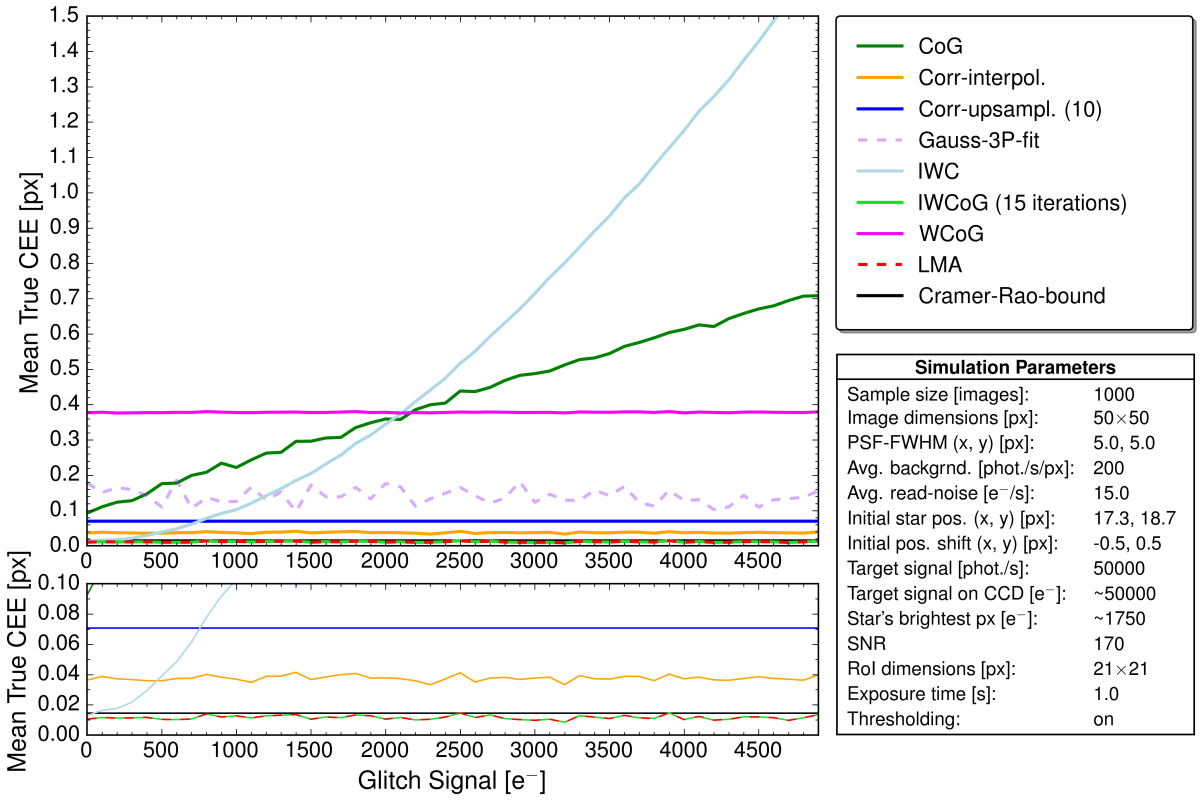

Figure 3.25.: The impact of a type I glitch on the performance of centroiding algorithms. A sample image is provided in Figure 3.24. The mean true CEE was computed out of twenty images for each glitch signal. The lower panel represents a zoom on the Cramer-Rao bound.

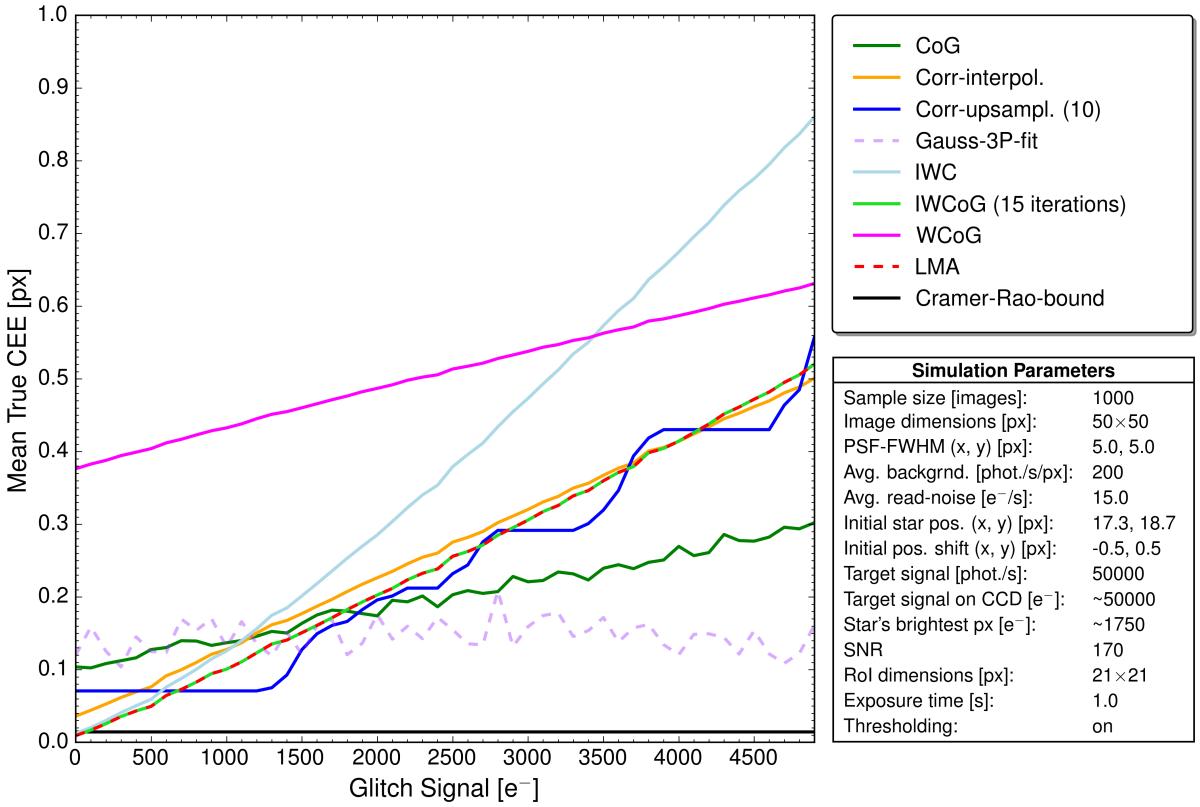

Figure 3.26.: The impact of a type II glitch on the performance of centroiding algorithms. A sample image is provided in Figure 3.24. The mean true CEE was computed out of twenty images for each glitch signal.

CoG showed a flat linear increase of the CEE in both scenarios. In the case where the glitch was situated farther away from the stellar spot, the impact was even stronger. This is due to the principle of the CoG algorithm, as all signals are weighted only by their position.

WCoG exhibited a distortion only in the second scenario. This is because the weighting function cut off all the signal outside the target spot. This is only possible for precise start positions. The initial offset in the CEE of 0.4 px was caused by the small displacement of the start position.

IWC squares the image prior computation of the centre of gravity. Therefore the glitch signal is also squared, thus the implications of glitches were even worse compared to the standard CoG.

IWCoG and **LMA** were not affected by glitches outside the target spot. Glitches inside the spot distorted the centroid estimation, where increased glitch signals caused larger CEEs.

Gauss-3P-fit displayed no influence form the glitch in both scenarios. This can be explained by the fact that only three pixels are used in both dimensions and none of these contained a glitch. However, if a cosmic ray hits one of these pixels, Gauss-3P-fit is definitely not capable of providing useful results.

Correlation-upsampling featured a slightly different behaviour than the other methods. In the analysis of type I glitches, correlation-upsampling provided unaltered centroid positions like many other methods. The computed CEEs correspond to the lowest possible limit that is defined by the upsampling factor of 10 (see Section 3.3.4). For glitches inside the stellar spot, this method featured the same dependency on the glitch signal as IWCoG and LMA at first sight. However, by looking closer at the region for glitch signals below 1500 e^-, it can be seen that correlation-upsampling was not affected by weak cosmic ray hits. In fact, type II glitches only affected the performance for glitch signals close to or greater than the brightest pixel in the stellar spot.
Correlation-interpolation displayed a similar behaviour than IWCoG and LMA. Glitches inside the stellar spot may produce large CEEs, whereas glitches outside the spot cause no distortion.

Best performance was shown by the correlation-upsampling method, due to the advantage for low glitch signals, followed by IWCoG, LMA and the correlation-interpolation method. The IWC featured the worst performance in the presence of glitches. It should be mentioned that no method was capable of providing accurate results near the Cramer-Rao bound in the second scenario. Glitches are a non-negligible source of errors particularly in regions where glitch rates are high, such as in the South Atlantic Anomaly (see Section 4.2.2). Therefore, deglitching techniques, like applying a median filter, should be considered in order to improve the CEE in such regions. This possibility has been analysed further in Section 4.2.4.2.

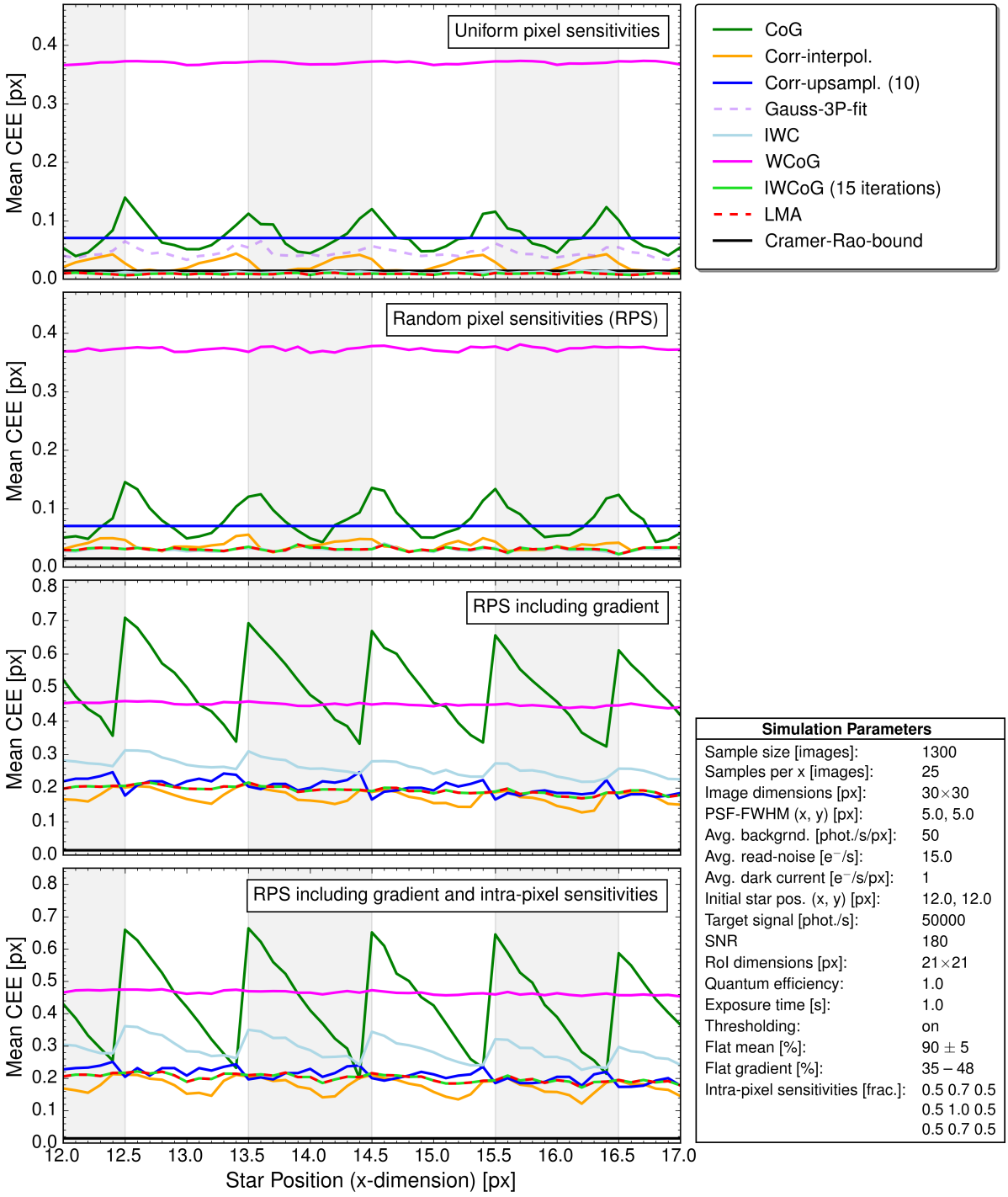

Figure 3.27.: The centroid estimation error for different intra-pixel positions of the target. The star position was moved along the x-axis in steps of 0.1 px. The gray shaded regions indicate the size of a pixel. Three scenarios are visualised. The panels are described from top to bottom as follows. **First panel:** Images were created with uniform pixel sensitivities. **Second panel:** Normal distributed pixel sensitivities were introduced with a mean sensitivity of $90 \pm 5\%$. **Third panel:** A sensitivity gradient was added where the sensitivity was increased from left to right. **Fourth panel:** Intra-pixel sensitivities were included according to the pixel-map listed in the simulation parameters. Pixel sensitivities vary from 100% (centre of pixel) to 50% (edge of pixel) inside each pixel. The Gauss-3P-fit is only shown in the first panel, as the provided CEEs were too high in the other scenarios. In fact, the results were one order of magnitude higher compared to the other methods. A more detailed description of these simulations is provided in the text.

3.5.5. Flat-Field: Intra-Pixel Star Positions and Pixel Sensitivities

The dependency on different intra-pixel star positions and flat fielding was tested by simulating the motion of the star along the CCD's x-axis. Therefore, the star was moved from position (12,12) to (17,12) in steps of 0.1 px. Figure 3.27 illustrates the mean CEE in four distinct cases for each position. From top to bottom the first panel represents images without any flat-field component, which implies that all pixels feature the same sensitivity. The second case includes random pixel sensitivities that were represented by a normal distribution with a mean of 90% and standard deviation of 5%. In simulations corresponding to the third panel, an additional sensitivity gradient was introduced to the images. This gradient was applied on the whole CCD along the x-axis with sensitivities ranging from $30 - 100\%$ percent (left edge to right edge). Thus, inside the RoI the pixel sensitivities were ranging from $35 - 48\%$ from left to right. The fourth panel depicts simulations including intra-pixel sensitivities. An introduction to the topic of intra-pixel sensitivities is provided in Section 2.7. For each star position, 25 images have been simulated, which led to a total number of 1 300 images per sample.
No flat-field correction was performed in any case. In fact, this analysis was carried out in order to determine the necessity of such a reduction technique. All scenarios shown in Figure 3.27 are now discussed for each method respectively.

CoG showed a strong dependency on the intra-pixel centroid positions even without the presence of a flat-field component. The mean CEE was clearly higher for positions closer to the pixel's edge. The introduction of random pixel sensitivities did not influence the performance. However, the introduction of the sensitivity gradient led to the saw tooth profile that is illustrated in panel three. The mean CEE was remarkably higher due to the gradient, as less signal was detected in total. By moving the star along the gradient the signal gets amplified, thus the peaks of the mean CEE decline. On intra-pixel scale the behaviour changed as well. The mean CEE was remarkably higher if the stellar peak was located close to the left edge of the pixel. This effect is caused by the direction of the gradient, as it was increased from left to right. In fact, if the peak is located on the left edge, more signal is contained in pixels on the left side. Therefore, the point spread function's left wing was affected stronger by the sensitivity gradient, which led to higher CEEs. A jump of the mean CEE on a pixel transition occurs directly at the pixel's edge, due to the gradient. This feature is slightly shifted by 0.1 px in Figure 3.27, due to the step size of 0.1 px. Panel four reveals the expected enlargement in the variation of the CEE, for different positions inside a pixel in the presence of additional intra-pixel sensitivities.

WCoG featured a similar behaviour as CoG on intra-pixel centroid positions (panel one). Here, this effect happened on a much smaller scale. Therefore it is negligible. Neither the introduction of small-scaled random pixel sensitivities (RPS), nor the inclusion of intra-pixel sensitivities did affect the CEE. Only the sensitivity gradient caused a hardly noticeable decrease of the CEE, when the star was moved along the gradient. Thus intra-pixel sensitivities are negligible as well.
IWC results were overlapped by LMA and IWCoG in the first two scenarios illustrated in Figure 3.27. In the first panel IWC is located at the Cramer-Rao bound and its performance was not affected by different intra-pixel positions. For the last two cases the behaviour was similar to the standard CoG, except that the total CEE and its variations were much lower.
IWCoG and **LMA** provided identical results in all four cases. No dependency on the intra-pixel position was observed. However, the RPS pushed the CEE slightly away from the Cramer-Rao bound and the gradient increased the total CEE significantly. Nonetheless, the gradient did not affect variations of the CEE on intra-pixel scale. The effect of intra-pixel sensitivities is also negligible.
Gauss-3P-fit was not included in some panels in Figure 3.27 due to its bad performance. In the first panel it can be seen that the method provides less accurate results if the peak is located at pixel edges. The introduction of only small RPS was sufficient to increase the mean CEE far beyond 0.4 px. Furthermore, the sensitivity gradient as well as the intra-pixel sensitivities caused small changes to the CEE, but their effects were completely overlaid by the RPS.
Correlation-upsampling performed with lowest possible errors, corresponding to the upsampling factor of 10 (see Section 3.3.4) in the first two cases. However, the introduction of the gradient led to an increased CEE. Furthermore, the behaviour was inverse to IWC on intra-pixel scale. However, the absolute CEE decreased along the gradient.
Correlation-interpolation strongly depends on the intra-pixel position of the stellar peak in all cases. Here, it is not safe to say that the CEE is lower at the pixel centre in general, as this could only be observed in the presence of the sensitivity gradient. Small RPS increased the total CEE and also changed the shape of the profile and therefore they are not negligible. Furthermore, the introduction of the gradient increased the total CEE significantly.

Best performance was achieved by LMA, IWCoG and both correlation methods in all cases. The Gaussian-3P-fit featured the worst performance as the introduction of RPS led to extreme CEEs. One of the most interesting outcomes of these analyses is the fact that the introduction of a sensitivity gradient affected all centroiding methods. Therefore, steep sensitivity gradients should be reduced prior to execution of any centroiding method. The behaviour of the algorithms for the different kinds of sensitivity variations are summarised in Table 3.1.

3.5.6. Computation Time

The computation time was analysed for different sizes of the RoI starting from 11×11 px to 30×30 px. The mean processing time for each algorithm was computed out of twenty images and is illustrated in Figure 3.28 for each centroiding method. It is normalised to the standard CoG (dashed green line) in order to make the comparison easier.

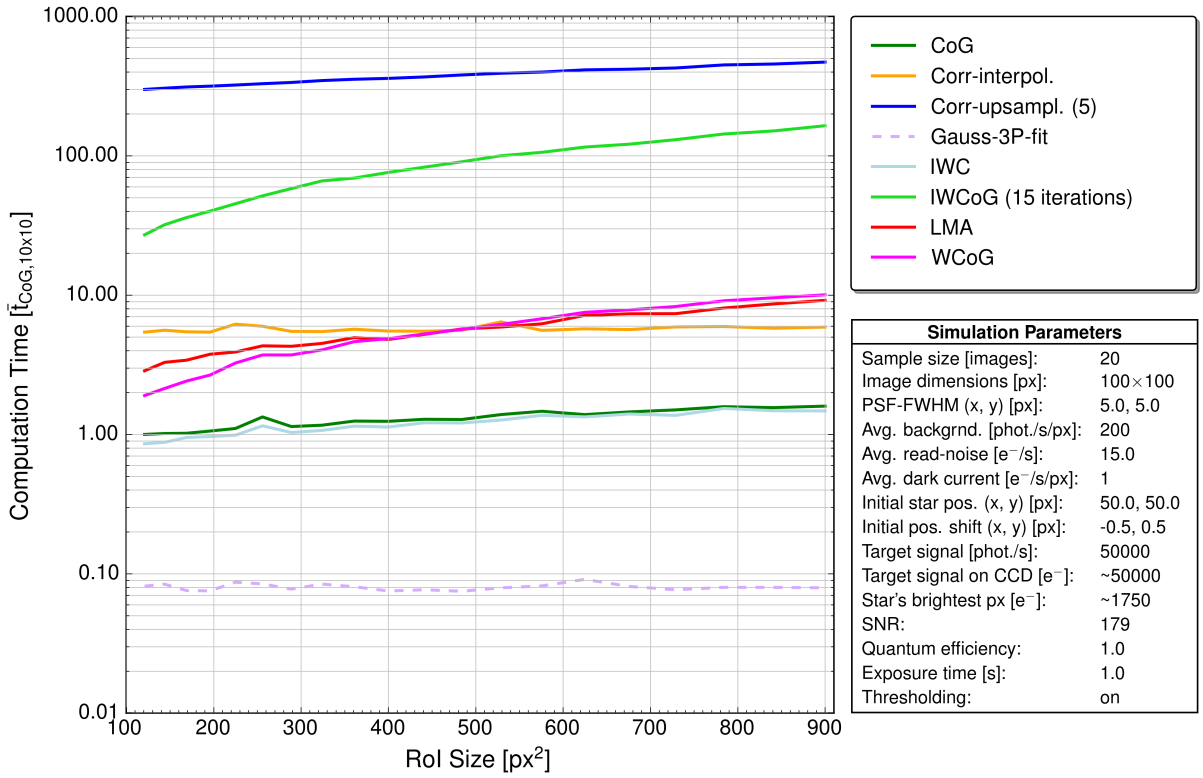

Figure 3.28.: Computation times of centroiding methods for different RoI sizes. The computation time has been normalised to the mean processing time of a standard CoG for a Roi size of 11×11 px. The RoI size was gradually increased from to 30×30 px.

CoG together with **IWC** displayed similar processing times. Both showed a non-linear dependency on increased RoI sizes. Only the three-point fit required less computation time. Apart from the computation time, a bigger RoI size also affects the CEE for CoG and IWC, as bigger windows contain additional noise and background.

WCoG's processing time increased for bigger RoIs, as the pixel-weighting is carried out on a larger scale. The simulation revealed that WCoG required almost four times the processing time of the standard CoG for a RoI of 20×20 px.

IWCoG was performed with fifteen iterations. This led to computation times that were exactly fifteen times higher than for WCoG.

Gauss-3P-fit indicated no dependency on the RoI size in terms of computation time. This can be explained by the fact that only three pixels are applied for the fit per dimension. In this case the size of the RoI does not matter. The mean processing time took only 6% of a standard CoG computation. Therefore, this method represents the fastest of all investigated centroiding algorithms.
LMA required less processing time than IWCoG, although their performance was almost identical in other analyses. The difference is mainly caused by the fact that IWCoG was carried out with a fixed number of iterations, whereas the iterations used for the parameter optimisation in the LMA stopped as the CEE reached a certain limit. It is not safe to say that LMA is faster than IWCoG in general.
Correlation-upsampling was computed with an upsampling factor of 5. The computation time was equal to about 300 computations of CoG, thus correlation-upsampling required by far the most processing time. Apart from that, the increase in the computation time for bigger RoIs was not linear.
Correlation-interpolation only took about $4-5$ times the computation of standard CoG. It was much faster than the upsampling method, but it also featured a slight dependency on the RoI size.

The lowest processing time was scored by the Gauss-3P-fit, followed by CoG and IWC. In general, these are the fastest of the investigated centroiding methods. The highest processing time was required for the correlation-upsampling method.

3.6. Conclusion

The centroiding methods presented in this chapter were grouped into three different types. Centre of gravity algorithms, correlation-based centroiding and finally centroid detection by direct fitting. A summarised description of each centroiding method is provided in their corresponding section. Simulations revealed that there exists no ultimate centroiding algorithm that is suitable in each and every case. The choice for the best centroiding method depends on factors like the signal-to-noise ratio, cosmic ray hit frequency and others that were introduced in Section 3.5. The dependency map in Table 3.1 represents a compact overview of the performance analysis presented in this Chapter.

The impact of each component was categorised as *invariant*, *very low*, *low*, *medium*, *high* or *very high* with respect to the relative performance of the other methods. Additionally, *n/a* denotes that the method is 'not applicable', or more precisely it's not capable of providing useful results. 'SNR' stands for signal-to-noise ratio and 'background thresholding' represents the process of stripping unwanted signal originating from background objects (see Sections 3.5.1 and 3.5.3). 'RoI displacement' represents a region-of-interest that is not centred on the target. 'Position offset' denotes the offset of the initial centroid estimation to the true centroid, which may be located anywhere inside the RoI (see Figure 3.22). The impact of two different types of cosmic ray hits was analysed. A visualisation of these two types is depicted in Figure 3.24. 'Intra-pixel positions' were analysed in order to determine whether there is a difference in performance if the stellar peak is situated at the pixel's edge or centre (see Section 3.5.5). Simulations regarding the influence of different pixel sensitivities included, 'Random pixel sensitivities', a 'sensitivity gradient' as well as 'intra-pixel sensitivities' which were introduced in Section 2.7. Finally, the mean computation times were presented as multiples of the processing time that is required for the standard CoG.

By looking at Table 3.1 one can identify the techniques which performed best in most analyses. In fact, those are IWCoG, LMA and the correlation methods. However, these methods also required the highest processing times. Therefore, they may not be applicable for all space telescopes, due to the hardware limitations. However, IWC featured low processing times and it performed really well in the absence of glitches, if appropriate thresholds were applied. A selection of centroiding methods for the two space missions, EChO and CHEOPS is discussed in the next Chapter.

	CoG	IWC	WCoG	IWCoG[a]	Correlation-interpolation	Correlation-upsampling	Gauss-3P-Fit	LMA
SNR	high	medium	medium	low	low	low	very high	low
Background Thresholding	very high	high	high	very low	low	invariant	very low	very low
RoI Displacement	high	medium	invariant	invariant	invariant	very low[b]	invariant	invariant
Position Offset	invariant[c]	invariant[c]	very high	invariant[d] very high[e]	invariant	invariant	very high	invariant[d] n/a[e]
Intra-Pixel Positions	high	very low	very low	very low	medium	invariant	medium	very low
Cosmic Ray Hits								
Glitch Type I	high	very high	invariant	invariant	invariant	invariant	invariant	invariant
Glitch Type II	medium	very high	high	high	high	invariant[f] high[g]	n/a[h]	high
Flat-Field Components								
RPS[i]	low	low	low	low	low	low	very high	low
Sensitivity Gradient	very high	very high	low	low	medium	medium	very low	low
Intra-pixel Sensitivities[j]	very low	very low	very low	very low	very low	very low	medium	very low
Computation Times[k] $[\bar{t}_{\mathrm{CoG}}]$	1	$\sim$1	4	12[k]	4-5	300	0.06	4

Table 3.1: Dependency map illustrating the impact of various components on the performance of the centroiding methods. For further understanding please read Section 3.5.

[a] performed with fifteen iterations.
[b] for star positions close to the RoI's border, otherwise invariant.
[c] No initial centroid position is required for this method, except for the RoI placement.
[d] for offsets below $5 \cdot \sigma_{\mathrm{PSF}}$.
[e] for offsets greater than $5 \cdot \sigma_{\mathrm{PSF}}$.
[f] for glitch signals lower than the stellar spot's brightest pixel.
[g] for glitch signals higher than the stellar spot's brightest pixel.
[h] Not applicable if the glitch is affecting one of the five pixels used for the fit. Otherwise invariant.
[i] Random pixel sensitivities (RPS). Their impact depends on the size of the pixel sensitivity distribution.
[j] for high SNRs. The influence of intra-pixel sensitivities becomes more important for lower SNRs.
[k] for a RoI size of 20×20 pixel and a position offset of 0.5 pixel in both dimensions. IWCoG was performed with three iterations.

4. Space Applications

4.1. EChO - Exoplanet Characterisation Observatory

The **E**xoplanet **Ch**aracterisation **O**bservatory had been proposed for the third medium class launch slot (M3) for the years 2022-2024 as part of the European Space Agency (ESA) programme *Cosmic Vision*. In the context of this programme several large and medium class missions as well as one small class mission **CHEOPS** (see Section 4.2) were specified so far. In addition to EChO, the following mission candidates competed for the M3 launch opportunity: **PLATO** (Planetary Transits and Oscillations of stars), **LOFT** (the Large Observatory For x-ray Timing), **MarcoPolo-R** and **STE-Quest** (Space-Time Explorer and Quantum Equivalence principle Space Test). Eventually, **PLATO** was selected by ESA's Science Programme Committee in February 2014[1]. The contribution of Vienna's Institute of Astrophysics to EChO included the payload instrument software for the Fine Guidance Sensor (see Section 4.1.2).

4.1.1. Mission Description

A summarised mission description, including the scientific objectives and the mission requirements, is provided below. Mission details were taken from the EChO Assessment Study Report (Drossart et al., 2013) if not stated otherwise.

EChO was designed to carry out spectroscopic measurements of exoplanet atmospheres in order to answer questions like: *Why are exoplanets as they are? What are they made of? What are the causes for the observed diversity? Can their formation history be traced back from their current composition and evolution?* The target list covers Hot Jupiters, Neptune-like planets as well as lower-sized planets down to super-Earths. The final target list contained a few hundred host stars with spectral types varying from F to M and visual magnitudes ranging from 6 to 12. The list was considered to be extended by results of future instruments like the European Extremely Large Telescope (E-ELT) or the James Webb Telescope (JWST). Observations were planned to be carried out within four years using transmission and eclipse spectroscopy.

[1] ESA Media Relations Office, http://sci.esa.int/plato/53707-esa-selects-planet-hunting-plato-mission/, accessed: 06 October 2014

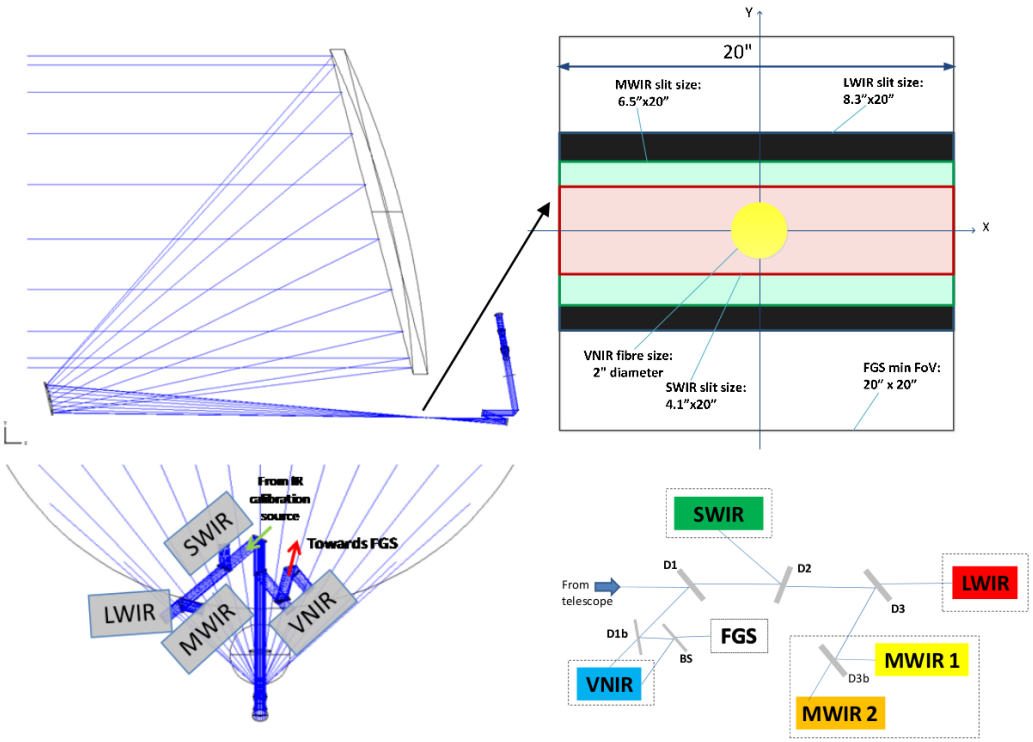

Figure 4.1.: **Upper left**: Optical configuration of EChO consisting of three elliptical conic mirrors in a Korsch-like assembly. **Upper right**: Field of view for the different instruments. **Lower left/right**: Beam divided into different channels by using beam splitters and dichroics. Eccleston et al. (2014)

The configuration of the optical system consists of the three conic mirrors illustrated in Figure 4.1. The Korsch-like assembly leads to an effective total collecting area of 1.13 m^2. The intended range of wavelength coverage is 0.55 - 11 µm with the additional goal to extend the range to 0.4 - 16 µm. Only a few molecules could be detected in exoplanet atmospheres so far. The broad wavelength coverage together with EChO's design and stability enables to detect four times more molecules compared to current methods. This can already be achieved with low spectral resolutions of R∼300 for $\lambda < 3$ µm and R∼30 for $\lambda > 3$ µ. Besides the detection of atmospheric constituents, EChO will reveal the atmospheric temperature as well as the planet's albedo. The satellite was planned to perform operations at the Sun-Earth Lagrange point L2.

Planet transits have to be observed in order to gain spectra of exoplanet atmospheres. An illustration of such a scenario is provided in Figure 4.2. As a planet transit occurs, the stellar light passes through the planet's atmosphere at the terminator. Therefore the resulting spectrum contains information about the present molecules within high-altitude layers of the planet's atmosphere. This information can be extracted by comparing the measurements with the unmodified stellar spectrum (*transmission spectroscopy*). EChO's goal is to measure atmospheric features with signals at least 10^{-4} times lower than stellar spectral features. The planet is com-

pletely occulted during a secondary eclipse, because the transit occurs on the backside of the host star. The planet day-side spectrum may be obtained by comparing spectra taken during such an eclipse with spectra observed before or afterwards (*secondary-eclipse spectroscopy*). This allows to probe lower atmosphere layers with higher pressure-levels, which is not possible with transmission spectroscopy. Additionally, more time consuming measurements are carried out to catch *planet phase variations*. Such variations are caused by changes in the observed part of the planet's atmosphere, due to the planet's rotation. They allow to derive details about atmospheric dynamics. The three methods work best between optical and mid-IR wavelengths.

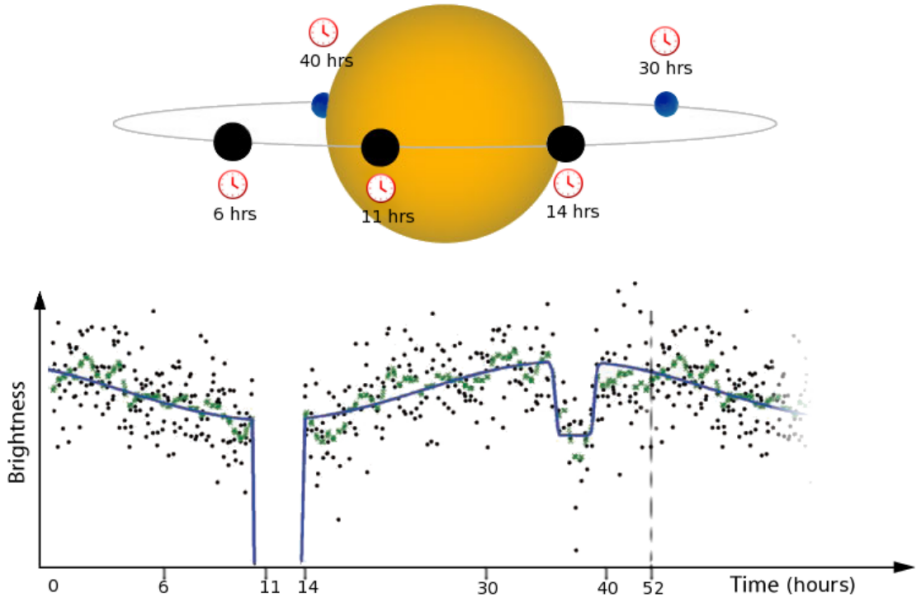

Figure 4.2.: **Upper image**: Planet orbit including transit and eclipse. **Lower panel:** Depicts the light curve of planet HAT-P-7b observed by Kepler (Borucki et al., 2009). Blue and black circles indicate the planet's day side and night side respectively. (Drossart et al., 2013)

4.1.2. Fine Guidance Sensor

A Fine Guidance Sensor (FGS) is a spacecraft subsystem dedicated to provide high precision pointing information for focusing, centering and guiding. A long-term photometric stability of about ten hours is one of EChO's most stringent mission prerequisites (Ottensamer et al., 2014). During such operation, the mean pointing error relative to the target star should not exceed 10 milliarcseconds. This may only be achieved by implementing a FGS, in addition to the star trackers. Primarily two different effects contribute to the pointing errors. The drift of the spectrum along the detector's spectral axis (*spectral jitter*) and the drift along the spatial direction (*spatial jitter*). As the FGS represents an additional imaging device, it may also be used as a science instrument (Nelan et al., 1998; Doyon et al., 2012). In the particular case of EChO it was envisaged to utilise the FGS for performing on-board photometry. The additional

astrometric and photometric information would further improve on-ground data calibration of the measured spectra.

As it can be seen in Figure 4.1, the main beam is divided by dichroics into four different spectral channels (VNIR, SWIR, MWIR and LWIR). The wavelength range of each channel is depicted in Figure 4.3. Only a fraction of light that is passed to the VNIR channel is deflected towards the FGS (0.4 - 1.0 μm). A beam splitter divides the light into equal parts for the VNIR and FGS subsystem. Therefore changes in the line of sight can be measured directly. However, such an optical configuration reduces the incident signal that is reaching the FGS, significantly. In general, this causes lowered signal-to-noise ratios, which may affect the performance of the implemented centroiding method (see Section 3.5.1).

System specifications in particular for the distinct optical configuration of the FGS (see Figure 4.4) are listed in Table 4.1. The star trackers are capable of locating the target with an accuracy of 10" rms. The FGS takes over centroiding, focusing and guiding tasks as soon as the target is positioned within the 20"×20" FGS field of view (handover task). Guiding is performed at a rate of 10 Hz in a 7"×7" region-of-interest (RoI) within this field of view (science mode). The pixel scale is about 0.1", therefore the size of the RoI corresponds to a 70×70 pixel array. The mission requirements demand that the mean centroid estimation error shall not exceed 10 milliarcseconds or (0.1 pixel).

The FGS consists of two distinct physical components: the FGS Control Unit (FCU) that is mounted on the service module and the opto-mechanical box with Gregorian optics (see Figure 4.4). Further details about these subsystems including their hardware specifications are provided by Ottensamer et al. (2014). The FGS is only dedicated to perform astrometry, but it does not control the attitude of the spacecraft. In fact, the centroid estimations are used as input for the Attitude and Orbit Control Subsystem (AOCS) to maintain the required pointing stability.

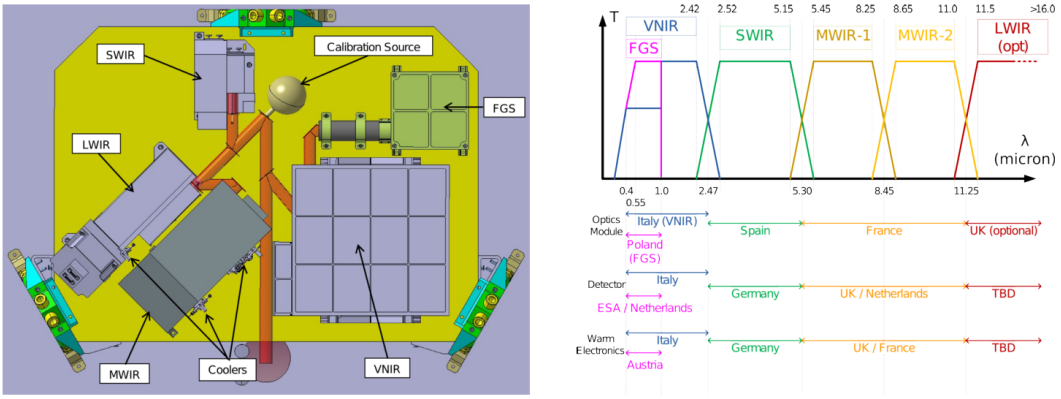

Figure 4.3.: **Left**: Instrument channel configuration on the optical bench. See also schematic design given in Figure 4.1 in the lower-right panel. **Right**: Operating wavelength range for each channel respectively. (Eccleston et al., 2014)

System property	
Focal length	1 300 mm
Optical System length	210 mm
Internal FOV	0.33 deg × 0.33 deg (full)
F-number (total)	52
PSF size (FWHM)	50 µm × 34 µm
Mirrors	1: parabolic, 2: spherical, flat folding mirrors
Central obscuration	8%

Table 4.1.: Properties of the FGS optics. (Ottensamer et al., 2014)

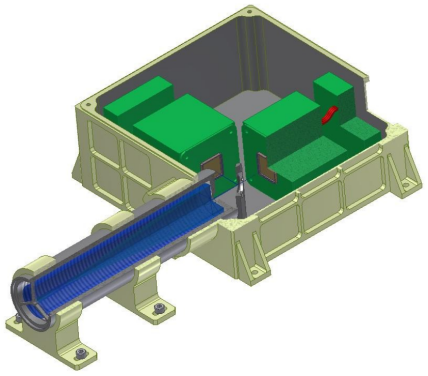

Figure 4.4.: Layout of the opto-mechanical box. The mirror set-up conforms to a Gregorian design. Eccleston et al. (2014)

4.1.3. Point Spread Function

The model of the diffraction limited PSF is illustrated in Figure 4.5. It features a FWHM at 50×34 µm, which corresponds to 3.33×2.26 pixels on sensor scale (see Section 4.1.4). The model depicted in the lower right panel of Figure 4.5 was created with *StarSim*. It was applied in the following analyses regarding the selection of the centroiding methods. The model of the PSF may also be expressed by standard deviations as $(\sigma_x, \sigma_y) = (1.42, 0.96)$ pixel, according to Equation (2.5).

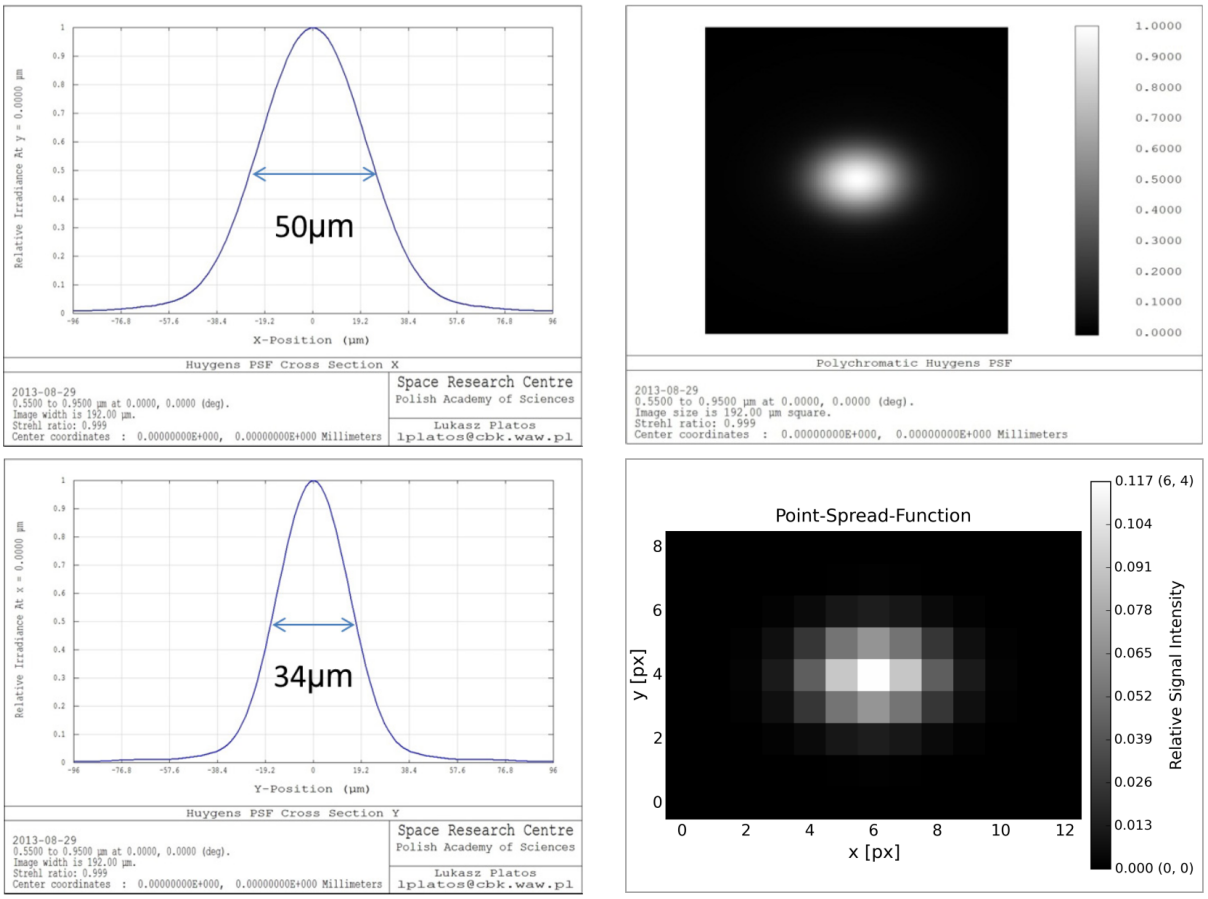

Figure 4.5.: Model of the diffraction limited PSF of the FGS provided by the Polish Academy of Sciences. In the left panels cross-sections of the PSF-model are depicted for both dimensions. The corresponding two-dimensional model is plotted in the upper right panel for high resolution. The lower-right panel illustrates the two-dimensional model on sensor resolution for a pixel scale of 15 µm. It was created with *StarSim*.

4.1.4. Centroiding in Different Operating Modes

The choice of the optimal centroiding method depends on the characteristics of the optical sensor. Therefore, an analysis with respect to the planned instrument hardware was carried out. Two possible detectors were considered for the FGS (EChO Assessment Study Design Report EChO Consortium Study Team (2013)). These are the HAWAII-2RG from Teledyne and a MCT device from SELEX ES. Both detectors mainly feature similar parameters. The following simulations were performed with respect to these sensors, whereas worst-case values were considered (in terms of centroiding). Eventually, the simulated hardware features included pixel scales of 15 µm, a read noise component of about 15 $e^{-}px^{-1}$ rms and the quantum efficiency was set to 70%. The mean dark current at operating temperature is below 0.03 e^{-} s^{-1} px^{-1} and the full-well capacity (FWC) was set to 50 000 e^{-}. The sensor was simulated with 200×200 px according to the 20”×20” field of view. It has been assumed that the amount of hot-pixels is

one percent on average. Observations were simulated with an exposure time of 0.1 s, as guiding is performed in a rate of 10 Hz. Slight variations in pixel sensitivities were assumed. However, image calibration was performed with simulated dark and flat-field frames before the centroids were estimated. Such calibrations were performed in order to reduce bias, pixel sensitivity variations and dark current (including hot-pixels). The simulated intra-pixel sensitivities, which featured a 60% loss of signal on pixel edges compared to the pixel centres, were not removed.
For EChO, two different centroiding tasks are required. Initially the star has to be positioned on the detector of the FGS as soon as it enters the field of view. Later on, centroiding is performed during science mode in which the centroid estimation can already be provided from previous runs of the centroiding task. In the sections below, analyses of applicable centroiding methods are presented for these two distinct cases. Observations of GJ 1214 were simulated as it represents the faintest star according to the target list in the Science Requirement Document (EChO Science Study Team, 2013). This extrasolar planetary system includes the exoplanet GJ 1214 b, which is classified as super-Earth. The distance of GJ 1214 is 13 pc and it features an apparent magnitude of V = 14.71. The expected total flux in the 0.4 - 0.8 µm band is $F_\star$ = 6.4·10^{-14} Wm^{-2} (EChO Science Study Team, 2013). The corresponding power on sensor level, P_{det}, can be computed by

$$P_{\star,det} = F_\star \cdot A \cdot F_{BS} \tag{4.1}$$

where A is the collecting area of the telescope (1.13 m^2) and F_{BS} is the amount of flux transmitted by the beam splitter to the sensor (50%). It was assumed that the aperture of the FGS subsystem is large enough to avoid an additional loss of signal. According to Equation (4.1), the total power on sensor level for GJ 1214 is $P_{\star,det}$ = 3.619·10^{-14} W. By assuming a worst-case scenario, where all photons have short wavelengths (0.4 µm), a total amount of about 72 880 ph s^{-1} reach the detector. Further signal reduction, due to the quantum efficiency of the sensor, is separately included in later simulations. The observation of GJ 1214 represents a requirement defined as R-SCI-120 in the Science Requirement Document (EChO Science Study Team, 2013). Additionally, the goal G-SCI-125 includes the observation of an even fainter source. In particular, the observation of a M5V star with a distance of 20.6 pc and a Ks-band magnitude of 9.8 is desired. By applying the same steps as for GJ 1214, a total photon flux on sensor level of 23 900 ph s^{-1} was derived. Both targets are included in the simulations presented in Sections 4.1.4.1 and 4.1.4.2.

The background signal was estimated by the amount of Zodiacal light that is emitted in the observed wavelength band. A profile of the zodiacal background is provided in the Mission Requirements Document (EChO Team, 2013) and can be rewritten as

$$Z(\lambda) = 3.5 \cdot 10^{-14} \cdot B_\lambda(5500\ \mathrm{K}) + 3.58 \cdot 10^{-8} \cdot B_\lambda(270\ \mathrm{K}) \tag{4.2}$$

where Z is in units of $\mathrm{Wm^{-2}sr^{-2}m^{-1}}$ and B_λ represents Planck's law for given temperature. The intensity of the zodiacal background depends on the viewing direction. Worst-case observations are performed at an elliptical latitude of 0° and a solar angle of 55°. In this case, the zodiacal background was estimated with $8\mathrm{Z}(\lambda)$ (EChO Team, 2013). With respect to the observed wavelength band of 0.4 - 0.8 µm, Equation (4.2) can be expressed as follows

$$Z(0.4\mu\mathrm{m}\text{ - }0.8\mu\mathrm{m}) = 3.5 \cdot 10^{-14} \int_{0.4\,\mu\mathrm{m}}^{0.8\,\mu\mathrm{m}} B_\lambda(5500\ \mathrm{K})\,\mathrm{d}\lambda + 3.58 \cdot 10^{-8} \int_{0.4\,\mu\mathrm{m}}^{0.8\,\mu\mathrm{m}} B_\lambda(270\ \mathrm{K})\,\mathrm{d}\lambda \tag{4.3}$$

In order to obtain the background flux per pixel, the steradians need to be converted to the pixel scale of 0.1"×0.1". This leads to the following final expression for an upper limit of the background flux.

$$F_Z = 8 \cdot Z(0.4\mu\mathrm{m}\text{ - }0.8\mu\mathrm{m}) \cdot \left[\frac{0.1 \cdot \pi}{180 \cdot 3600}\right]^2 \tag{4.4}$$

Similar to Equation (4.1) the power on sensor level can be computed by

$$P_{Z,det} = F_Z \cdot A \cdot F_{BS} \tag{4.5}$$

which leads to a maximum total power of $P_{Z,det}$ = 2.751·10^{-19} W for the zodiacal background. In a worst-case scenario, I assume that all photons feature a low energy (0.8 µm), which leads to a maximum photon count of 1.1 ph s^{-1} px^{-1}.

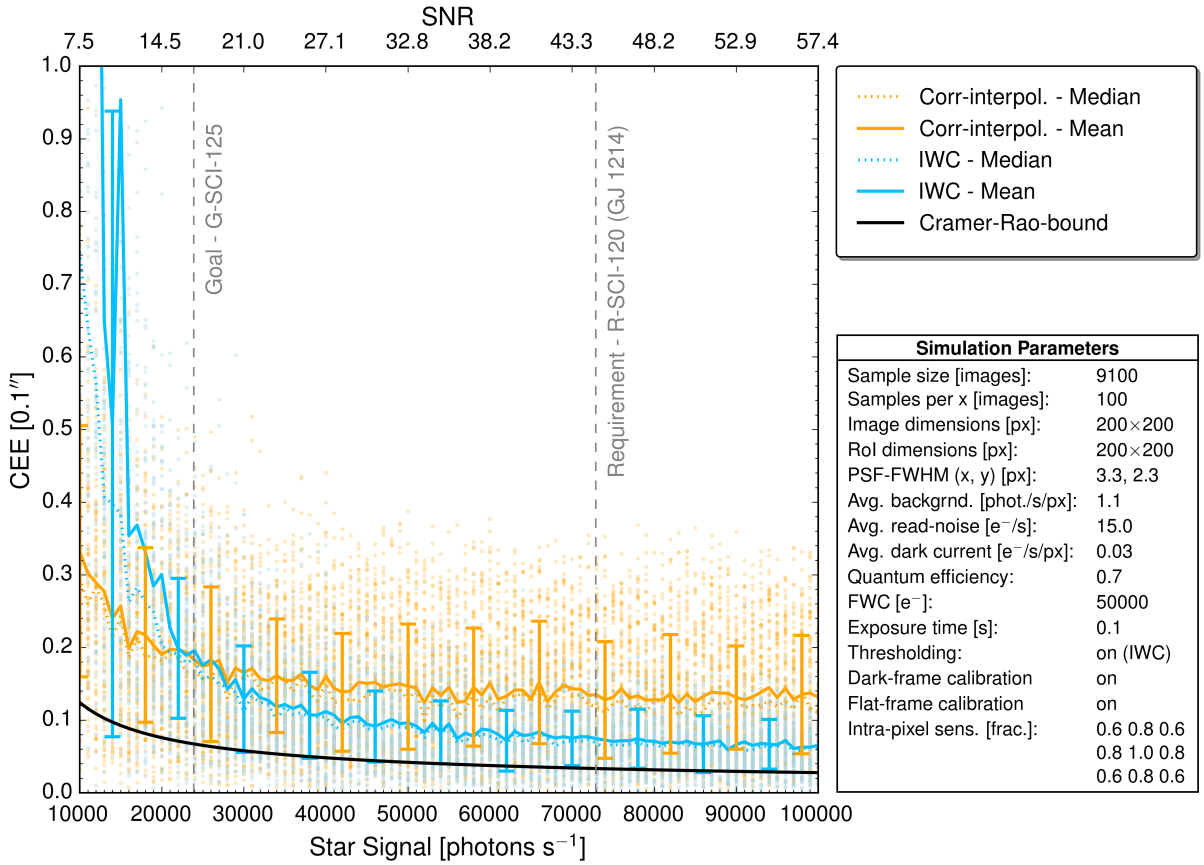

Figure 4.6.: In total 9 100 images were generated with *StarSim* with respect to the hardware characteristics of the FGS, where the target was placed at a random position in each image. The dots represent the computed CEEs, where yellow dots indicate results of correlation-interpolation and blue dots indicate results of IWC. The coloured solid lines represent the mean true CEEs for both methods. The median values represent the most likely CEE. Error bars are symmetric as they represent the standard deviation of the mean true CEE. The vertical dashed lines indicate the observed photon flux of the faintest targets defined in the Science Requirement Document (EChO Science Study Team, 2013) (see also Section 4.1.4). Further details about the observation of GJ 1214 are provided by Tinetti et al. (2015).

4.1.4.1. Handover from Star Trackers

Star trackers provide the initial centroid estimation with an error of 10” rms in the handover task. Due to this large error, only centroiding methods which are invariant to initial position offsets are applicable. Such algorithms may easily be identified by looking at Table 3.1, which represents a summary of the analyses presented in Chapter 3. Therefore, the pre-selection of algorithms for the initial handover task includes CoG, IWC, correlation-interpolation and correlation-upsampling. In fact, CoG is not further considered as IWC resembles an improved version of CoG, that provides more accurate results in almost any case. Furthermore, correlation-upsampling is discarded as the processing times are too high to maintain a 10 Hz control loop. Therefore, the two remaining methods are IWC and correlation-interpolation. A Monte-Carlo simulation including 9 100 images was carried out to test the performance of both methods during the handover task on simulated FGS hardware under worst-case conditions (see hardware

description in Section 4.1.4). Figure 4.6 illustrates the centroiding performance, where each individual point indicates the centroid estimation error (CEE) for one simulated image. The target was placed on the sensor at random positions inside a 200×200 pixel grid. The blue and yellow line indicate the mean true CEE for IWC and correlation-interpolation, respectively. Thesholding was only carried out for IWC, since correlation-interpolation is invariant to such process (see Figure 3.23). Without setting appropriate thresholds, IWC is not capable of providing useful results, otherwise large errors may occur if the star is not centred inside the RoI (see Section 3.5.2). In this simulation, the RoI spans across the entire FGS as the star may be located anywhere on the sensor.
Figure 4.6 reveals that IWC and correlation-interpolation are both applicable for the centroiding task during the initial handover from the star trackers. The mean true CEE is below 0.02" for observations of the faintest target defined by the goal G-SCI-125. Such accuracy is assumed to be sufficient, by considering that no science data is obtained during the temporary handover task. IWC performs slightly closer to the Cramer-Rao bound for brighter stars. However, I suggest to implement correlation-interpolation instead of IWC as it provides more reliable results. In fact, correlation-interpolation also provides accurate results in the presence of a second star inside the RoI. Furthermore, it does not require background thresholding and it is also applicable if cosmic ray hits occur (see Section 3.5). The processing times of both methods were almost equal in the simulations. Both methods are not capable of providing results with sufficient accuracy during science mode. Therefore, additional methods were analysed in the next section.

4.1.4.2. Centroiding during Science Mode

It can be seen in Figure 4.6 that IWC, and correlation-interpolation do not fulfil the pointing requirement of 0.01" during science mode. For this mode, LMA and IWCoG are potential methods, as they provide more accurate results in general (see Figure 3.5.1). However, these are not applicable for the handover task, as they require precise centroid estimations as input, which cannot be obtained by the star trackers for the FGS. However, for guiding during science mode the centroid estimations from the handover task may be applied as initial input to enable the execution of IWCoG and LMA. This can be seen in Figure 4.6, as the expected initial centroid estimation is below 0.02", which is significantly lower in comparison to the pointing error of 10" rms provided by the star trackers. In the worst-case the pointing jitter due to spacecraft stabilisation methods is about 0.13" rms (Waldman & Pascale, 2013). Therefore the following simulations were performed with a mean initial CEE of 0.15" (or 1.5 px). According to Figure 3.5.2, LMA and IWCoG are applicable for offsets below $5\sigma_{\mathrm{PSF}}$. Therefore, it is assumed that even larger pointing errors up to 0.71" in x and 0.48" in y are feasible in general.

In contradiction to the previous simulation, the random star positions are not spread across the whole sensor area, as it is intended during science mode to keep the star's position fixed in a specific sensor area. Therefore, random positions were restricted to a 10"×10" area around the detector's centre. Furthermore, the centroid estimation was performed inside a RoI of 7"×7".

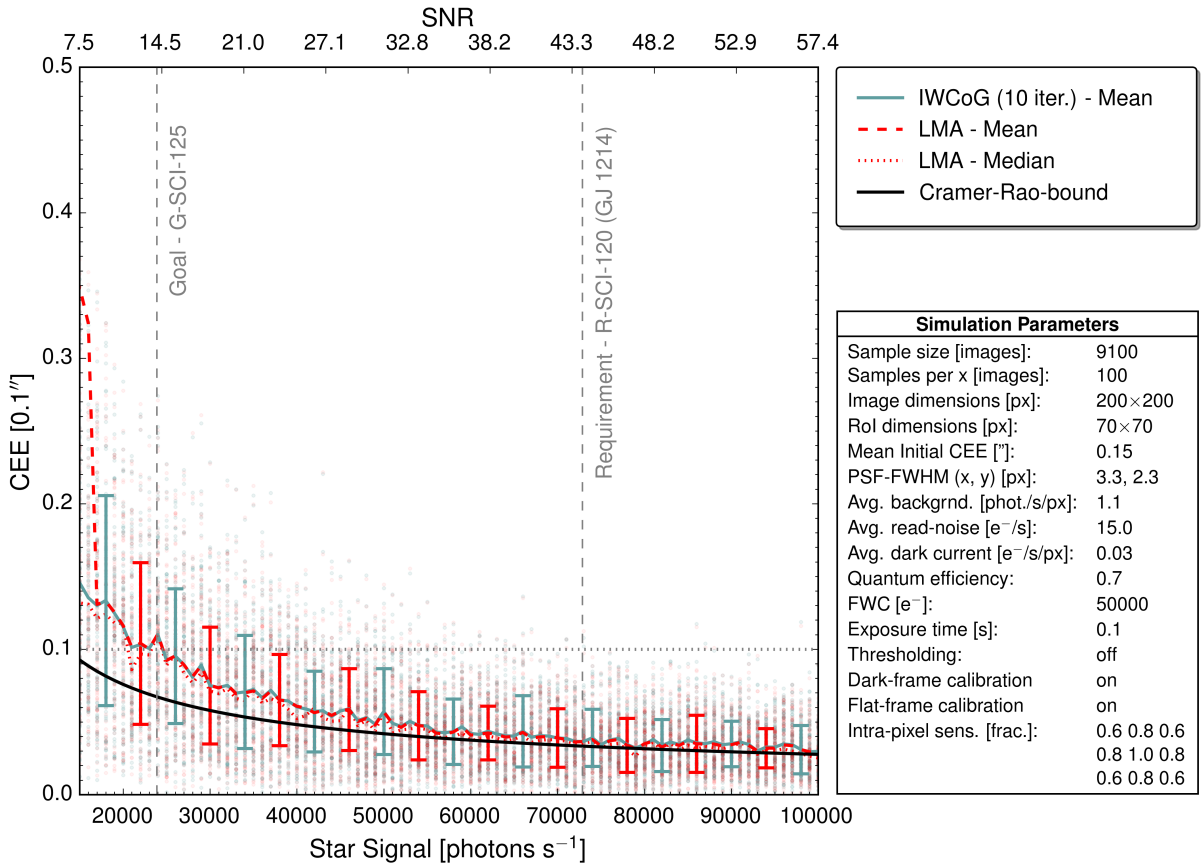

Figure 4.7.: In total 9 100 images were generated with *StarSim* with respect to the hardware characteristics of the FGS. The horizontal dotted line illustrates the fine guiding requirement of 0.01" during science mode. The target was placed at a random position in each image. The random positions were spread across an area of 100×100 pixel around the detector's centre. The dots represent the computed CEEs for both methods, as results overlap in almost any case. The coloured solid lines represent the mean true CEEs. The median values represent the most likely CEE. They overlap almost entirely for both methods, therefore the median was only plotted for one method. Error bars are symmetric as they represent the standard deviation of the mean true CEE. The vertical dashed lines indicate the observed photon flux of the faintest targets defined in the Science Requirement Document (EChO Science Study Team, 2013) (see also Section 4.1.4). Further details about the observation of GJ 1214 are provided by Tinetti et al. (2015).

Figure 4.7 illustrates that LMA and IWCoG are indeed capable of fulfilling the pointing requirements for the FGS. It is clearly visible that centroiding close to the Cramer-Rao bound is possible for observations of stars similar to GJ 1214. However, the performance for the observation of a fainter star, defined by the goal G-SCI-125, is slightly worse and already close to the limit of 0.01". Nevertheless, it is safe to say that the goal can be reached as the performed simulation represents a worst-case scenario. In fact, if it is not assumed that all photons feature short wavelengths, the true photon flux is between 23 900 and 47 800 ph s^{-1} (for G-SCI-125).

Within this region, the mean true CEE is clearly below 0.01” (dotted line). Certainly, based on the presented simulations, observations of even fainter stars are not feasible with current instrumental parameters, as the CEEs rise significantly for lower SNRs. The characteristic overlap of LMA’s and IWCoG’s results occurs as the same Gaussian PSF-model was used for the LMA fit-model as well as for the weighting function in IWCoG. The largely increased mean error of LMA for SNRs below 10 is caused by a few outliers. In this case the median error still overlaps with results provided by IWCoG. The processing time for 10 iterations of IWCoG was on average about 1.6 times higher than the mean computation time of LMA. No thresholding was performed in these simulations, as IWCoG and LMA are both invariant to the presence of background signal (see Section 3.5.3).

4.1.5. Conclusion

In order to achieve the pointing requirements for EChO, I analysed the application of centroiding methods on the fine guidance sensor (FGS). The star trackers are only capable of providing pointing information with an accuracy of 10” rms, which is too large to maintain the required photometric stability. Therefore the necessity of the FGS is inevitable in order to achieve the demanded fine pointing errors of about 0.01”. High precision centroiding methods were analysed for two distinct operating modes. These operations are the initial pick-up of the star by the FGS (handover task) and observations during science mode. Monte-Carlo simulations were carried out to create samples of 9 100 images with respect to the hardware specifications of the FGS. The two centroiding methods IWC and correlation-interpolation were considered for the handover task. Simulations revealed that both methods are applicable. However, the correlation method is preferred, as it is more stable in the presence of background signal and cosmic ray hits. Furthermore, this method is also applicable if additional stars are imaged by the FGS. Correlation-interpolation cannot be applied in science mode as the CEE of about 0.02” is too large. For centroiding during science observations, the centroiding techniques IWCoG and LMA were analysed. These methods require a precise initial centroid estimation, which is already acquired by the handover task. Simulations of worst-case scenarios showed that both methods are capable of fulfilling the goals defined in the Mission Requirements Document (EChO Team, 2013). These goals require performance close to the Cramer-Rao bound, which represents a theoretical limit for centroid estimation (see Section 3.1). It was assumed that cosmic ray hits play a less important role for centroiding, as exposure times are very short in this mission. Detailed analyses about the impact of cosmic ray hits on centroiding performance are presented in Section 4.2.2.

4.2. CHEOPS - CHaracterising ExOPlanets Satellite

The **CH**aracterising **E**x**OP**lanets **S**atellite is the first small (S) class mission in ESA's Cosmic Vision programme. The focus of CHEOPS is to reinvestigate discovered exoplanets by observing planetary transits in front of nearby bright stars. The mission was selected by ESA in October 2012 out of 26 proposed missions. The satellite is scheduled for launch in late 2017 and it will operate 3.5 years. Switzerland as the initiator of the original proposal is designing the mission together with several other ESA member states.
Vienna's *Institut für Astrophysik* (Institute of Astrophysics) contributes to the CHEOPS mission by developing the Instrument Flight Software (IFSW).

4.2.1. Mission Description

A summarised mission description including the scientific objectives and the mission requirements is provided below. The following mission details correspond to the CHEOPS Science Requirements Document (Ehrenreich et al., 2015) if not stated otherwise.
The main task of CHEOPS is to perform high precision photometry of transiting planets, which allows to determine their radii with an uncertainty of about 10%. CHEOPS will focus on super-Earth and Neptune-sized planets (1-20 $\mathrm{M_{Earth}}$) for which the mass has already been determined by ground-based spectroscopic measurements. In fact, CHEOPS will target known exoplanet host stars, which are located anywhere in the sky. Follow-up observations of planets discovered by the ground-based Next-Generation Transits Survey (NGTS) are envisaged in order to improve the precision of measured planet radii. This survey performs photometry up to a precision of 1 mmag for stars with V$<$13 mag. It has been estimated that the NGTS will provide about fifty new targets within the super-Earth to Neptune mass range (1-6 $\mathrm{R_{Earth}}$), until CHEOPS is launched. The observation of such targets allows to study their structure and make implications about planetary formation and evolution. The characterisation of the observed planets includes their approximate composition, as the mass of the planet is known in advance. The science goals of CHEOPS further involve the accurate determination of planetary mass-radius relation and the constitution of optimal targets for spectroscopic studies with future instruments (e.g. with JWST, E-ELT). Furthermore, CHEOPS is aiming to constrain the migration paths of planets during their formation and evolution. Other goals involve the study of atmospheres of planets, such as Hot-Jupiters. In order to achieve these goals photometry must be performed on the stellar photon noise level. For an Earth-sized planet that orbits a G5 dwarf star of 0.9 $\mathrm{R_{Earth}}$ with visual magnitudes 6$\leq$V$\leq$9 mag, the expected transit depth is 100 parts-per-million (ppm). Depending on the revolution time of the planet, the integration time must be adapted in order to reach the required $\mathrm{S/N_{transit}}$. The pointing error during transit measurements shall not exceed

8” at 68% confidence. The operational wavelength band is ranging from 0.4-1.1 µm. The optical configuration follows the Ritchey-Chrétien style with a focal length of F/8 and an effective aperture of 30 cm. The circular field of view of the instrument features a diameter of 0.32°. The satellite’s baseline orbit depicted in Figure 4.8 represents a Sun-synchronous low-Earth orbit at an altitude of 650-800 km. The orbit is close to the terminator and the orbital period is about 101 minutes. A large external baffle is mounted on the satellite in order to reduce implications on photometric measurements, due to Earth’s stray light. It is required that less than 10 ppm stray light is detected in the worst case. Due to the low-Earth orbit, CHEOPS will pass the South Atlantic Anomaly on its orbit (see Section 2.8). In the next Section, I present models that were derived from MOST observations for the description of cosmic ray hit characteristics on a CCD. This has been done to estimate their impact on observations and centroiding performance (see Section 4.2.4.2).

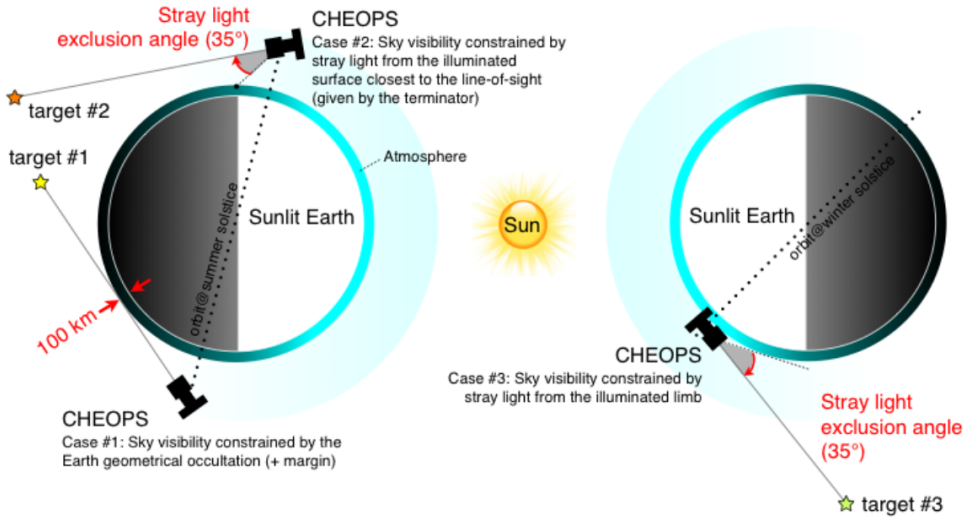

Figure 4.8.: Illustration of the CHEOPS baseline orbit at summer and winter solstice. (Ehrenreich et al., 2015)

4.2.2. Observations in the South Atlantic Anomaly

In order to estimate the impact of glitches on observations performed during a SAA passage I analysed images of HD 189733 which had been acquired in the MOST (Microvariability and Oscillations of Stars) mission. A brief description of the South Atlantic Anomaly (SAA) is given in Section 2.8. The used dataset is considered to be very well-suited for estimating the expected glitch rate for CHEOPS observations during SAA passages, since the orbits of both spacecraft (CHEOPS and MOST) are quite similar, as well as the CCD.

4.2.2.1. MOST - Microvariability and Oscillations of STars

The MOST mission is a space telescope hosted by the Canadian Space Agency (CSA). It has been launched in June 2003 and it represents the first spacecraft dedicated to measure astroseismology of solar type stars, metal-poor subdwarfs and microvariability in Wolf-Rayet winds. Photometry of giant extrasolar planets with short orbital periods is carried out as well. In fact, MOST measures low-degree oscillations with precisions up to micromagnitudes (Walker et al., 2003). In 2011, MOST was able to detect transits of the extrasolar planet 55 Cancri e based on photometric measurements that were carried out in two consecutive weeks.
The satellite's dimensions are $65\times65\times30$ cm^3 and it has as a mass of 54 kg (including 14 kg payload). Therefore, MOST categorised as microsatellite. Attitude control and pointing stabilisation are guaranteed by three-axis reaction wheels and magentotorquers. The pointing accuracy must be better than half an arcminute during the observations. The telescope is built in Rumak-Maksutov (also Rutten Maksutov-Cassegrain or RM) design with an aperture of 150 mm and a primary mirror diameter of 173 mm. The Rumak-Maksutov design is similar to a Cassegrain configuration despite that the secondary mirror is attached to a meniscus corrector. As a consequence, coma and chromatic aberration are reduced. Before entering this optical system the light is initially reflected by a periscope mirror tilted to the Rumak-Maksutov (RM) configuration by 45°. Therefore the telescope is completely shielded from the Sun. The field of view has a diameter slightly bigger than 2° and is exposed by two separate CCDs, one dedicated for science and the other for guidance tasks. A single broadband filter is restricting observations to the wavelength band ranging from 350 - 780 nm. An array of microlenses allows to observe bright targets (V$\leq$10 mag) in Fabry mode, whereas fainter sources are imaged directly on the science CCD at a different spot. Targets within declinations from $-19°$ to +36° can be observed continuously for up to sixty days.
The characteristics of MOST's polar, Sun-synchronous orbit are listed in Table 4.2 and a schematic illustration is shown in Figure 4.9. There are three dedicated ground segments situated in Vancouver, Toronto and Vienna which are used for communication with MOST during its orbit. Stray light from Earth significantly affects the photometric accuracy of MOST, as it is not equipped with a baffle. As defined in Table 4.2, CHEOPS and MOST follow a similar low-Earth orbit. In addition to MOST, CHEOPS features a large external baffle for reducing the stray light implications.

Orbit characteristics		
Altitude of apogee	839.62	km
Altitude of perigee	825.92	km
Semimajor axis	7 210.62	km
Inclination	98.72	deg
Orbital period	6 084.90	s
Local time of ascending node (LTAN)	18:01:06	hh:mm:ss
Continuous Viewing Zone (CVS)	48.00	deg

Table 4.2.: Orbit characteristics of MOST. (Walker et al., 2003)

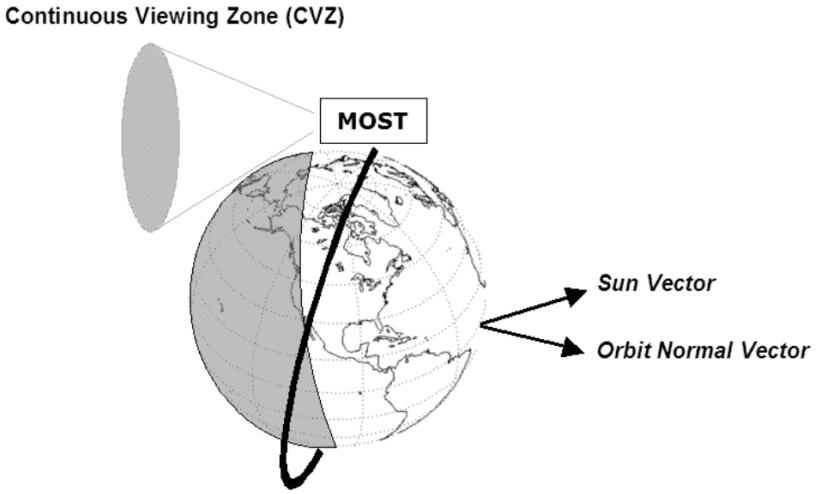

Figure 4.9.: Illustration of the MOST orbit including the continuous viewing zone (CVZ). (Walker et al., 2003)

4.2.2.2. Observations of HD 189733

In 2006 August, MOST observed a transit of the exoplanet HD 189733b in front of its host star which features an apparent magnitude V = 7.67 mag. The host star is classified as K1.5V star. In total, 10 consecutive transits could be observed within 21 days, as the orbital period is about 2.2 days (Miller-Ricci et al., 2008). The main science objective was to find additional planets by analysing variations in the orbital period. So far no additional planets have been confirmed for the planetary system HD 189733. However, the images obtained for the photometric measurements have been reinvestigated in this study for the purpose of understanding glitches in low-Earth orbits. As the tracking CCD of MOST failed due to a particle hit, the science instrument was additionally used for guiding tasks. Therefore, only short exposures of 1.5 s were carried out to avoid significant tracking errors. Furthermore, 14 consecutive images were stacked on-board before their transmission to the ground-stations in order to reach a higher SNR (Croll et al., 2007).

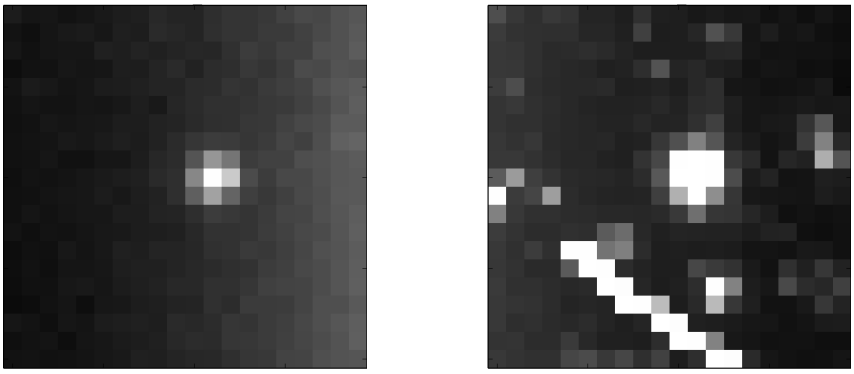

Figure 4.10.: Both images show observations of HD 189733 carried out by the MOST satellite in 2006. Each image is a composition of fourteen short time exposures. A sub-frame of size 20×20 pixels containing the primary target is shown. The left panel represents a typical observation outside the SAA. The right panel depicts an observation carried out inside the SAA containing multiple distinct glitches that are affecting the image quality.

I used these observations for estimating the impact of glitches on observations that are performed with optical sensors in space. Reduction pipelines for cosmic ray hits are run on ground. Therefore, the raw data set could be investigated. The images, each composed of 14 short exposures, are treated as single observations with exposure times of 21 s. A subset consisting of 40 000 images was gathered in the first 14 days. This subset represents the basis for modelling the glitches in the observations. Subsequent records were excluded, due to the increased amount of stray light reflected by Earth during the last days of observation. The investigated dataset included a total number of 147 SAA passages (for further information about the SAA see Section 2.8.1) as a consequence of MOST's short orbital period. The average time for a flight through the SAA's took about 15 minutes or 14.7 % of a complete orbit. On average, 30 images were obtained within this region per orbit, which led to a total number of 4 072 observations inside and 35 928 observations outside the SAA. Figure 4.10 provides an impression for the difference between regular measurements and measurements in the SAA. The effect of stray light implications can be seen clearly in the left panel. The most significant glitch event of the whole sample is illustrated in the bottom left corner of Figure 4.10. This glitch is affecting more than 15 pixels. The entire glitch has a total signal that is about 30 times higher than the total target's signal. Such extreme events occurred very rarely in the whole data set. However, in addition to the large-scaled glitch, Figure 4.10 represents a regular observation inside the SAA, which contains multiple glitches per frame. Another example for a typical observation in the SAA is depicted in Figure 2.9.

A self-composed glitch identifying algorithm was applied to the dataset in order to detect glitch signatures above a certain threshold. Results and statistical statements are summarised in the next section.

4.2.2.3. Results and Statistical Statements

The results listed below have been applied in simulations with *StarSim* to estimate the centroiding performance in the SAA (see Section 4.2.4.2). The full size of MOST's CCD is a 1024×1024 pixel array. In general, only small windows of size 20×20 pixels that contain the science target are transmitted via the downlink to Earth. The pixel size of the CCD is 13×13 μm^2. Therefore, the small sub-frames represent a total detector area of 260×260 μm^2. However, results are provided in units per pixel as CHEOPS features the same pixel size. Furthermore, the measured signal is provided in e^- by applying the gain of 6.1 e^- ADU^{-1} (Walker et al., 2003).

Glitch Rates

Applying the glitch detection algorithm to the dataset of 40 000 observations showed that 409 glitch events occurred during the SAA passage and only 13 events took place outside. By correcting for the time spent in each region, it could be derived that $4.308 \cdot 10^{-8}\ px^{-1}s^{-1}$ *glitches are expected outside the SAA*. The *glitch rate inside the SAA is* $1.196 \cdot 10^{-5}\ px^{-1}s^{-1}$. Therefore, glitch events are about *278 times more likely to occur inside the SAA*. It should be added that no observation performed outside the SAA contained multiple glitches. It can be seen in Figure 2.8 that the SAA spans over a wide region with various magnetic field strengths. In general, it can be stated that a lower magnetic field strength implies a higher glitch rate. Therefore, I also investigated the glitch rate for a worst-case scenario, by analysing the observations performed in regions with extraordinarily low magnetic flux densities ($<$ 19 000 nT). Thus, within such regions the *rate of glitches is expected to be* $(8.7 \pm 3.9) \cdot 10^{-4}\ px^{-1}s^{-1}$. Results are provided in the Appendix in Table A.4 and A.5.

Affected Pixels per Glitch

Out of the 422 events that were identified by the glitch detection algorithm, 189 unambiguous, non-overlapping signatures were selected for studying signature characteristics such as the electron deposition distribution. It turned out that only 23.8 $\%$ of all glitches affected single pixels. A higher amount was expected, since multi-pixel signatures are caused by particle hits with trajectories almost parallel to the CCD surface (see Section 2.8) and the particles hitting the CCD are assumed to be distributed uniformly in space with respect to the detector surface. However, an explanation for the *ratio of single- to multi-pixel signatures being almost 1:3* can be given based on the CCD characteristics. Particles elongated almost normal to the CCD follow a much smaller path through the CCD substrate compared to particles with flat impact angles. Hence, the probability for electron deposition is much higher for flat angles.

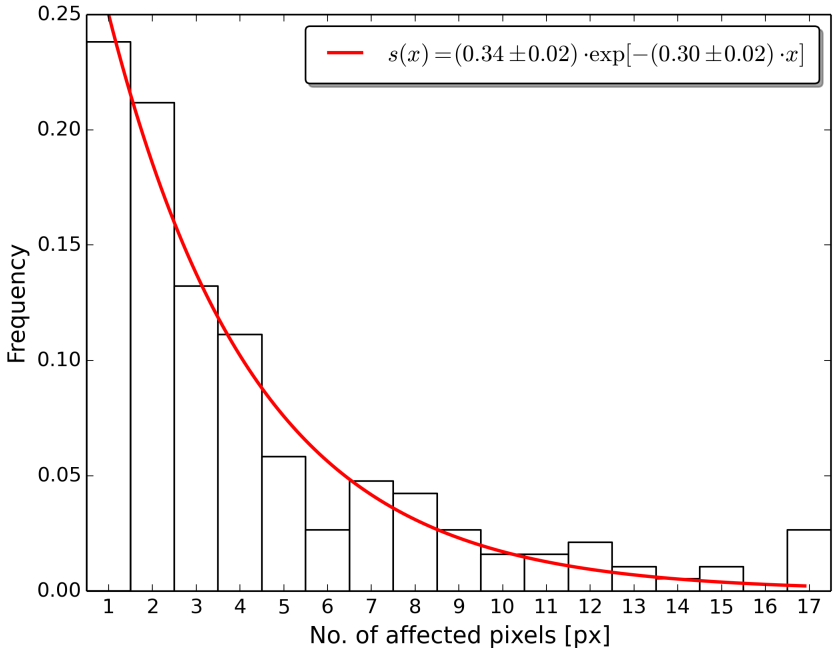

Figure 4.11.: Distribution of number of affected pixels per glitch for the sample of 189 glitches provided in Table A.4. The data were modelled by an exponential distribution (red line).

Figure 4.11 shows a normalised distribution for the number of affected pixels per glitch. As it has been stated before, multi-pixel characteristics are more likely. The amount of affected pixels per glitch was modelled with the normalised exponential distribution presented in Equation (4.6).

$$S(p) = \begin{cases} 0 & (p < 1,\ p > 20) \\ 0.4025 \cdot \exp[-0.2984 \cdot p] & (1 \leq p \leq 20) \end{cases} \tag{4.6}$$

This equation is derived from the data presented in Figure 4.11 and represents a probability density function (PDF). Parameter p is the number of pixels affected per glitch. The shape is illustrated in the left panel of Figure 4.13. Glitches that affected more than 20 pixels were not observed.

Electron Deposition per Glitch

The measured electron deposition distribution is illustrated in Figure 4.12. The total signal per glitch was calculated and the bin-width was set to 3 700 e^-. It is clearly visible that glitches with low intensity ($< 18\,000$ e^-) are more likely to occur and they follow an exponential distribution. Additionally, the distribution features a distinct right tail, that represents the most intense glitches, which occur less frequently. This part of the distribution should not be considered

complete, due to the low number of samples. Nevertheless, it still provides valuable information about the relative frequency of high energy events. A conversion of electron deposition to the particle energy was not performed, as the electron deposition also depends on impact angle, particle type and hardware characteristics. Due to this degeneracy, a direct conversion is not feasible. Particularly, a flat impact angle can result in a much higher amount of electron deposition, therefore the right wing of the distribution is composed of both, low and high-energetic particles. However, it is expected that the high energy particles dominate this part of the distribution. The data were modelled using two distinct exponential distributions resulting in the PDF given in Equation (4.7).

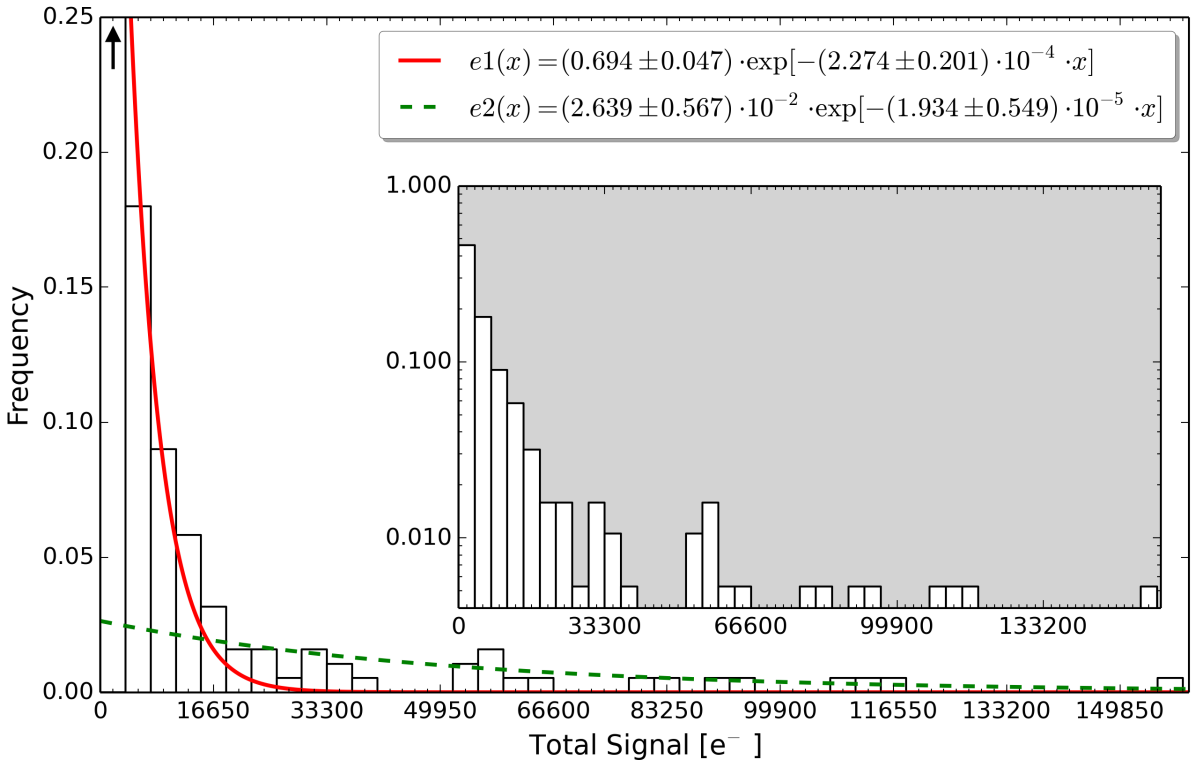

Figure 4.12.: Distribution of electron deposition per glitch for the sample of 189 glitches provided in Table A.4. The embedded plot shows the same distribution with logarithmic y-axis. The data were modelled by using a combination of the two exponential distributions *e1* and *e2*.

$$E(x) = \begin{cases} 0 & (x < 150,\ x > 200000) \\ 1.806 \cdot 10^{-4} \cdot \exp[-2.274 \cdot 10^{-4} \cdot x] & (150 \leq x \leq 15714) \\ 6.868 \cdot 10^{-6} \cdot \exp[-1.934 \cdot 10^{-5} \cdot x] & (15714 \leq x \leq 200000) \end{cases} \tag{4.7}$$

Parameter x represents the total signal (in all pixels) per glitch. Events with less than 150 e^- are excluded, because they were not identified in the dataset, as background and noise are dominat-

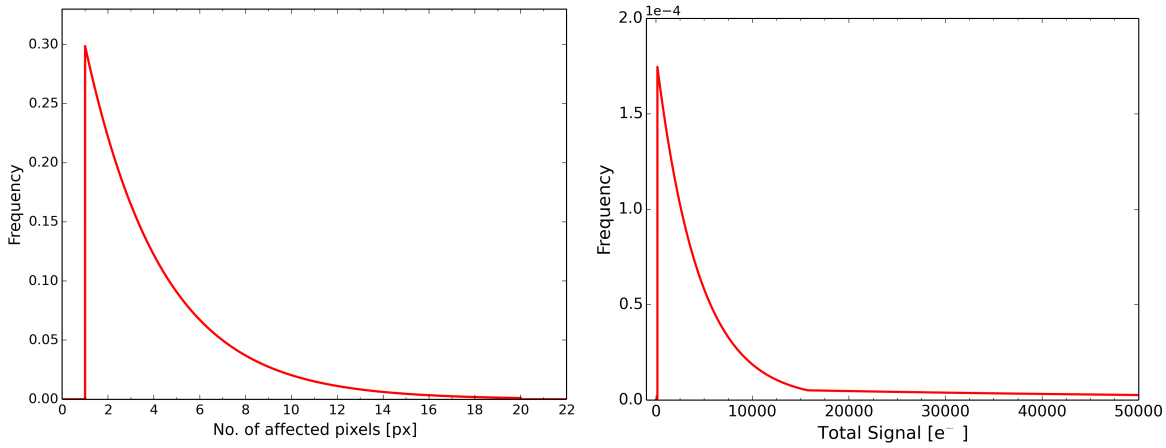

Figure 4.13.: Probability density functions that describe the number of affected pixels $S(p)$ (left) and the total electron deposition $E(x)$ (right) per glitch event. These models were derived by the distributions illustrated in Figure 4.11 and 4.12.

ing such low signals. Furthermore, glitches with intensities greater than 200 000 e^- are excluded as well, due to limitations by the CCD's capacity. The shape of this model is illustrated in the right panel of Figure 4.13.

The two distinct distributions in Figure 4.12 could be caused by different particle sources, where low-energy events have solar origin and high-energy events are related to galactic and extra-galactic sources.

4.2.3. Point Spread Function

The point spread function illustrated in Figure 4.14 is caused by an intentional defocused optical configuration. Therefore, the secondary maxima are significantly higher and cannot be neglected. The purpose of defocusing is to distribute the signal across a large number of pixels. This improves the statistical significance of observations, by lowering the impact of individual pixel characteristics. As a consequence, the PSF has to be approximated by custom models, instead of two-dimensional Gaussian functions (as presented in Section 4.1.3). The illustrated model was applied in simulations regarding the selection of centroiding algorithms. In particular, it was applied for image generation with *StarSim*. Furthermore, it represents the weighting function that was applied in the IWCoG.

For the Levenberg-Marquardt algorithm (LMA) a mathematical model of the PSF is required as input. Figure 4.15 depicts a one-dimensional cross-section through the centre of the radial symmetric PSF (see right panel of Figure 4.14). Equation (4.8) represents a mathematical description of this PSF. It was constructed by combining multiple linear fits in the one-dimensional cross-section. In order to obtain a two-dimensional model the fit was extended by considering radial symmetry.

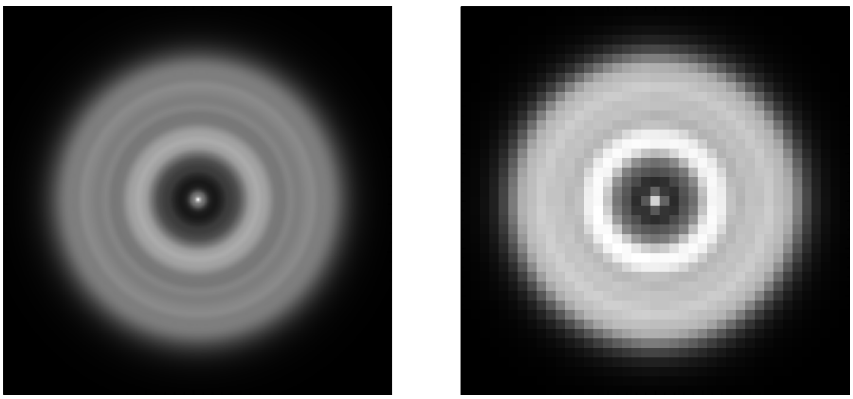

Figure 4.14.: *Left*: High resolution model of the point spread function of CHEOPS. *Right*: Downscaled model in final image resolution. Model data were obtained in personal communication (Roland Ottensamer, September, 2014)

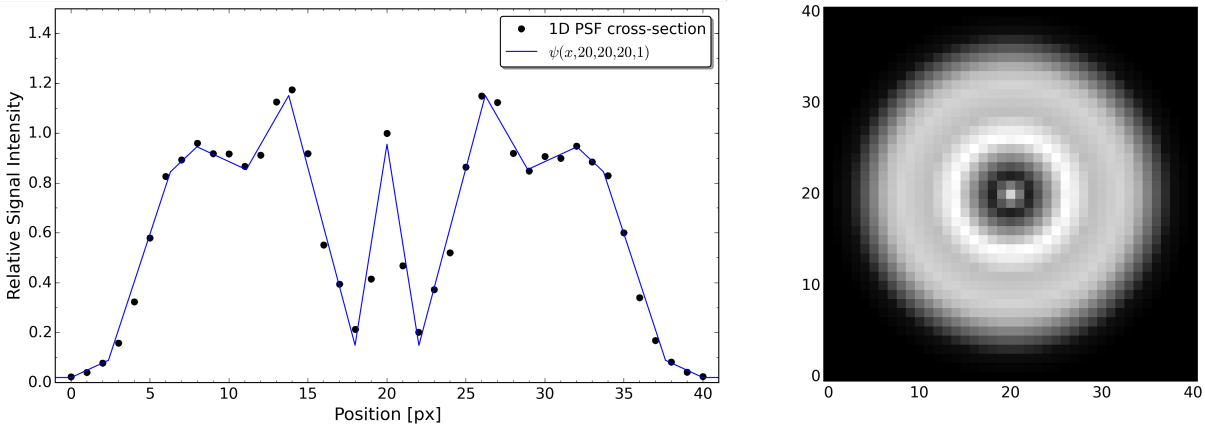

Figure 4.15.: *Left*: Cross-section through the downscaled PSF model (black data points) presented in Figure 4.14. Blue lines indicate an approximation with combined linear fits. *Right*: Reconstruction of the PSF with Equation (4.8) for $(x_0, y_0) = (20, 20)$.

The following Equation was derived by fitting the provided PSF data.

$$\psi(x, y, x_0, y_0, A) = \cdot \begin{cases} A \cdot (-0.39943 \cdot \Delta + 0.95605) & (0 \leq \Delta < 2.02) \\ A \cdot (+0.23869 \cdot \Delta - 0.33329) & (2.02 \leq \Delta < 6.22) \\ A \cdot (-0.11061 \cdot \Delta + 1.83983) & (6.22 \leq \Delta < 8.91) \\ A \cdot (+0.02927 \cdot \Delta + 0.59358) & (8.91 \leq \Delta < 12.03) \\ A \cdot (-0.05969 \cdot \Delta + 1.66382) & (12.03 \leq \Delta < 13.71) \\ A \cdot (-0.19274 \cdot \Delta + 3.48786) & (13.71 \leq \Delta < 17.63) \\ A \cdot (-0.02924 \cdot \Delta + 0.60465) & (17.63 \leq \Delta \leq 20) \\ A \cdot 0.0198 & (\Delta < 0,\ \Delta > 20) \end{cases} \tag{4.8}$$

where

$$\Delta = \sqrt{(x - x_0)^2 + (y - y_0)^2} \tag{4.9}$$

The parameter Δ represents the absolute 2D-distance to the central pixel (x_0, y_0), which is used for the centroid estimation. The function is normalised in a way that the central pixel $(x_0, y_0) = 1$, if $A = 1$. By changing A, the function can be adjusted to the stellar signal. Equation (4.8) was applied to create the image in Figure 4.15, which illustrates a reconstruction of the PSF depicted in Figure 4.14.
The Cramer-Rao bound from Section 3.1 represents a lower limit for the centroid estimation error. In its derivation, it is assumed that the PSF is represented by a two-dimensional radial symmetric Gaussian, which is not the case in the CHEOPS mission. However, I computed the Cramer-Rao bound for a Gaussian PSF with the assumption that 99.73% ($3\sigma_{\mathrm{PSF}}$) of the incident stellar flux are located within a radius of 19 pixels around the spot centre. This assumption is based on the size of the PSF-model that is depicted in Figure 4.14 and leads to a standard deviation of $\sigma_{\mathrm{PSF}} = 6.33$ px. The Cramer-Rao bound that corresponds to such a Gaussian is applied in Section 4.2.4 to assess the centroid quality.

4.2.4. Selection of Centroiding Algorithms for Fine-Guiding

In contradiction to EChO, the fine guiding task is carried out directly on science frames. The star trackers initiate the pick-up of the target by the Sensor Electronics Subsystem (SES) of the spacecraft, similar to the hand-over task introduced in Section 4.1.4.1. However, they are limited in precision as thermoelastic deformations of the telescope cannot be compensated by the star trackers. Once the target is imaged in the science frame it will be identified via star identification algorithms (Padgett & Kreutz-Delgado, 1997; Liebe, 1992) as multiple stars are expected to appear in the image, in particular if the observed stellar field is crowded. Eventually, centroids shall be frequently computed and passed to the AOCS in order to maintain the pointing stability. In fact, the pointing error shall not exceed 8" rms (Beck & Malvasio, 2015) during a full orbit. Observations throughout the entire orbit are not possible as the payload instrument's line of sight is occulted by Earth. During such payload occultations no fine-guiding information is available for the AOCS. In the other case, the target is in the field of view of the payload telescope. This pointing-mode is referred to as 'payload in the loop', which demands centroid estimation errors below 1" rms for fine guidance. This shall represent the main requirement for the following selection of the centroiding methods.
According to the CHEOPS Instrument Requirements Specification (Beck & Malvasio, 2015), the nominal detector type for the SES is the back illuminated CCD sensor model CCD47-20 AIMO (advanced inverted mode operation) with a mid-band coating. At sensor level, a single pixel corresponds to approximately one arcsecond of the instruments field of view. The mission parameters relevant for fine guiding as well as the sensor characteristics are summarised in Table 4.3. The centroiding algorithms were tested with respect to these requirements. The pixel

scale of the sensor is equal to MOST's CCD. Therefore, the characteristics of cosmic ray hits such as rate, size and intensity, were simulated with the models presented in Section 4.2.2.3.
The mission requirements demand a low contribution of stray light, which is achieved by applying baffles. Simulations regarding Earth's stray light are further discussed by Kuntzer et al. (2014). In the simulations presented below, I applied a combined background that contains zodiacal light and stray light. As centroids are computed on science frames, the centroid update frequency depends on the length of exposures. In the next sections, I present centroiding performance analyses for short (1 s) and long (60 s) exposure times, respectively. In particular for long time exposures, the spacecraft's spatial jitter may lower the centroiding performance as the shapes of the stars appear blurred on the science frames. It was assumed that a high-frequency jitter is affecting the spacecraft's yaw-axis and pitch-axis. This jitter component is below 1" rms for short exposures and does not exceed 4" rms for long exposures. The rotation of the spacecraft induces a similar blurring effect on science frames in particular at the outer regions of the sensor. However, it is assumed that the target's position on the detector is close to the rotation axis. Hence, the impact of rotation as well as the roll-axis jitter is considered to be negligible in terms of centroiding. Furthermore, it was assumed that the low-frequency component of the jitter features a period of time that is similar to the orbital period. Therefore, this component does not affect observations with exposure times below sixty seconds. Hot pixels were neglected in simulations for the centroiding performance as they can be reduced via periodically updated hot pixel maps and their effect is supposed to be lower than the effect of glitches. The pixel-to-pixel flat field has been modelled with a normal distribution that has a standard deviation of 1%.
The four centroiding algorithms, IWC, IWCoG, correlation-interpolation and LMA are possible candidates for the CHEOPS fine-guiding task as they are invariant to initial position offsets (see Table 3.1). Such invariance is important, as the centroid estimations of the implemented star identification algorithm will differ from the true centroid. It was assumed that these algorithms will provide CEEs below 4 px rms. Furthermore, the spacecraft's jitter may lead to displaced initial centroid estimations as well. Imprecise position estimations always go ahead with a displacement of the RoI, which means that the star is not centred in the window. This is mainly an issue for the IWC algorithm, which can be counterbalanced by setting appropriate thresholds.
The performance of the preselected centroiding methods was tested under different conditions. In particular, large samples of observations were created in Monte-Carlo simulations. In the next sections, results are presented for simulations of different exposure times, observations of crowded fields as well as observations inside the SAA.

Parameter	Value	Source
CCD Model	E2V CCD47-20 AIMO	Beck & Malvasio (2015)
Full Frame	1024×1024 px[b]	Beck & Malvasio (2015)
Science Frame	200×200 px	Beck & Malvasio (2015)
Exposure Times	$1-60$ s	Beck & Malvasio (2015)
Operating Temperature	233 K	Beck & Malvasio (2015)
ADC	14 bit	Beck & Malvasio (2015)
FWC CCD	10^5 $e^-px^{-1}s^{-1}$	Sensor Datasheet[a]
FWC ADC	$6\cdot10^4$ $e^-px^{-1}s^{-1}$	Sensor Datasheet[a]
Bias Level	100 ADU	Assumption
Dark Current	0.042 $e^-px^{-1}s^{-1}$	Fortier (2015)
Dark Current Requirement	$<$0.08 $e^-px^{-1}s^{-1}$	Beck & Malvasio (2015)
Readout Noise	10 e^-px^{-1}	Beck & Malvasio (2015)
Flat Field Variation	1% px^{-1}	Assumption
Flat Field Knowledge	0.1% px^{-1}	Ehrenreich et al. (2015)
Zodiacal Background	8.35 photons$^-px^{-1}s^{-1}$	Fortier (2015)
Stray Light	2 photons $px^{-1}s^{-1}$	Ehrenreich et al. (2015)
Optical bandpass	400-1100 nm	Beck & Malvasio (2015)
Pixel Size	13×13 μm^2	Beck & Malvasio (2015)
Pixel Scale	$\sim$1” px^{-1}	Beck & Malvasio (2015)
Global Throughput (QE)	65%	Beck & Malvasio (2015)
Jitter	$<$8” rms	Ehrenreich et al. (2015)
Fine Guiding Error	$<$1” rms	Ehrenreich et al. (2015)
Glitch Rate Nominal	<0.002 s^{-1} $(200\times200$ px$)^{-1}$	See Section 4.2.2.3
Glitch Rate SAA	<0.5 s^{-1} $(200\times200$ px$)^{-1}$	See Section 4.2.2.3
Glitch Rate SAA Peak	34.8 ± 15.6 s^{-1} $(200\times200$ px$)^{-1}$	See Section 4.2.2.3
Faintest Target (SciReq 1.2)	K-Star, V=12 mag, 63 364 photons s^{-1}	Ehrenreich et al. (2015)[c]
Faintest Target (goal)	K-Star, V=13 mag, 25 226 photons s^{-1}	Ehrenreich et al. (2015)[c]

Table 4.3.: CHEOPS mission parameters and sensor characteristics of the chip e2v CCD47-20 AIMO.

[a] Sensor Datasheet: e2v technologies datasheet: A1A-100041 Issue 6, March 2006
[b] exclusive the dark current reference fields and overscan elements which are situated at the outer sensor border
[c] The photon counts are already reduced by the global throughput of the telescope. They were obtained in personal communication (Roland Ottensamer, August, 2014).

4.2.4.1. Nominal Observation Mode

Monte-Carlo simulations were performed in order to assess the centroiding performance with respect to the requirements listed in Table 4.3. The nominal observations were simulated with a glitch rate of 0.002 glitches per second and per science frame. Figure 4.16 depicts the centroiding performance in the nominal observation mode for one second exposures. It can be seen that the centre of gravity algorithms (IWC and IWCoG) do not converge against the Cramer-Rao bound. This can be explained by the newly introduced jitter as centre of gravity based algorithms perform worse if the image is blurred. IWCoG and LMA showed similar results in all previous analyses. This is not the case in this analysis, because the fit-model for the LMA differs from the pixel weighting used by the IWCoG. Best performance was clearly achieved by the correlation-interpolation method, as it reached the Cramer-Rao bound. However, all four methods fulfilled the fine-guiding requirement which demands pointing errors below 1" rms. As expected, the median of the CEEs is slightly lower than the mean CEE for fainter stars. The mean CEE of IWC is extraordinarily high at a particular photon count of 51 000 photons per second. This statistically significant deviation is caused by the occurrence of a type I glitch in a single image, that is illustrated in Figure 4.17. The other algorithms are not affected by this, as they are invariant to type I glitches (see Section 3.5.4).
Figure 4.18 represents the results for the same simulation carried out with an exposure time of sixty seconds. Due to the longer exposure time, the glitch rate increased to 0.12 glitches per second and per science frame. This led to the frequently occurring peaks in the mean CEE of the IWC. The performance of the other algorithms was not affected by the increased glitch rate, as the expected rate of type II glitches is lower. The expected rate of type I glitches inside the RoI is 0.0084 glitches per image and the expected rate of type II glitches inside the PSF (16 pixels radius) is only 0.0024 per image. The mean CEEs of the algorithms are very low and they are saturated at high SNRs. IWCoG's saturation value was slightly higher than others, as the blurred spots (caused by the jitter) differed from the non-blurred weighting function. The outliers of IWC may be identified during observations by comparing current results with previous centroids. By doing so, all four presented algorithms are capable of carrying out the fine-guiding task in the nominal observation mode.

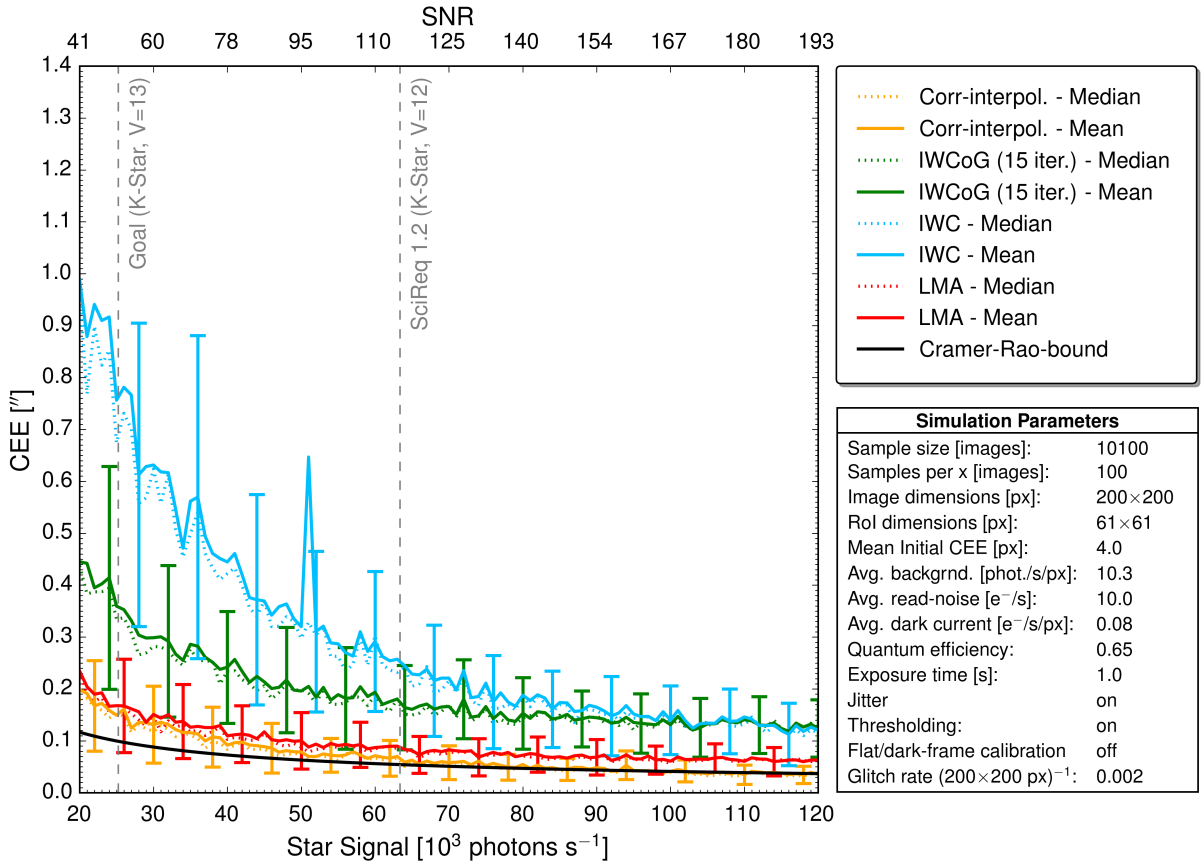

Figure 4.16.: The centroiding performance for nominal CHEOPS observations with a constant exposure time of one second. *StarSim* was used to create a sample of 10 100 images. An illustration of the images is depicted in Figure 4.17. In each image the target was placed at a random position inside a 50×50 pixel area that is centred in the science frame. The spacecraft's jitter was considered to be below 1" rms. As described in Section 4.2.3, the PSF was approximated with a Gaussian to compute the Cramer-Rao bound. The error bars are symmetric as they represent the standard deviation of the mean true CEE. The vertical dashed lines indicates the photon counts of the faintest targets as listed in Table 4.3.

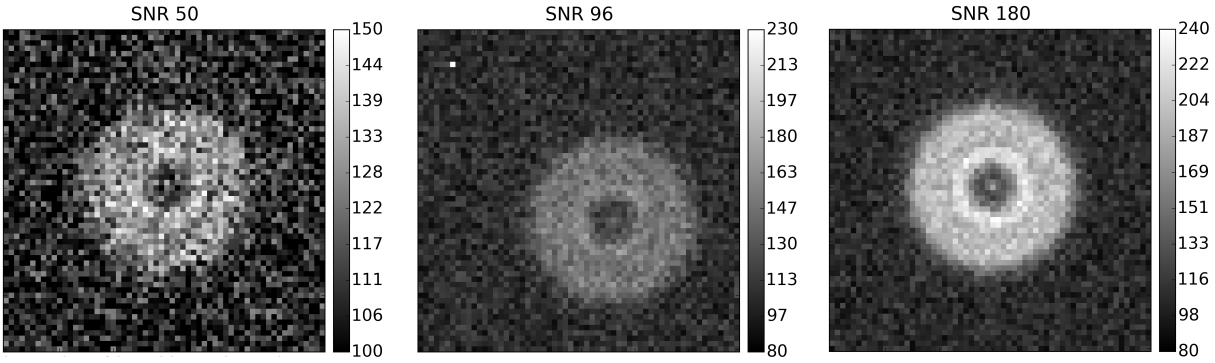

Figure 4.17.: Samples of RoIs for one second exposures for SNRs 50, 96 and 180 from left to right. Colour bar values are in unit electrons. The central image contains a type I glitch in the upper left corner. The glitch signal is about 1 100 e⁻. Images appear slightly blurred due to the simulated jitter.

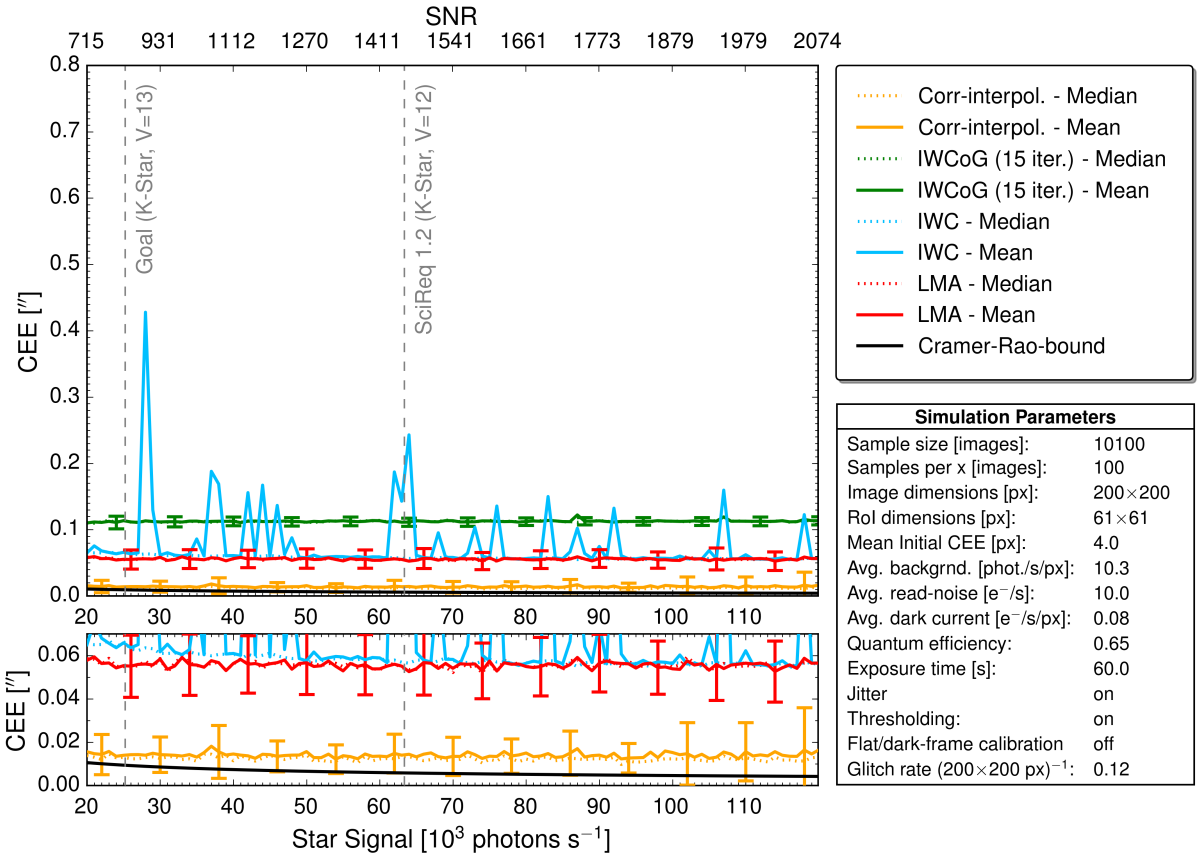

Figure 4.18.: The centroiding performance for nominal CHEOPS observations with a constant exposure time of sixty seconds. *StarSim* was used to create a sample of 10 100 images. The lower panel represents a zoom on the Cramer-Rao bound. In each image the target was placed at a random position inside a 50×50 pixel area that is centred in the science frame. The spacecraft's jitter was considered to be below 4" rms. As described in Section 4.2.3, the PSF was approximated with a Gaussian to compute the Cramer-Rao bound. The error bars are symmetric as they represent the standard deviation of the mean true CEE. The vertical dashed lines indicates the photon counts of the faintest targets as listed in Table 4.3.

4.2.4.2. Observations Inside the South Atlantic Anomaly

During the low-Earth orbit of CHEOPS the satellite will pass the SAA. In order to evaluate the impact on fine-guiding, I simulated a worst-case scenario where the mean glitch rate is represented by the peak rate in the SAA (see Table 4.3). For a five second exposure, a rate of 174 glitches per science frame is expected. Therefore, the expected number of type I glitches inside the RoI is 12.7 and the expected rate of type II glitches inside the PSF (16 pixel radius) is about 3.5 glitches per observation. The centroiding performance for such a case is depicted in 4.19. The glitch signals are not contributing to the noise that is illustrated by the SNRs. The median CEEs reveal that none of the four preselected algorithms was capable of fulfilling the fine-guiding requirement. In fact, the centroid estimation errors are more than one magnitude higher compared to the nominal mode. The scale of the error bars that represent the standard deviation was too large to be included in the graph. Therefore, the success rate of the algorithms was analysed by comparing the results with the fine-guiding requirement. As expected, the

IWC showed a constantly high mean CEE due to the large number of glitches. The CEE was below 1” only in 2.3% of all simulated observations. For the other methods, this occurred more frequently, but in less than 50% of all cases. The exact numbers are listed in Table 4.4.

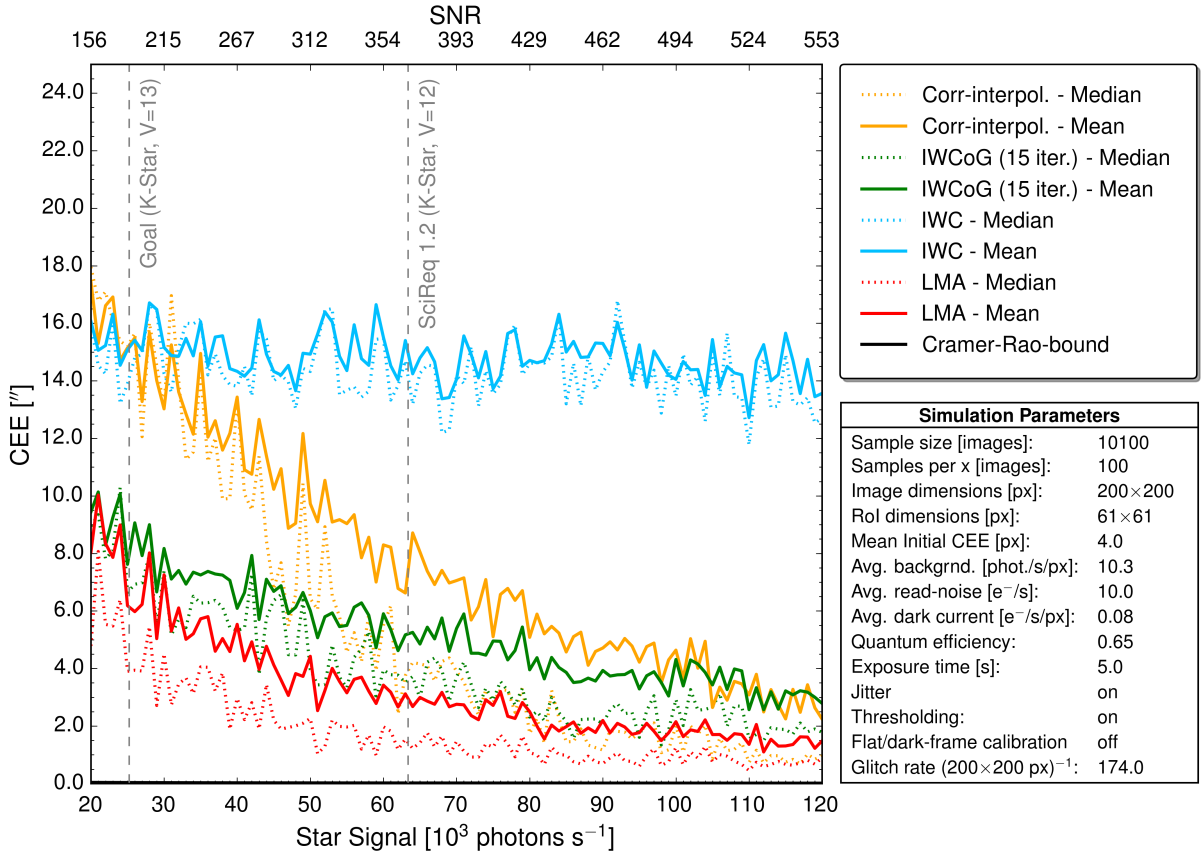

Figure 4.19.: The centroiding performance for CHEOPS observations in the SAA with a constant exposure time of five seconds. This is a worst-case simulation where the peak glitch rate in the SAA was considered as mean glitch rate. An example of a RoI simulated with this glitch rate is depicted in the left panel of Figure 4.20. *StarSim* was used to create a sample of 10 100 images. In each image the target was placed at a random position inside a 50×50 pixel area that is centred in the science frame. As described in Section 4.2.3, the PSF was approximated with a Gaussian to compute the Cramer-Rao bound. The vertical dashed lines indicates the photon counts of the faintest targets as listed in Table 4.3. Error description is provided in the text.

Algorithm	no filter CEE < 1”	median filter CEE < 1”
IWC [%]	2.3	51.3
IWCoG [%]	27.4	93.7
LMA [%]	43.1	97.0
Corr-interpol. [%]	30.5	96.2

Table 4.4.: Illustrates the success rate of the centroiding algorithms during SAA observations with respect to the fine-guiding requirement. ’No filter’ corresponds to the simulation illustrated in Figure 4.19 and ’median filter’ corresponds to the results depicted in Figure 4.20.

In order to lower the CEE, the glitches inside the RoI may be reduced with image processing techniques. Therefore, I applied a 3×3 median filter to the RoI before the execution of the centroiding algorithms. By doing so, each pixel value is replaced by the median computed out of the pixel itself and its eight adjacent pixels. The result of the application of such a median filter is illustrated in Figure 4.20. It depicts that most of the glitches could be removed except for the large-scaled structures, which occur less frequently in the derived models (see Figure 4.11). Figure 4.21 shows the performance of the centroiding methods for the exact same sample of 10 100 images that are were analysed in Figure 4.19, but after the application of a median filter. It can be seen that the median CEE still differs from the mean, due to of a few outliers with large CEEs. These outliers correspond to frames where the glitches could not be removed entirely. However, the median values of LMA, correlation-interpolation and IWCoG are all far below the 1" limit. For these three methods, the computed CEE did not exceed the limit in more than 93% of the simulated images (see Table 4.4). For IWC, this is only the case in about 50% of the images. Therefore, IWC is ruled out for fine-guiding during a passage of the SAA. As the CEE may exceed the limit in a very few cases, I suggest to estimate the centroid quality on-board by tracking deviations with previously computed centroids.

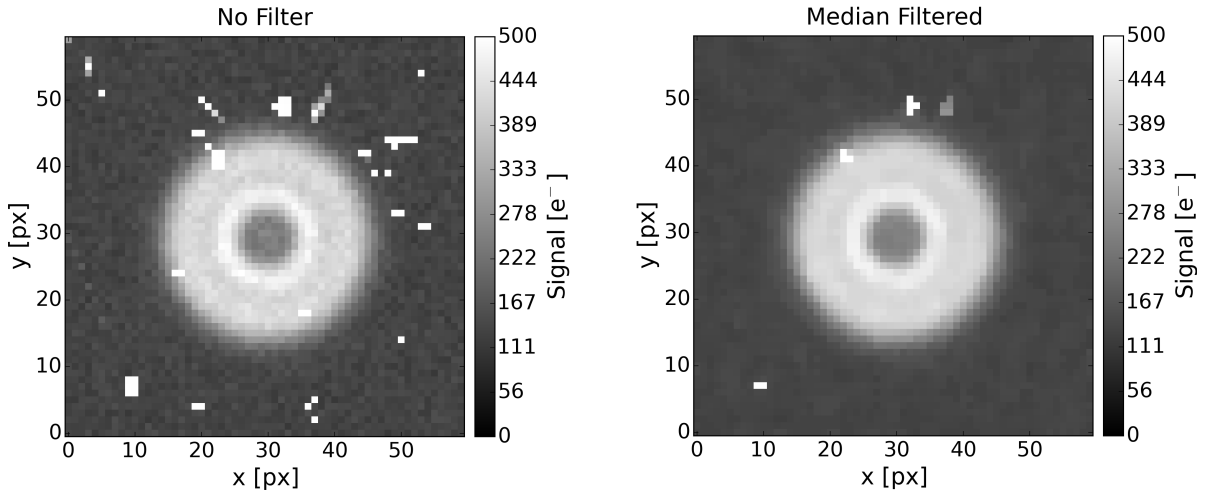

Figure 4.20.: *Left:* Sample RoI for an observation in the SAA with a mean glitch rate of 174 glitches per 200×200 px. *Right:* The same image after a 3×3 median filter was applied.

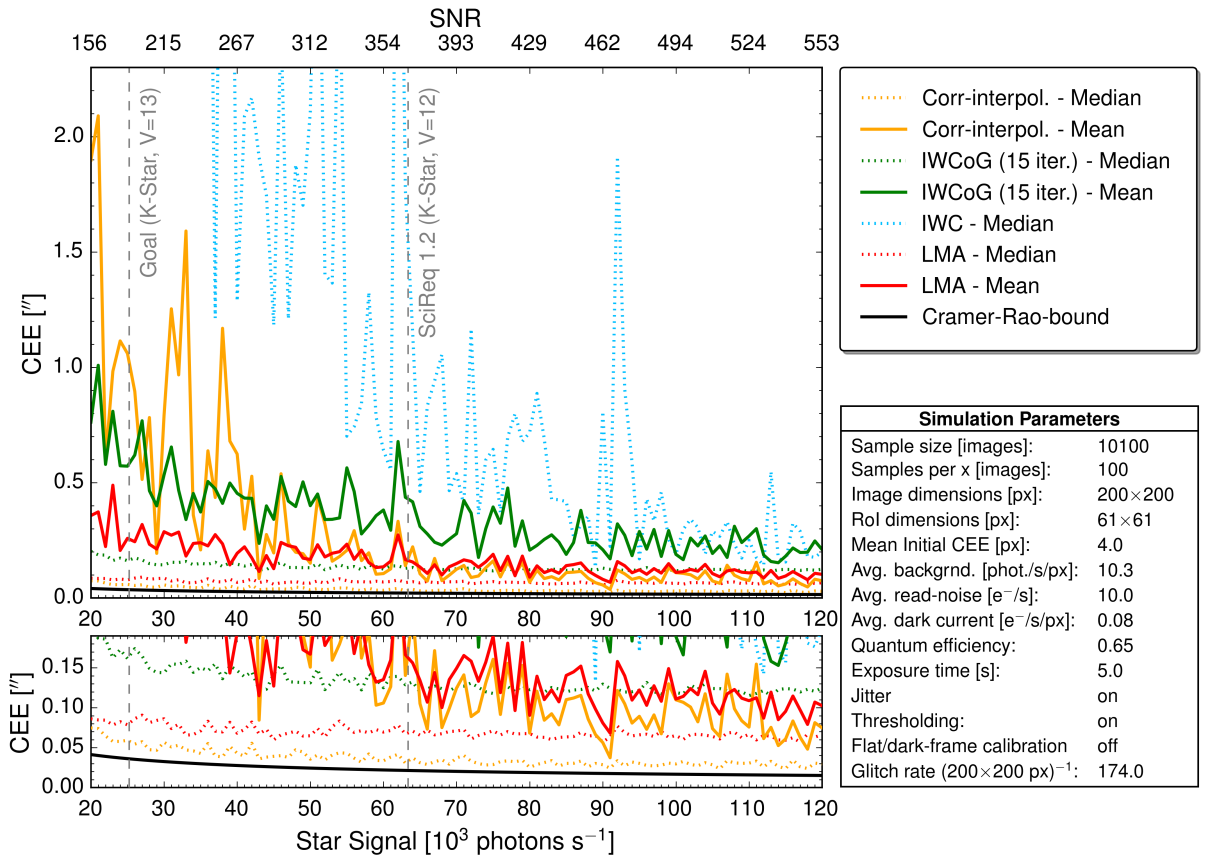

Figure 4.21.: The same simulation as shown in Figure 4.19 with the only difference that a 3×3 median filter was applied to the RoI before the centroids were computed. A sample of such median filtered RoI is depicted in the right panel of Figure 4.20. The mean CEE of IWC was above 4" during the entire simulation and therefore it is not included in this plot. The lower panel represents a zoom on the Cramer-Rao bound.

4.2.4.3. Observations of Crowded Fields

In order to find out which centroiding methods are capable of providing useful results if crowded fields are observed, I simulated an observation of Omega Centauri. The simulated full frame is depicted in Figure 4.23 with the embedded science frame. The extracted RoI is illustrated in Figure 4.22 including the computed centroid estimations. The target star has a brightness of V=11.6 mag and it is partly overlapped by another star of equal brightness. This represents an unusual observation that was selected to test the centroiding methods under extreme conditions. As expected, both centre of gravity algorithms (IWC and IWCoG) show displaced centroids in the bright region between the two stars. In fact, IWC always provides displaced centroids if a second star is observed inside the RoI, even if the PSFs do not overlap. One way to improve the behaviour of IWC in such cases is to entirely remove the additional star by thresholding. However, this is only possible if the target is brighter than the other star. For IWCoG, the target brightness as well as the degree of overlap are decisive for precise centroid estimations. In this sample the degree of overlap was already too high for IWCoG. Best results were achieved by the correlation-interpolation method and the LMA. For such observations, the correlation-interpolation method also requires an initial centroid estimation, in order to select the peak of the target in the correlation function. Both methods also work if a faint target is slightly overlapped by a brighter star. Although their centroids depicted in Figure 4.22 are mispositioned by about 3”, fine-guiding may still be possible with an accuracy below 1”, as the displacement is constant in time if the rotation is negligible. However, such observations of extensively overlapping PSFs are assumed to be not carried out by CHEOPS.

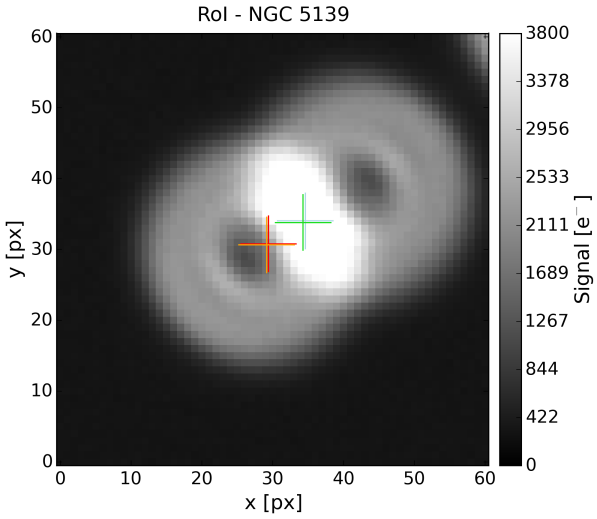

Figure 4.22.: The region of interest (RoI) that was cut out of the science frame (white rectangle in Figure 4.23). Two stars of equal brightness (V = 11.6 mag) are observed, where the target star is situated in the centre of the RoI. An initial CEE of 4” was assumed. The crosses illustrate results of centroiding algorithms, where the results of the centre of gravity algorithms (green crosses) overlap and the centroids provided by LMA (red) and correlation-interpolation (orange) overlap as well.

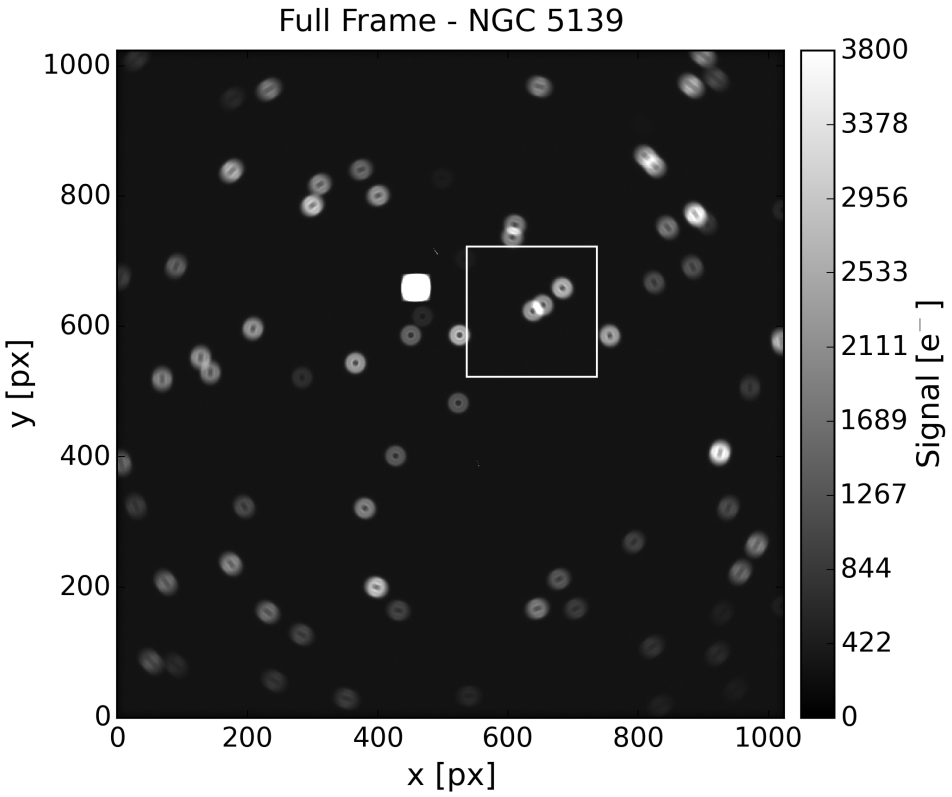

Figure 4.23.: Simulated full frame of the globular cluster Omega Centauri (NGC 5139) with an exposure time of five seconds. The white rectangle determines the 200×200 px science frame that includes the target. The image corresponds to a field of view of about 17'. Stars in outer regions appear blurred as the rotation of the spacecraft was simulated with 0.059° s^{-1}. The star positions were obtained in personal communication (Philipp Löschl, December, 2015). They originate from a SIMBAD (Strasbourg Astronomical Data Center, 2015) query for: RA = 201.696° and DEC = -47.479°.

4.2.5. Conclusion

The CHEOPS pointing stability demands fine-guiding errors below 1" rms. The four algorithms IWC, IWCoG, correlation-interpolation and LMA were tested under various conditions with respect to this requirement. It turned out, that the jitter of the spacecraft has no significant impact on centroiding performance. Only for the centre of gravity algorithms, the saturation of the CEE occurred on slightly increased values compared to the non-jittered case. All four algorithms are applicable for nominal observations, but median filtering is suggested for IWC to avoid extreme CEEs in longer exposures. Such filtering is indispensable for all methods during a flight through the SAA, as glitch rates may be increased by a factor of 278. Models for glitch intensities, sizes and rates were derived from MOST observations. All algorithms except IWC showed a success rate of at least 93% in simulated worst-case observations in the SAA and it is suggested that exposure times are rather low to avoid a large number of glitches in single frames. In observations of crowded fields multiple stars may be imaged in the RoI. In such cases IWC is not applicable if the additional stars cannot be entirely removed by thresholding. For the extreme case of overlapping PSFs, LMA and correlation-interpolation performed best.

However, such observations are currently not envisaged as the photometric errors may be too high. Eventually, the algorithms can be sorted in descending order by their performance the following way: correlation-interpolation, LMA, IWCoG and IWC. Some of the algorithms may be too expensive in terms of computation time as fast processing is required on-board in a closed loop. Here, I provide the mean computation times normalised to IWC as it is the fastest of all four methods. The processing time for fifteen iterations of IWCoG is about 50 t_{IWC}. However, this amount of iterations was only required to reach error saturation in order to evaluate the best performance. The fine-guiding requirement is still fulfilled if only 7 iterations are carried out. The computation of correlation-interpolation was about 63 t_{IWC} and it may only be lowered by further decreasing the size of the RoI. The computation of LMA took on average 375 t_{IWC}, because the fit-model was rather complex. The processing time may be lowered by reducing the accuracy in the PSF reconstruction.

5. Conclusion

The focus of this thesis is on the investigation of centroiding techniques that are applicable for the fine guidance of space telescopes. Any of the presented methods may be applied in ground-based observations. In addition to its application in fine guiding, centroiding is also an important step for star identification algorithms (Liebe, 1992; Padgett & Kreutz-Delgado, 1997). For such methods, centroiding is initially carried out to determine the star positions on the sensor. In this case low-precision centroid estimations on pixel scale are usually sufficient. Therefore, the standard centre of gravity algorithm (see Section 3.2) suits the needs of most star identification techniques. However, these are beyond the scope of this thesis.

A major part of this thesis was the development of *StarSim*. This tool, which initially has been designed for testing centroiding algorithms, represents a sophisticated data simulator that may also be used to simulate telescopic observations of stars for other purposes, such as for photometric studies or for the analysis of star identification algorithms. The code of *StarSim* can be provided on request and is part of the CHEOPS open-source software.
The models that were applied to describe glitches on optical sensors which are associated to cosmic ray hits, may be extended by measurements from other satellites. In fact, the current models may overestimate the glitch rates at higher altitudes as the strength of Earth's magnetosphere declines. Furthermore, the results may also differ if observations are performed with a different sensor model.

The analysis of the centroiding methods was targeted on their reliability as well as on the processing times, which are essential characteristics in most space missions. However, the implementations that are provided in the Appendix will be adapted and used for CHEOPS data processing unit. The next steps for the CHEOPS mission include the implementation of the selected centroiding methods on the prototype flight software. Subsequently, centroiding will be tested in a closed loop, where a simulation of the attitude control of the satellite is embedded as well. Although EChO was not selected, it was revised and competes as one of three missions for the ESA M4 mission under the name ARIEL (The Atmospheric Remote-Sensing Infrared Exoplanet Large-survey). Therefore, the centroiding performance analysis for the FGS (as done in Section 4.1) will be continued in the context of ARIEL.

This thesis serves as a handbook for the implementation of various centroiding techniques. In order to simplify the selection of centroiding methods, a compact overview of the characteristics of eight methods is provided in Table 3.1. It should be pointed out, that there is no default centroiding algorithm which is applicable in all cases. In fact, each of the investigated methods is capable of providing centroid estimations that are close to the analytically derived limits (Winick, 1986; Chen, 1987), if favourable conditions prevail.

A. Appendix

A.1. Configuration of StarSim

Simulation parameters can be configured by manipulating the files `stars_config.xml` and `datasim_config.xml`. For settings of type 'boolean', the values 'True', 'False', '0' and '1' are supported.

A.1.1. Entries of stars_config.xml

The simulated stellar field can be specified in this configuration file. Each star must be defined individually.

Setting Name	Type	Description
star	Xml-tag	Xml-entry that specifies a single star. The attributes 'pos_x', 'pos_y', 'signal' and 'is_target' must be specified per entry.
pos_x	float	The x-position of the star on the detector area in pixels.
pos_y	float	The y-position of the star on the detector area in pixels.
signal	float	The total signal of the star on sensor level prior to the reduction of quantum efficiency, gain, etc. in photons per second.
is_target	boolean	Specifies whether this entry is the target in the simulated observation. The final position of this star will be included in the output files.

Table A.1.: List of settings in `stars_config.xml`.

A.1.2. Entries of datasim_config.xml

This is the main configuration file of *StarSim*, where hardware and observational parameters can be specified.

Setting Name	Type	Description
detector_x	integer	The total number of pixel-columns on the sensor.
detector_y	integer	The total number of pixel-rows on the sensor.
flat_mean	float	The mean value for pixel sensitivities. Allowed value range: 0.0-1.0. Note that the simulated flat field is clipped at 1.0 per pixel.
flat_sigma	float	The standard deviation of the pixel sensitivities. Allowed value range: 0.0-1.0. Note that the simulated flat field is clipped at 1.0 per pixel.
flat_grad_lower_limit	float	A lower limit for the linear sensitivity gradient (start value). Allowed value range: 0.0-1.0.
flat_grad_upper_limit	float	A upper limit for the linear sensitivity gradient (end value). Allowed value range: 0.0-1.0.
flat_grad_changes	float	Changes of the upper and lower limits of the sensitivity gradient in multiple image generation mode. Allowed value range: 0.0-1.0.
flat_grad_angle	float	The angle of the sensitivity gradient in degrees. Allowed value range: 0-360
subpixelflat	custom	The intra-pixel flat field. Columns are delimited by '/' and rows are delimited by '//'. Example: 0.6/0.8/0.6//0.8/1.0/0.8//0.6/0.8/0.6 represents a 3×3 intra-pixel flat field. This corresponds to an oversampling factor of 3.
flatfield_count	integer	Number of flat field frames that are used for calibration if 'calibrate_image' is true.
psf	Xml-tag	Xml-entry that combines settings for the instrument's PSF. The following six settings must be specified within this tag.
plot_psf	boolean	Indicates if the PSF is plotted during the simulation.
psf_mode	string	Allowed values are 'read' and 'generate'.
psf_filename	string	If 'psf_mode' is 'read', this is the name of the text file, where the PSF is defined. Input file directory is specifed in setting 'data_input_directory'.
psf_is_oversampled	boolean	If 'psf_mode' is 'read', this indicates if the PSF file is oversampled or in final sensor resolution. The oversampling factor is defined by the resolution of the provided subpixelflat.
psf_fwhm_x	float	If 'psf_mode' is 'generate', this value defines the FWHM in x-dimension for the simulated Gaussian-PSF in units pixel.
psf_fwhm_y	float	If 'psf_mode' is 'generate', this value defines the FWHM in y-dimension for the simulated Gaussian-PSF in units pixel.
bias	float	Fixed bias value, that is included in the final image.
bias_count	integer	Number of bias frames that are used for calibration if 'calibrate_image' is true. These frames include read noise.

Setting Name	Type	Description
readout_noise	float	The mean read noise of a pixel in electrons per second.
exposure_time	float	The exposure time for the observation in seconds.
background_signal	float	The mean background of a pixel in photons per second.
QE	float	The fixed quantum efficiency of the sensor. This setting may only be used if flat_mean is not 1. Allowed value range: 0.0-1.0.
full_well_capacity	integer	The maximum value that is allowed for single pixels. Higher values will be truncated.
dark_average	float	The mean dark current in a pixel in electrons per second.
dark_count	integer	Number of dark frames that are used for calibration if 'calibrate_image' is true. These frames include bias and read noise.
hotpixel_amount	float	The fraction of hotpixels of all sensor pixels. These are randomly selected during the simulation. Allowed value range: 0.0-1.0.
hotpixel_lower_limit	float	A lower limit for the factor that is used to compute the increased dark values of hotpixels.
hotpixel_upper_limit	float	An upper limit for the factor that is used to compute the increased dark values of hotpixels.
calibrate_image	boolean	Indicates if calibration steps for dark current, read noise, bias and flat field shall be performed.
plot_subpixflat	boolean	Used to visualise the intra-pixel flat field.
plot_flat	boolean	Used to visualise the flat field.
plot_bias	boolean	Used to visualise the bias frame.
plot_starmask	boolean	Used to visualise the bias frame.
plot_background	boolean	Used to visualise the background frame.
plot_final_image	boolean	Used to visualise the final image that contains all noise sources.
plot_reduced_image	boolean	Used to visualise the calibrated image (if 'calibrate_image' is true).
plot_masterflat	boolean	Used to visualise the calibration frame for the flat field.
plot_masterbias	boolean	Used to visualise the calibration frame for the bias.
plot_hotpixel	boolean	Used to visualise current hotpixels only.
plot_dark	boolean	Used to visualise the dark frame.
plot_masterdark	boolean	Used to visualise the calibration frame for the flat field.
show_on_screen	boolean	If true, images are shown via pop-up during the simulation.
use_same_calibration_data	boolean	If true, the same bias, dark and flat frames are reused in simulations, where multiple images are generated.
save_image	boolean	If true, the images are saved as '.png' in the specified output directory.

Setting Name	Type	Description
save_data	boolean	If true, the images are saved as numpy-binaries in the specified output directory.
save_image_as_fits	boolean	If true, the final images are written into a single output file of type FITS in the specified output directory.
fits_file_name	string	The name of the FITS-output file.
data_input_directory	string	Input directory of the PSF in mode 'read'
data_output_directory	string	Directory that contains all the output generated by StarSim.
Rotation	Xml-tag	Xml-entry that holds information about the rotation of the instrument.
angle_speed	float	The speed of the rotation in degrees per second. Allowed value range: 0-360.
rotation_direction	string	Direction of the rotation. Only 'anticlockwise' and 'clockwise' are allowed.
rotation_axis_pos_x	float	x-position of the rotation axis on the sensor.
rotation_axis_pos_y	float	y-position of the rotation axis on the sensor.
ignore_target_star	boolean	If true, the target star is not affected by rotation (only used for special analyses).
Jitter	Xml-tag	Xml-entry that holds information about the jitter of the instrument.
apply_jitter	boolean	If true, jitter is included in the simulation.
input_file	string	Absolute path to the jitter file. The jitter file must contain three columns. One column for the current time in seconds starting from zero. And two other columns for defining current pitch and yaw in units pixel.
jitter_mode	string	Allowed values 'random_starttime' and 'fixed_starttime'.
jitter_starttime	float	If 'jitter_mode' is set to 'fixed_starttime', this time defines the start point in the input jitter file.
MultipleImages	Xml-tag	Xml-entry that combines settings for multiple image generation within single runs.
image_count	integer	This amount of final images will be created as output.
place_target_on_random_position	boolean	If true, the specified position for the target is ignored and a random position is computed for each iteration.
mul_imgs_per_iteration	integer	The amount of images for the current target position/signal. Only useful, if the next two settings are specified.
pos_step	float	If not zero, the target is moved along the x-axis during the simulation. This has been used for centroiding analyses.
signal_steps	float	If not zero, the targets signal is increased in each iteration. This has been used for centroiding analyses.

Setting Name	Type	Description
Glitches	Xml-tag	Xml-entry that combines settings for cosmic ray hits.
glitch_rate_average	float	For random glitch generation. The mean glitch rate per pixel and per second. Allowed value range: 0.0-1.0.
glitch_rate_sigma	Xml-tag	For random glitch generation. The standard deviation of the glitch rate per pixel and per second. Allowed value range: 0.0-1.0.
glitch	Xml-tag	Xml-entry that specifies a glitch on a fixed position.
type	string	Glitch-setting. Determines the shape of the glitch. Possible values: 'linear' - The glitch is represented by linear line on the detector between two points. 'spline' - The glitch is will be created by a spline interpolation through the specified points. 'point' - The glitch is located only at a single point.
spread_type	string	Glitch-setting. Determines how fast the glitch values decrease with the distance from the glitch centre. Possible values: 'gaussian' – Values fall off smoothly from an intra-pixel interpolation line that is considered as centre. 'hard' - The glitch value decreases in steps of 50% with increased distances to the glitch center.
points	floats	Glitch-setting. The x and y position of the glitch on the detector. Multiple points can be specified for types 'linear' and 'spline'.
width	float	Glitch-setting. The width of the glitch in pixels.
signal	float	Glitch-setting. The total signal of the glitch in electrons.
decay	float	Glitch-setting. Determines how glitch values change from start to endpoint in width and intensity. Allowed value range: 0.0-1.0. For 0.0, the glitch has a constant width/intensity from start to endpoint.

Table A.2.: List of settings in `datasim_config.xml`.

A.2. Abbreviations

ADC	Analog-to-Digital Converter
AOCS	Attitude and Orbit Control System
ARIEL	Atmospheric Remote-Sensing Infrared Exoplanet Large-survey
CCD	Charge-Coupled Device
CEE	Centroid Estimation Error
CHEOPS	CHaracterising ExOPlanets Satellite
CoG	Centre of Gravity
Corr-interpol.	Correlation-interpolation
Corr-upsampl.	Correlation-upsampling
EChO	Exoplanet Characterisation Observatory
FGS	Fine Guidance Sensor
FITS	Flexible Image Transport System
FWC	Full-Well Capacity
FWHM	Full Width at Half Maximum
G3P	Gaussian Three-Point Fit
IFSW	Instrument Flight Software
IWC	Intensity-Weighted Centre of Gravity
IWCoG	Iteratively-Weighted Centre of Gravity
JWST	James Webb Space Telescope
LMA	Levenberg-Marquardt Algorithm
MOST	Microvariability and Oscillations of Stars
NGST	Next Generation Space Telescope
ph	photons
ppm	parts per million
PSF	Point Spread Function
px	pixel
rms	root mean square
RoI	Region of Interest
RPS	Random Pixel Sensitivities
SAA	South Atlantic Anomaly
SNR	Signal-to-Noise Ratio
WCoG	Weighted Centre of Gravity

A.3. Implementations of Centroiding Algorithms

A.3.1. Centre of Gravity

Implementation in Python

```
""" Author: Roman Ferstl """
import numpy as np

"""
    Calculates the centre of gravity (or centre of mass) for given image
    without applying weights.

    Parameters
    ----------
        image -- 2D numpy array, the given image for which CoG will be
                 computed.

    returns centroid information (x,y) as floats.
"""
def CoG(image):

    total_intensity = np.sum(image)
    if (total_intensity == 0):
        print "Warning: Could not calculate CoG!"
        print "Total intensity in given image is zero."
        return 0,0
    # center[0] = x-coordinate.
    # center[1] = y-coordinate.
    center = [0,0]

    # calculate y coordinate
    for i in range (image.shape[0]) :
        center[1] += np.sum(image[i,:]) * i
    center[1] = center[1] / total_intensity

    # calculate x coordinate
    for j in range (image.shape[1]) :
        center[0] += np.sum(image[:,j]) * j
    center[0] = center[0] / total_intensity

    return center[0], center[1]
```

Implementation in C

```c
/*  Author: Roman Ferstl */
/*  Calculates the centre of gravity for given image.
 *
 *  Parameters
 *  ----------
 *      int size_x: Size of image in x.
 *      int size_y: Size of image in y.
 *      double **img: The centroid will be calculated for this two
 *                    dimensional image data.
 */
position CoG(int size_x, int size_y, double **img)
{
    int i, j;
    double total_intensity = 0.0;
    double rowsCollapsed[size_x], colsCollapsed[size_y];
    position pos = {0.0, 0.0};

    /* calculate y position of centroid */
    for (i=0; i<size_y; i++)
    {
        colsCollapsed[i]=0;
        for (j=0; j<size_x; j++)
        {
            colsCollapsed[i] += img[i][j];
        }
        pos.y += colsCollapsed[i] * (i+1);
        total_intensity += colsCollapsed[i];
    }

    pos.y = pos.y / total_intensity - 1;
    /* -1 cause centre of first cell is defined as position zero. */

    /* calculate x position of centroid */
    for (i=0; i<size_x; i++)
    {
        rowsCollapsed[i]=0;
        for (j=0; j<size_y; j++)
        {
            rowsCollapsed[i] += img[j][i];
        }
        pos.x += rowsCollapsed[i] * (i+1);
    }
    pos.x = pos.x / total_intensity - 1;

    return pos;
}
```

A.3.2. Weighted Centre of Gravity

Implementation in Python

```
""" Author: Roman Ferstl """
"""
Calculates the Weighted Centre of Gravity (WCoG) by using specified
weighting function. Also supports Intensity Weighted CoG (IWC).

Parameter
----------
    image     -- the given image for which CoG will be calculated.
    mode      -- function_weighted: the intensity distribution will be
                 weighted by a weighting function. Addtiionally,
                 parameter weighting_mode has to be set.
                 intensity_weighted: the intensity distribution itself
                                     is the weighting function.
                 psf_weighted: The given PSF template file located at
                               psf_location will be used.
    weighting_function --
        'gauss' -- A 2 dimensional gauss function. The spot center has
                   to be estimated.
        'psf' -- A psf template will be used for the weighting function.
                 Parameter psf_location has to be specified.
    psf_location -- Location of the psf file (for mode psf_weighted).
    estimated_star_pos -- (x,y) float coordinates representing an
                          initial guess of the star position.

returns x,y of the center of mass as floats.
"""
cog_mode_function_weighted  = 'function weighted'
cog_mode_intensity_weighted = 'intensity_weighted'
cog_weighting_mode_gauss    = 'gauss'
cog_weighting_mode_psf      = 'psf'
def WCoG(self, image, mode, weighting_function, psf_location = None,
    estimated_star_pos=None):

    if mode == cog_mode_function_weighted:
        if weighting_function == cog_weighting_mode_psf:
            pass
            # read your psf template file here
            #psfweighting = self.get_psf_weighting(psf_location,
                image.shape, estimated_star_pos)
            #image = image * psfweighting
        elif weighting_function == cog_weighting_mode_gauss:
            gaussweighting = generate_2D_gauss(estimated_star_pos[0],
                estimated_star_pos[1], self.psf_fwhm_x, self.psf_fwhm_y,
                image.shape)
            image = image * gaussweighting # weighting function is a
                gaussian with estimated center x0,y0
        else:
            raise Exception("Unknown weighting mode.")
    elif mode == cog_mode_intensity_weighted:
```

```
45          image = image * image # weighting function is the intensity
                distribution itself.
46
47      return CoG(image)
48
49  """
50  This function can be used to generate a 2 dimensional gaussian
51  distribution (e.g. on detector area).
52
53  Parameters
54  ----------
55      pos_x0: The x position of the gaussian centre. float value
56      pos_y0: The y position of the gaussian centre. float value
57      fwhm_x: The FWHM size in x direction given in pixel. float value
58      fwhm_y: The FWHM size in y direction given in pixel. float value
59      shape:  The two dimensional shape of the resulting area.
60      evaluation_limit:
61          If the shape of the resulting area is large it is useful to
62          evaluate the gauss function only near the center (spot) to
63          safe performance. This limit defines up to which pixel the
64          gauss function will be evaluated counted in pixel from spot
65          center (pos_y0, pos_y0). integer values.
66
67  Returns a 2 dimensional array containing the gaussian spot.
68  """
69  def generate_2D_gauss(pos_x0, pos_y0, fwhm_x, fwhm_y, shape,
        evaluation_limit = None):
70
71      gauss_array = np.zeros(shape)
72      upper_limit_x = shape[1] - 1
73      upper_limit_y = shape[0] - 1
74      # calculates the range for the gauss function
75      new_upper_limit_x = transform_to_zerobased(upper_limit_x,
            upper_limit_x)
76      new_upper_limit_y = transform_to_zerobased(upper_limit_y,
            upper_limit_y)
77
78      x0 = transform_to_zerobased(pos_x0, upper_limit_x)
79      y0 = transform_to_zerobased(pos_y0, upper_limit_y)
80
81      sx = utils.fwhm_to_sigma(fwhm_x)
82      sy = utils.fwhm_to_sigma(fwhm_y)
83
84      if evaluation_limit is None:
85          for i in range (upper_limit_x+1) :
86              for j in range (upper_limit_y+1) :
87                  if i >= 0 and i <= upper_limit_x and j >= 0 and j <=
                        upper_limit_y:
88                      gauss_array[j,i] = Gauss2D(x0, y0, sx, sy,
                            i-new_upper_limit_x, j-new_upper_limit_y)
89      else:
90          for i in range (int(pos_x0) - evaluation_limit, int(pos_x0) +
                evaluation_limit+1) :
91              for j in range (int(pos_y0) - evaluation_limit, int(pos_y0) +
                    evaluation_limit+1) :
```

```
                if i >= 0 and i <= upper_limit_x and j >= 0 and j <=
                    upper_limit_y:
                    gauss_array[j,i] = Gauss2D(x0, y0, sx, sy,
                        i-new_upper_limit_x, j-new_upper_limit_y)

    # normalize Gauss to 1.0
    gauss_array = gauss_array / np.sum(gauss_array)

    return gauss_array

"""
Generates a 2 dimensional gaussian spot. If sx and sy are equal, the spot
is circular. The function area is normalized to 1.

Parameters
----------
        x0 -- x position of the center
        y0 -- y position of the center
        sx -- sigma x = deviation in x direction
        sy -- sigma y = deviation in y direction
        x,y -- position of the spot that will be evaluated
"""
def Gauss2D(x0, y0, sx, sy, x, y):

    norm = 1./(sx*sy*2.*math.pi)
    return norm * np.exp(-( ((x-x0)**2.0)/(2.0*(sx**2.0)) +
        ((y-y0)**2.0/(2*(sy**2.0))) ))

"""
Transforms to a new coordinate system with detector center as (0,0).
Example: 0 to 256 --> -128 to +128

Parameters
----------
    pos              -- The position of in old coordinate system
                        (index based position starting with zero)
    old_upper_limit  -- upper limit of the old coordinate system
                        (e.g. 255) (index based)

returns the position in the center based coordinate system
"""
def transform_to_zerobased(pos, old_upper_limit):

    new_upper_limit = float(old_upper_limit / 2.0)
    return pos - new_upper_limit
```

Implementation in C

```
1  /*  Author: Roman Ferstl */
2
3  #define M_PI 3.141592653589793
4  #define FWHM_TO_SIGMA 0.424660900144010
5
6  /*  Calculates the Weighted Centre of Gravity (WCoG) or Intensity
7   *  Weighted Centre of Gravity (IWC) for given image.
8   *
9   *  Parameters
10  *  ----------
11  *      int size_x: Size of image in x.
12  *      int size_y: Size of image in y.
13  *      double **img: The centroid will be calculated for this
14  *                    two dimensional image data.
15  *      wcog_mode mode: GAUSS or IWC.
16  *      double estimated_x: estimated star position in x.
17                              Required in all weighting modes.
18  *      double estimated_y: estimated star position in y.
19                              Required in all weighting modes.
20  *      double gauss_fwhm_x: Only required in more 'GAUSS'.
21                 FWHM of 2D Gaussian function in x dimension.
22  *      double gauss_fwhm_y: Only required in more 'GAUSS'.
23                 FWHM of 2D Gaussian function in y dimension.
24   */
25 position WCoG(int size_x, int size_y, double **img, wcog_mode mode, double
       estimated_x, double estimated_y, double gauss_fwhm_x, double
       gauss_fwhm_y)
26 {
27     int i, j;
28     double **weights;
29
30     if (mode == GAUSS)
31     {
32         /* Weighting template is obtained by evaluation of analytical 2D
               Gaussian function */
33         weights = generate2DGauss(estimated_x, estimated_y, gauss_fwhm_x,
               gauss_fwhm_y, size_x, size_y, -1, -1);
34     }
35     else if (mode == IWC)
36     {
37         /* Weighting template is the intensity function */
38         weights = img;
39     }
40
41     /* In order to save performance, this loop can be moved into method
           CoG. */
42     for (i=0; i<size_y; i++)
43     {
44         for (j=0; j<size_x; j++)
45         {
46             weights[i][j] *= img[i][j];
```

```
47          }
48      }
49
50      /* see separate implementation of CoG */
51      return CoG(size_x, size_y, weights);
52  }
53
54  /*  This function can be used to generate a 2 dimensional
55   *  gaussian distribution (e.g. on detector area).
56   *
57   *  Parameters
58   *  ----------
59   *      double pos_x0: The x position of the gaussian centre.
60   *      double pos_y0: The y position of the gaussian centre.
61   *      double fwhm_x: The FWHM size in x direction given in pixel.
62   *      double fwhm_y: The FWHM size in y direction given in pixel.
63   *      int size_x: The size in x of the resulting 2d-array
64   *                  containing the gaussian spot.
65   *      int size_y: The size in y of the resulting 2d-array
66   *                  containing the gaussian spot.
67   *      evaluation_limits:
68   *      CURRENTLY NO IMPLEMENTED (can be used to increase performance)
69   *      If the shape of the resulting area is large it is useful to
70   *      evaluate the gauss function only near the center (spot) to safe
71   *      performance. This limit defines up to which pixel the gauss
72   *      function will be evaluated counted in pixel from spot center.
73   *
74   *  Returns a pointer to 2 dimensional double array containing the
75   *  gaussian spot.
76   */
77  double **generate2DGauss(double pos_x0, double pos_y0, double fwhm_x,
        double fwhm_y, int size_x, int size_y, int eval_limit_x, int
        eval_limit_y)
78  {
79      int upper_limit_x, upper_limit_y, i, j;
80      double x0, y0, sx, sy, new_upper_limit_x, new_upper_limit_y, total_sum;
81      double **result = (double **) malloc(sizeof(double *)*size_y);
82
83      upper_limit_x = size_x - 1;
84      upper_limit_y = size_y - 1;
85      /* transform to zero based coordinate system */
86      new_upper_limit_x = (double)(upper_limit_x - (double)(upper_limit_x /
            2.0));
87      new_upper_limit_y = (double)(upper_limit_y - (double)(upper_limit_y /
            2.0));
88      x0 = (double)(pos_x0 - upper_limit_x / (double)2.0);
89      y0 = (double)(pos_y0 - upper_limit_y / (double)2.0);
90      sx = fwhm_x * FWHM_TO_SIGMA;
91      sy = fwhm_y * FWHM_TO_SIGMA;
92      total_sum = 0;
93
94      for (j=0; j<=upper_limit_y; j++)
95      {
96          result[j] = (double *) malloc(sizeof(double)*size_x);
97
```

```
98          for (i=0; i<=upper_limit_x; i++)
99          {
100             if (i <= upper_limit_x && j >= 0 && j <= upper_limit_y)
101             {
102                 result[j][i] = gauss2D(x0, y0, sx, sy,
                        i-new_upper_limit_x, j-new_upper_limit_y);
103                 total_sum += result[j][i];
104             }
105         }
106     }
107     /* normalization must not be performed if gauss2D is normalized */
108     /* normalizeImage(size_x, size_y, result, total_sum); */
109
110     return result;
111 }
112
113 /*
114  * Evaluates a 2 dimensional Gaussian function. If sx equals sy,
115  * the function is circular symmetric around x0, y0.
116  * The function is normalized to 1.
117  *
118  * Parameters
119  * ----------
120  *          x0 -- x position of the center
121  *          y0 -- y position of the center
122  *          sx -- sigma x = standard deviation in x direction
123  *          sy -- sigma y = standard deviation in y direction
124  *          x,y -- position of the spot that will be evaluated
125  * Requires
126  * --------
127  *          #include <math.h>
128 */
129 double gauss2D(double x0, double y0, double sx, double sy, double x,
        double y)
130 {
131     return (double)1.0/(sx*sy*2.0*M_PI) * exp(-(
            (pow((x-x0),2))/(2.0*(pow(sx, 2))) + (pow((y-y0),2.0)/(2*(pow(sy,
            2)))) ));
132 }
```

A.3.3. Iteratively Weighted Centre of Gravity

Implementation in Python

```
""" Author: Roman Ferstl """
"""
    Iteratively Weighted Centre of Gravity (IWCoG)
    Calculates the CoG iteratively by using centroid estimations.

    Parameter
    ----------
        iterations     -- integer, indicates how many iterations
                          shall be performed.
        see method 'WCoG' for documentation of other parameters.

    returns x,y of the center of mass as floats.
"""
def calc_iterative_CoG(image, iterations, weighting_function,
   psf_location, estimated_star_pos, actualStarPosition = None, minError =
   None):

    x = estimated_star_pos[0]
    y = estimated_star_pos[1]

    for i in range(1,iterations+1):
        x, y = CoG(image, const.cog_mode_function_weighted,
           weighting_function, psf_location, (x,y))
        if minError != None:
            xtmp, ytmp = x,y
            xtmp, ytmp = self.backtransform_coords(xtmp, ytmp)
            cee = math.sqrt((xtmp-actualStarPosition[0])**2 +
                (ytmp-actualStarPosition[1])**2)

            if (cee <= minError):
                iterations = i
                break

    return x,y
```

Implementation in C

```
/*  Author: Roman Ferstl */
/*  Calculates the Iteratively Weighted Centre of Gravity (IWCoG) for
 *  given image. The number of iterations has to be specified. Centroid
 *  results form previous results are used to improve the position of
 *  the weighting function.
 *
 *  Parameters
 *  ----------
 *      int iterations: integer, indicates how many iterations
 *                      shall be performed.
 */
position IWCoG(int size_x, int size_y, double **img, wcog_mode mode,
    double estimated_x, double estimated_y, double gauss_fwhm_x, double
    gauss_fwhm_y, int iterations)
{
    int i;
    position centroid;

    centroid.x = estimated_x;
    centroid.y = estimated_y;

    for (i=0; i<iterations; i++)
    {
        centroid = WCoG(size_x, size_y, img, mode, centroid.x, centroid.y,
            gauss_fwhm_x, gauss_fwhm_y);
        printf("iteration %d - centroid: (%f, %f)\n", i+1, centroid.x,
            centroid.y);
    }
    return centroid;
}
```

A.3.4. Correlation-based Centroiding

Implementation in Python

```
""" Author: Roman Ferstl """
import math
import numpy as np
import scipy

"""
Calculates the spot of best correlation (centroid) between the image
and a reference template. The reference template can be created within
this method by given psf parameters or it can be read from file.
Upsampling has to be performed in order to reach subpixel accuracy.
The reference spot and the image will have the same sampling before
the correlation is performed.

Parameters
----------
    image       -- 2dim numpy array. The given image that will be
                   correlated with the template in order to find the star
                   position. Can be the whole detector image or only a
                   defined region of interest. Computational effort
                   is very high if the operation is performed for the
                   whole detector size.
    isWindow    -- boolean. Indicates if the given image is only a small
                   section of the whole image (region of interest).
                   If yes, the calculated coordinates will be re-
                   transformed to the original coordinate system of
                   the whole image.
    mode        -- 'upsampling' or 'interpolate'
    upsampling -- integer. Defines the factor that will be used for
                  upsampling the original image.
    boundary_type -- string. The boundary condition type as string.
                     available options: 'nearest', 'constant',
                     'mirror', 'reflect', 'wrap'
    boundary_val  -- float. This parameter is only used in combination
                     with boundary_type = 'constant'
    psf_mode      -- 'generate': template psf will be generated
                                 using a 2D-Gauss function.
                  -- 'read':     template will be read from file.
    psf_location  -- string. Path to template file that will be used
                     for correlation including file name.

returns the position of best correlation as float values.
"""
def calc_correlation(self, image, isWindow, mode, upsampling,
   boundary_type, psf_mode, psf_location, boundary_val = 0.0):

    if psf_mode == "generate":
        # generate psf template
        if (mode == "upsampling"):
            fwhm_x = self.psf_fwhm_x * upsampling
```

```
            fwhm_y = self.psf_fwhm_y * upsampling
        else:
            fwhm_x = self.psf_fwhm_x
            fwhm_y = self.psf_fwhm_y

        psf_frame_width  = int(math.ceil(fwhm_x * 2))
        psf_frame_height = int(math.ceil(fwhm_y * 2))
        # force uneven pixel counts for psf template (for centering)
        if psf_frame_width % 2 == 0:
            psf_frame_width += 1
        if psf_frame_height % 2 == 0:
            psf_frame_height += 1
        # determine the centre of the image
        psf_pos_x0 = int((psf_frame_width-1)  / 2.0)
        psf_pos_y0 = int((psf_frame_height-1) / 2.0)
        #for definition of generate_2D_gauss: see code section of
        #'Weighted Centre of Gravity'
        psf = glob.generate_2D_gauss(psf_pos_x0, psf_pos_y0, fwhm_x,
            fwhm_y, (psf_frame_height, psf_frame_width))
    elif psf_mode == "read":
        psf = np.loadtxt(psf_location, delimiter="\t")
        psf = np.around(psf, decimals=5)
        psf = scipy.ndimage.zoom(psf, upsampling, order=0)

    # upsampling of the original image to determine sub-pixel accuracy
    upsampled_image = image
        if mode == "upsampling" and upsampling > 1:
            upsampled_image = scipy.ndimage.zoom(upsampled_image,
                upsampling, order=0)

    # compute the correlation function
    output = scipy.ndimage.filters.correlate(upsampled_image, psf,
        mode=boundary_type, cval=boundary_val)

    # get coordinates of peak correlation function
    maxIndexTuple = np.unravel_index(np.argmax(output), output.shape)

    if mode == "upsampling":
        x = glob.transform_to_original(maxIndexTuple[1], upsampling)
        y = glob.transform_to_original(maxIndexTuple[0], upsampling)
        xerr = 1.0 / float(upsampling*2.0)
        yerr = 1.0 / float(upsampling*2.0)
    elif mode == "interpolate":
        x, xerr = self.get_correlation_centroid_and_error(upsampled_image,
            psf, noise_estimator, maxIndexTuple[1], maxIndexTuple[0],
            dim='x')
        y, yerr = self.get_correlation_centroid_and_error(upsampled_image,
            psf, noise_estimator, maxIndexTuple[1], maxIndexTuple[0],
            dim='y')

    if isWindow:
        x, y = self.backtransform_coords(x, y)

    return x, y, xerr, yerr

```

```
"""
Transforms from oversampled coordinates to the actual detector sized
coordinate system.
"""
def transform_to_original(pos, oversampling):

    if oversampling == 1:
        return pos
    return (pos - (oversampling / 2.0 - 0.5)) / oversampling

"""
Back-transform from ROI coordinates to original coordinates.
window_position: x and y pos. of window in original coord. system.
"""
def backtransform_coords(self, x, y):

    x_orig = x + self.window_position[0]
    y_orig = y + self.window_position[1]
    return x_orig, y_orig

"""
Based on 'Correlation wave-front sensing algorithms for Shack-
Hartmann-based Adaptive Optics using a point source' (Poyneer, 2003)
Solutions had to be adapted for zero shift, as size of reference spot
is not equal to size of image.
Zero shift is assumed to be at provided centroid estimation.

Parameters
----------
        image   --  2d numpy array, the image containing the target
        pattern -- 2d numpy array, the template used for cross-correlation
                   (e.g. psf model)
        noise_estimator -- float, average Gaussian noise inside a pixel.
        est_x   -- estimated centroid x position based on cross-
                   correlation result. Assumed to be zero shift position.
        est_y   -- estimated centroid y position based on cross-
                   correlation result. Assumed to be zero shift position.
"""
def get_correlation_centroid_and_error(self, image, pattern,
   noise_estimator, est_x, est_y, dim):

    m0 = 0
    m1 = 0
    mm1 = 0
    sig_one_square  = 0
    sig_mone_square = 0
    rnterm = 0

    # transform to zero shift
    dx = est_x - pattern.shape[1]//2
    dy = est_y - pattern.shape[0]//2

    if dim == 'x':
        for i in range (pattern.shape[1]-1) :
            for j in range (pattern.shape[0]-1) :
```

```
                m0 += pattern[j,i]*image[dy+j,dx+i]
                mm1 += pattern[j,i+1]*image[dy+j,dx+i]
                m1 += pattern[j,i-1]*image[dy+j,dx+i]
                sig_one_square  += pattern[j,i-1]**2*noise_estimator**2
                sig_mone_square +=
                   pattern[j,i+1]*pattern[j,i-1]*(noise_estimator**2)
                rnterm = pattern[j,i-1]* (pattern[j,i-1] - pattern[j,i+1])

        centr = 0.5 * (m1 - mm1) / (m1 + mm1 - 2*m0)
        centr = est_x - centr
    elif dim == 'y':
        for i in range (pattern.shape[1]-1) :
            for j in range (pattern.shape[0]-1) :
                m0 += pattern[j,i]*image[dy+j,dx+i]
                mm1 += pattern[j+1,i]*image[dy+j,dx+i]
                m1 += pattern[j-1,i]*image[dy+j,dx+i]
                sig_one_square  += pattern[j-1,i]**2*noise_estimator**2
                sig_mone_square +=
                   pattern[j+1,i]*pattern[j-1,i]*(noise_estimator**2)
                rnterm = pattern[j-1,i]* (pattern[j-1,i] - pattern[j+1,i])

        centr = 0.5 * (m1 - mm1) / (m1 + mm1 - 2*m0)
        centr = est_y - centr

    var = (sig_one_square - sig_mone_square)/(8*(m0-m1)**2) +
       noise_estimator**2*rnterm/(8*(m0-m1)**2)
    err = np.sqrt(var)

    return centr, err
```

A.3.5. Gaussian Three-Point Fitting

Implementation in Python

```
""" Author: Roman Ferstl """
import numpy as np
"""
Performs a Gaussian fit to 3 points in order to determine the star
position and returns the positions x,y of the maximum of the Gaussian
curves.

Parameter
----------
    image         -- 2D numpy array. Given image for which the Gaussians
                     will be fitted.
    collapse_cells-- boolean. If 'True' the 2d image will be collapsed to
                     a single row/column.
    estimated_pos -- integer tuple, containing estimated centroid
                     position (x,y).
    isWindow      -- indicates if the given image is only a small section
                     of the whole image (region of interest). If 'true'
```

```
                    the calculated coordinates will be re-transformed to
                    the original coordinate system of the whole image.

returns best fit in x and y.
"""
def calc_gauss_3point_fit_center(self, image, collapse_cells,
    estimated_pos, isWindow = False):

    bPixX = estimated_pos[0]
    bPixY = estimated_pos[1]
    if collapse_cells == True:
        collapsedImgForX = collapseTo1D(image, collapse_rows)
        collapsedImgForY = collapseTo1D(image, collapse_cols)

    if collapse_cells == True:
        x = fit_gauss_3points(collapsedImgForX[bPixX-1],
            collapsedImgForX[bPixX], collapsedImgForX[bPixX+1], bPixX)
        y = fit_gauss_3points(collapsedImgForY[bPixY-1],
            collapsedImgForY[bPixY], collapsedImgForY[bPixY+1], bPixY)
    else:
        x = fit_gauss_3points(image[bPixY, bPixX-1], image[bPixY, bPixX],
            image[bPixY, bPixX+1], bPixX)
        y = fit_gauss_3points(image[bPixY-1, bPixX], image[bPixY, bPixX],
            image[bPixY+1, bPixX], bPixY)

    if isWindow:
        # for implementation of backtransform_coords see code for
        # correlation-based centroiding'
        x, y = self.backtransform_coords(x, y)

    return x, y

"""
Performs a Gaussian fit to 3 points.
If the 3 points lie on a Gaussian curve. The solution is a 2nd grade
polynomial represented by the given formula (for equal pixel/point
distances). Returns the position of maximum of the Gaussian curve
as x,y coordinate.

Parameter
----------
    leftCell     -- the value of the 1st (left)   point of Gaussian fit
    middleCell   -- the value of the 2nd (middle) point of Gaussian fit
    rightCell    -- the value of the 3rd (right)  point of Gaussian fit
    middleCellCoordinate  -- the coordinate of the point in the middle.

returns the position x0 (maximum of Gaussian fit) as float
"""
def fit_gauss_3points(self, leftCell, middleCell, rightCell,
    middleCellCoordinate):
    if rightCell <= 0.0 or leftCell <= 0.0 or middleCell <= 0.0:
        print "WARNING: Cannot fit Gaussian curve if a fit point is zero.
            Set fit point to 0.01"
        rightCell  = 0.01 if rightCell  <= 0.0 else rightCell
        middleCell = 0.01 if middleCell <= 0.0 else middleCell
```

```
        leftCell   = 0.01 if leftCell   <= 0.0 else leftCell

    return (np.log(rightCell) - np.log(leftCell)) /
        (2.0*(2.0*np.log(middleCell) - np.log(leftCell) -
        np.log(rightCell))) + middleCellCoordinate

"""
Collapses a 2 dimensional numpy array to a single row or column.
In this case collapse means that the sum of all pixel
e.g in one column will be represented by a single cell in the new array.

Parameters
----------
    source -- 2D numpy array. Represents the image.
    collapse_type -- string. Can be "rows" or "cols".
             "rows" -- Array rows will be collapsed to a single row.
             "cols" -- Array columns will be collapsed to a single column.

returns a 1D numpy array.
"""
collapse_rows      =  "rows"
collapse_cols      =  "cols"
def collapseTo1D(source, collapse_type = collapse_rows):

    if collapse_type == collapse_rows:
        resultSize = source.shape[1]
    elif collapse_type == collapse_cols:
        resultSize = source.shape[0]

    result = np.zeros(resultSize)
    for i in range (resultSize) :
        if collapse_type == collapse_cols:
            result[i] = np.sum(source[i,:])
        elif collapse_type == collapse_rows:
            result[i] = np.sum(source[:,i])

    return result
```

A.3.6. Levenberg-Marquardt Algorithm

Implementation in Python

```
""" Author: Roman Ferstl """
import numpy as np
import scipy
import math
"""
Performs a least square fit on the given image to find the star position.
The two dimensional Gaussian which is also used for PSF generation can be
used for the fit. Alternatively a fit function designed for CHEOPS can be
used. The method uses the function scipy.optimize.curve_fit which copmutes
the Levenberg-Marquardt-algorithm.

Parameter
----------
    image            -- 2D numpy array. The given image for which the fit
                        will be performed.
    useKnownPsfSize  -- boolean. If 'True' the unknown fit parameters will
                        only be x0 and y0 and the algorithm will be
                        faster. Otherwise sigma x and sigma y will also be
                        determined by the fit.
    estimated_star_pos -- (x,y) float coordinates representing an initial
                          guess of the star position. The coordinates has
                          to be given in correct coordinate system.
    fit_mode         -- 'gauss'  -- A two dimensional gauss function will
                        be used for the fit. If useKnownPsfSize is True:
                        FWHM in x and y must be provided.
                     -- 'cheops' -- A fit function specificilly designed
                        for the CHEOPS psf will be used.
    isWindow      -- indicates if the given image is only a small section
                     of the whole image (region of interest). If 'true'
                     the calculated coordinates will be re-transformed to
                     the original coordinate system of the whole image.

returns the expected star position x,y obtained by fit parameters.
"""
def calc_least_squares(self, image, fit_mode, useKnownPsfSize,
    estimated_star_pos=None, isWindow=False):

    #1. normalize the image to 1
    image = image / np.sum(image)

    #2. prepare data for fit: x, y, z have to  be separate arrays
    xsize = image.shape[1]
    ysize = image.shape[0]
    # for x:
    xcoords = np.tile(np.arange(xsize), ysize)
    # for y:
    # use alternative implementation instead of iterating to avoid
    # performance issues.
    ycoords = np.empty_like(xcoords)
```

```
    j = 0
    for i in range(ycoords.shape[0]):
        if i % (xsize) == 0 and i > 0:
            j = j + 1
        ycoords[i] = j
    # for z:
    zcoords = image.flatten()

    #3. combine x and y again to one single parameter in order to use
       curve_fit.
    xy = np.array([xcoords, ycoords])

    #4. get initial guess of the star position
    initial_guess = estimated_star_pos

    #5. fit the data
    if fit_mode == 'gauss':
        if useKnownPsfSize == True:
            popt, pcov =
               scipy.optimize.curve_fit(Gauss2D_for_curvefit_psfknown, xy,
               zcoords, p0=initial_guess)
        else:
            if initial_guess is not None:
                initial_guess = (initial_guess[0], initial_guess[1],
                   fwhm_to_sigma(self.psf_fwhm_x),
                   fwhm_to_sigma(self.psf_fwhm_y))
            popt, pcov = scipy.optimize.curve_fit(Gauss2D_for_curvefit,
               xy, zcoords, p0=initial_guess)
    elif fit_mode == 'cheops':
            if initial_guess is not None:
                initial_guess = (initial_guess[0], initial_guess[1])
            popt, pcov =
               scipy.optimize.curve_fit(cheops_psf_fit2D_for_curvefit, xy,
               zcoords, p0=initial_guess)
    else:
        raise Exception("calc_least_squares: unsupported fit_mode!")

    #6. get errors
    x = popt[0]
    y = popt[1]

    perr = np.sqrt(np.diag(pcov))
    print "ERROR x0: ", str(perr[0])
    print "ERROR y0: ", str(perr[1])

    if isWindow:
        # for implementation of backtransform_coords see code for
        # correlation-based centroiding'
        x, y = self.backtransform_coords(x, y)

    return x,y

"""
This function is a wrapper of 'Gauss2D' for the use of  scipy.optimize.
curve_fit. The function is normalized to 1. PSF size will be fitted as
```

```
well. For definition of function 'Gauss2D' see code section of
'Weighted Centre of Gravity'.
"""
def Gauss2D_for_curvefit(self, xy, x0, y0, sx, sy) :
    return Gauss2D(x0, y0, sx, sy, xy[0], xy[1])

"""
This function is a wrapper of 'Gauss2D' for the use of scipy.optimize.
curve_fit. Function can be used when PSF size is known. For definition
of function 'Gauss2D' see code section of 'Weighted Centre of Gravity'.
"""
def Gauss2D_for_curvefit_psfknown(self, xy, x0, y0) :
    sx = fwhm_to_sigma(self.psf_fwhm_x)
    sy = fwhm_to_sigma(self.psf_fwhm_y)
    return Gauss2D(x0, y0, sx, sy, xy[0], xy[1])

"""
Converts full width half maximum to standard deviation.
"""
def fwhm_to_sigma(fwhm):
    return fwhm / (2*math.sqrt(2*np.log(2)))
```

Implementation in C

```
/*  Author: Roman Ferstl */
/*  Calculates centroid by solving 2D non-linear least squares problem.
 *  A Levenberg-Martquardt algorithm is carried out using the MINPACK
 *  implementation.
 *
 *  Parameters
 *  ----------
 *      int size_x: Size of image in x.
 *      int size_y: Size of image in y.
 *      double **img: The centroid will be calculated for this two
 *                    dimensional image data.
 *      lq_fit_mode mode: 'LQ_GAUSS': a 2D gauss function will be fit
 *                        (sig_x and sig_y must be provided).
 *                        'LQ_CHEOPS_MODEL': The provided fit function
 *                        has been derived from cheops psf model.
 *      double estimated_x: star position in x. Required in all modes.
 *      double estimated_y: star position in y. Required in all modes.
 *      double sig_x: Only required in more 'LQ_GAUSS'. STDEV of the
 *                    2D Gaussian function in x dimension.
 *      double sig_y: Only required in more 'LQ_GAUSS'. STDEV of the
 *                    2D Gaussian function in y dimension.
 *      double amplitude: Used to adjust the amplitude of the fitting
 *                    function (this is crucial for a good fit).
 *                    Required in all modes.
 */
position leastSquaresFit(int size_x, int size_y, double **img, lq_fit_mode
   mode, double estimated_x, double estimated_y, double sig_x, double
   sig_y, double amplitude)
```

```
{
    const int n = 2, m = size_x * size_y;
    int iwa[n], lwa=m*n+5*n+m, info = 0;
    real tol, fnorm, par[n], fvec[m], wa[lwa];
    position centroid;
    minimize_function_data_t data;

    par[0] = estimated_x;
    par[1] = estimated_y;
    tol = sqrt(__cminpack_func__(dpmpar)(1));
    data.imgdata = img;
    data.xsize = size_x;
    data.ysize = size_y;
    data.amplitude = amplitude;
    if (mode == LQ_GAUSS)
    {
        data.sigma_x = sig_x;
        data.sigma_y = sig_y;
        info = __cminpack_func__(lmdif1)(minimize_gauss2D, &data, m, n,
           par, fvec, tol, iwa, wa, lwa);
    }
    else if (mode == LQ_CHEOPS_MODEL)
    {
        info = __cminpack_func__(lmdif1)(minimize_cheops_psf_model, &data,
           m, n, par, fvec, tol, iwa, wa, lwa);
    }
    if (info != 1) {
        /* info = 0  improper input parameters.                     */
        /* info = 1  algorithm estimates that the relative error    */
        /*           in the sum of squares is at most tol.          */
        /* info = 2  algorithm estimates that the relative error    */
        /*           between x and the solution is at most tol.     */
        /* info = 3  conditions for info = 1 and info = 2 both hold. */
        /* info = 4  fvec is orthogonal to the columns of the       */
        /*           jacobian to machine precision.                 */
        /* info = 5  number of calls to fcn has reached or          */
        /*           exceeded 200*(n+1).                            */
        /* info = 6  tol is too small. no further reduction in      */
        /*           the sum of squares is possible.                */
        /* info = 7  tol is too small. no further improvement in    */
        /*           the approximate solution x is possible.        */
        printf("leastSquaresFit: exit parameter - %10i\n\n", info);
    }

    fnorm = __cminpack_func__(enorm)(m, fvec);
    centroid.x = par[0];
    centroid.y = par[1];

    return centroid;
}

/*  This is the function called by Levenberg Marquarth algorithm of
 *  MINPACK (lmdif1). This function header is defined by MINPACK and
 *  must not be changed. The function points to evaluate are provided
 *  as pointer in p. Those are not used within MINPACK (only here).
```

```
 *  This function must provide the difference between fmodel(x,y) to
 *  fmeasured(x,y) and NOT THE SQUARES.
 *
 *  Here x[0], x[1], x[2] = x0, y0, Amplitude
 *
 *  MINPACK DOCUMENTATION:
 *  ----------------------
 *  calculate the functions at x and return the values in fvec[0]
 *  through fvec[m-1]. The value of iflag should not be changed by fcn
 *  unless the user wants to terminate execution of lmdif_
 *  (or lmdif1_). In this case set iflag to a negative integer.
 *
 *  m is a positive integer input variable set to the number of functions.
 *  n is a positive integer input variable set to the number of variables.
 *  n must not exceed m.
 *  x is an array of length n. On input x must contain an initial estimate
 *  of the solution vector. On output x contains the final estimate of the
 *  solution vector.
 *  fvec is an output array of length m which contains the functions
 *  evaluated at the output x.
 */
int minimize_gauss2D(void *p, int m, int n, const real *x, real *fvec, int
    iflag)
{
  int i, j, k=0;
  double **img = ((minimize_function_data_t*)p)->imgdata;
  const int xsize = ((minimize_function_data_t*)p)->xsize;
  const int ysize = ((minimize_function_data_t*)p)->ysize;
  const double sig_x = ((minimize_function_data_t*)p)->sigma_x;
  const double sig_y = ((minimize_function_data_t*)p)->sigma_y;
  const double A = ((minimize_function_data_t*)p)->amplitude;
  (void)iflag;

  assert(m == xsize * ysize);
  for (i = 0; i < ysize; ++i) /* i -> y position */
  {
      for (j = 0; j < xsize; ++j)  /* j -> x position */
      {
          fvec[k] = img[i][j] - gauss2D_withAmpl(x[0], x[1], A, sig_x,
              sig_y, j, i);
          ++k;
      }
  }
  return 0;
}
```

A.4. Data Tables on Cosmic Ray Hit Analysis

Glitch No.	Exposure No.	Pixels affected	Total Signal [ADU]	Brightest Pixel [ADU]	In SAA
1	924	6	34879	13614	yes
2	924	1	2828	2828	yes
3	924	2	1014	552	yes
4	924	1	515	515	yes
5	924	1	607	607	yes
6	924	1	2252	2252	yes
7	924	1	9949	9949	yes
8	2359	2	527	200	yes
9	2359	1	232	232	yes
10	2359	4	1777	870	yes
11	2359	1	549	549	yes
12	2359	4	1401	423	yes
13	2359	7	4466	2546	yes
14	2359	17	95619	15753	yes
15	2363	15	3708	561	yes
16	2363	15	5355	235	yes
17	2363	1	400	400	yes
18	2363	2	1473	1111	yes
19	2363	2	580	487	yes
20	2363	5	2010	886	yes
21	2363	4	1082	417	yes
22	2585	1	189	189	yes
23	2585	1	389	389	yes
24	2585	1	244	244	yes
25	2585	3	471	200	yes
26	2585	3	556	329	yes
27	2585	9	1854	504	yes
28	2585	4	537	200	yes
29	2585	1	162	162	yes
30	2585	1	114	114	yes
31	5614	6	17948	9149	yes
32	5614	5	778	274	yes
33	5614	9	3085	680	yes
34	5614	11	1885	282	yes
35	5614	2	342	251	yes
36	7005	11	18894	11792	yes
37	7005	2	386	257	yes
38	7005	1	168	168	yes
39	7005	6	1635	588	yes
40	7005	2	288	171	yes
41	7005	1	110	110	yes
42	7005	4	1035	371	yes

Glitch No.	Exposure No.	Pixels affected	Total Signal [ADU]	Brightest Pixel [ADU]	In SAA
43	7005	1	391	391	yes
44	7005	4	878	347	yes
45	7005	1	345	345	yes
46	7005	1	154	154	yes
47	7005	10	1774	416	yes
48	7222	10	15602	5552	yes
49	7222	4	2468	1622	yes
50	7222	12	3243	611	yes
51	7222	3	896	611	yes
52	7222	2	733	441	yes
53	7222	3	406	204	yes
54	7222	1	121	121	yes
55	7222	3	745	363	yes
56	7222	4	399	122	yes
57	7222	2	645	452	yes
58	7222	3	320	128	yes
59	8445	17	60756	14495	yes
60	8445	1	150	150	yes
61	8445	5	1069	326	yes
62	8445	4	637	206	yes
63	8445	2	342	188	yes
64	8445	2	432	237	yes
65	8445	2	854	583	yes
66	8445	1	321	321	yes
67	8445	1	584	584	yes
68	8445	2	253	142	yes
69	8445	4	582	256	yes
70	8445	7	963	259	yes
71	8445	2	434	321	yes
72	9836	3	643	420	yes
73	9836	4	547	163	yes
74	9836	3	786	337	yes
75	9836	2	289	229	yes
76	10054	8	13767	6870	yes
77	10054	1	1319	1319	yes
78	10054	5	1284	390	yes
79	10054	2	430	239	yes
80	10054	7	2092	762	yes
81	10054	7	2777	940	yes
82	10054	1	152	152	yes
83	10054	1	262	262	yes
84	10054	1	177	177	yes
85	10054	5	895	203	yes
86	10054	2	270	138	yes
87	10054	6	1507	479	yes

Glitch No.	Exposure No.	Pixels affected	Total Signal [ADU]	Brightest Pixel [ADU]	In SAA
88	11699	4	13231	6203	yes
89	11699	1	191	191	yes
90	11699	2	249	151	yes
91	11699	1	112	112	yes
92	18012	4	9443	4899	yes
93	18012	5	1712	793	yes
94	18012	8	2972	1047	yes
95	18012	1	120	120	yes
96	18012	3	506	222	yes
97	20620	7	18398	8700	yes
98	20620	3	864	322	yes
99	20620	1	163	163	yes
100	20620	2	549	365	yes
101	20620	2	934	750	yes
102	20620	2	1001	807	yes
103	20620	3	1489	1013	yes
104	20620	2	456	296	yes
105	20629	7	6164	3561	yes
106	20629	5	8517	7726	yes
107	20629	3	1422	846	yes
108	22657	1	173	173	yes
109	22657	7	8515	3807	yes
110	22657	2	230	118	yes
111	22657	2	246	136	yes
112	22657	1	352	352	yes
113	22657	1	288	288	yes
114	22657	2	325	185	yes
115	22657	1	168	168	yes
116	25482	1	160	160	yes
117	25482	17	48409	2122	yes
118	25482	2	1614	1274	yes
119	30198	12	55169	15729	yes
120	30198	8	2355	1112	yes
121	30198	2	431	336	yes
122	30198	2	857	675	yes
123	30198	5	2109	1276	yes
124	30198	2	208	114	yes
125	30198	4	537	162	yes
126	30198	1	283	283	yes
127	30198	3	786	324	yes
128	30198	1	159	159	yes
129	31578	2	474	280	yes
130	31578	1	164	164	yes
131	31578	17	5373	438	yes
132	31578	4	659	424	yes

Glitch No.	Exposure No.	Pixels affected	Total Signal [ADU]	Brightest Pixel [ADU]	In SAA
133	31578	1	166	166	yes
134	31578	2	631	550	yes
135	31578	4	906	342	yes
136	31578	4	619	254	yes
137	31578	12	1793	821	yes
138	31578	10	2069	813	yes
139	31578	2	188	134	yes
140	31578	1	208	208	yes
141	31578	3	260	100	yes
142	31578	4	303	110	yes
143	31578	7	1187	287	yes
144	33016	3	308	124	yes
145	33016	4	944	650	yes
146	33016	9	5726	3052	yes
147	33016	3	506	341	yes
148	33016	12	4974	218	yes
149	33016	13	3759	1430	yes
150	33016	1	230	230	yes
151	33016	13	1906	505	yes
152	33016	2	376	282	yes
153	33016	2	233	122	yes
154	33016	1	122	122	yes
155	33016	3	300	156	yes
156	33016	3	1287	1119	yes
157	33228	2	188	115	yes
158	33228	1	234	234	yes
159	33228	3	474	194	yes
160	33228	14	2752	698	yes
161	33228	11	25738	11542	yes
162	33228	2	520	367	yes
163	33228	8	1126	244	yes
164	33228	1	88	88	yes
165	33246	2	372	223	yes
166	33246	4	893	325	yes
167	33246	8	10344	5760	yes
168	33246	9	2289	555	yes
169	37889	1	520	520	yes
170	37889	3	499	300	yes
171	37889	3	237	117	yes
172	37889	2	285	147	yes
173	37889	7	9380	4079	yes
174	38102	17	68103	5304	yes
175	38102	3	1165	899	yes
176	38102	5	919	339	yes
177	327	3	1310	842	no

Glitch No.	Exposure No.	Pixels affected	Total Signal [ADU]	Brightest Pixel [ADU]	In SAA
178	2024	4	1657	719	no
179	5021	5	2637	1266	no
180	5272	5	2281	812	no
181	13355	2	332	207	no
182	18907	8	14861	9006	no
183	21957	9	4177	1287	no
184	22490	2	1092	974	no
185	27753	3	617	412	no
186	33694	8	3216	815	no
187	34440	6	9122	5339	no
188	35680	8	5645	3312	no
189	36764	3	1612	802	no

Table A.4.: List of identified glitches in observations of HD 189733 carried out by MOST.

Exposure No.	Latitude [°]	Longitude [°]	Altitude [m]	Magnetic Field Strength [nT]	In SAA	Glitches
924	-18.08	-53.22	824760	16642	yes	7
2359	-17.61	-46.3	831884	16699	yes	7
2363	-7.8	-44.01	831949	17368	yes	7
2585	-31.6	-75.29	831327	18492	yes	9
5614	-7.34	-64.39	831516	18010	yes	5
7005	-29.07	-40.65	825399	17233	yes	12
7222	-29.75	-65.82	825416	17431	yes	11
8445	-14.78	-35.86	831189	17207	yes	13
9836	-21.63	-12.33	825547	18838	yes	4
10054	-20.68	-37.93	825557	17063	yes	12
11699	-2.87	-53.61	830857	18312	yes	4
18012	-15.72	-72.16	830078	17450	yes	4
20620	-14.04	-16.15	829824	18686	yes	9
20629	-28.74	-19.83	829395	18272	yes	3
22657	-32.56	-65.98	826624	17792	yes	8
25482	-34.91	-35.04	826862	17909	yes	3
30198	-32.94	-57.09	828119	17409	yes	10
31578	-21.4	-28.94	827727	17601	yes	8
32010	-26.03	-78.49	827670	18190	yes	7
33016	-19.17	-23.21	828241	18017	yes	13
33228	-10.3	-46.49	828317	17121	yes	8
33246	-39.67	-54.3	827585	18436	yes	4
37889	-11.82	-46.83	828501	16989	yes	5
38102	-19.05	-70.48	828429	17207	yes	3
327	71.21	-13.25	822610	38321	no	1
2024	3.06	175.11	824284	23362	no	1
5021	-72.3	-145.33	826093	40041	no	1
5272	-18.69	159.62	825177	31590	no	1
13355	44.24	-74.97	824072	37048	no	1
18907	-56.78	172.34	827942	42766	no	1
21957	-66.39	169.51	827109	43837	no	1
22490	49.74	-62.56	824777	37303	no	1
27753	54.1	-113.9	825907	39560	no	1
33694	-60.8	-115.19	826686	33807	no	1
34440	36.72	-12.86	826527	29349	no	1
35680	-67.88	-128.77	826442	37579	no	1
36764	-72.45	111.61	826340	42327	no	1

Table A.5.: List of images obtained by MOST during observation of HD 189733. Only images that contain glitches above a certain threshold are listed.

Bibliography

Abdu, M., Batista, I., Carrasco, A., & Brum, C. 2005, Journal of Atmospheric and Solar-Terrestrial Physics, 67, 1643

Baker, K. L. & Moallem, M. M. 2007, Opt. Express, 15, 5147

Beck, T. & Malvasio, L. 2015, CHEOPS Instruments Requirements Specification No. 2 (ESA)

Blanter, Y. & Büttiker, M. 2000, Physics Reports, 336, 1

Borucki, W. J., Koch, D., Jenkins, J., et al. 2009, Science, 325, 709

Chen, H. & Rao, C. 2009, Optics Communications, 282, 1526

Chen, Y. 1987, Cramer-Rao bounds on the accuracy of location and velocity estimations using CCD optical sensors, Tech. rep., Massachusetts Institute of Technology

Croll, B., Matthews, J. M., Rowe, J. F., et al. 2007, Astrophysical Journal, 671, 2129

Delabie, T., Raskin, G., Vandenbussche, B., & De Schutter, J. 2014, Proc. SPIE, 9151, 39

Doyon, R., Hutchings, J. B., Beaulieu, M., et al. 2012, Proc. SPIE, 8442, 84422R

Drossart, P., Hartogh, P., Krause, O., et al. 2013, ESA/SRE(2013)2, 9

Eccleston, P., Swinyard, B., Tessenyi, M., et al. 2014, Experimental Astronomy, 40, 427

EChO Consortium Study Team. 2013, EChO Assessment Study Design Report No. 3 (ESA)

EChO Science Study Team. 2013, EChO Science Requirements Document, 3 No. 2 (ESA)

EChO Team. 2013, EChO Mission Requirements Document, 3 No. 2 (ESA)

Ehrenreich, D., Heras, A., & Isaak, K. 2015, CHEOPS Science Requirements Document No. 2 (ESA)

Esposito, G. 2002, Dissertation, Università Degli Studi di Perguia

Fortier, A. 2015, CHEOPS Noise Budget, 3 No. 4 (ESA)

Greisen, K. 1966, Physical Review Letters, 16, 748

Groom, D. 2004, Astrophysics and Space Science Library, Vol. 300, Cosmic Rays and Other Nonsense in Astronomical CCD Imagers (Springer Netherlands), 81–94

Heirtzler, J. 2002, Journal of Atmospheric and Solar-Terrestrial Physics, 64, 1701

Jacobs, D. A. H. 1977, The state of the art in numerical analysis (Academic Press London ; New York), xix, 978 p.

Jia, H., Yang, J., Li, X., et al. 2010, Science China Technological Sciences, 53, 3145

Kitchin, C. 2003, Astrophysical Techniques, Fourth Edition (Taylor & Francis)

Kuntzer, T., Fortier, A., & Benz, W. 2014, Proc. SPIE, 9149, 91490W

Lee, S. 2002, Proc. SPIE, 4635, 65

Levenberg, K. 1944, Quarterly Journal of Applied Mathmatics, II, 164

Liebe, C. C. 1992, IEEE Aerospace Electronic Systems Magazine, 7, 34

Marquardt, D. W. 1963, Journal of the Society for Industrial & Applied Mathematics, 11, 431

Miller-Ricci, E., Rowe, J. F., Sasselov, D., et al. 2008, Astrophysical Journal, 682, 593

Moré, J. J., Sorensen, D. C., Hillstrom, K. E., & Garbow, B. S. 1984, The MINPACK Project, in Sources and Development of Mathematical Software (Upper Saddle River, NJ, USA: Prentice-Hall, Inc.), 88–111

Nelan, E. P., Lupie, O. L., McArthur, B., et al. 1998, Proc. SPIE, 3350, 237

Nicolle, M., Fusco, T., Rousset, G., & Michau, V. 2004, Opt. Lett., 29, 2743

Nocedal, J. & Wright, S. J. 2006, Numerical Optimization, 2nd edn. (New York: Springer), 256–264

Ottensamer, R., Rataj, M., Schrader, J., **Ferstl, R.**, et al. 2014, Proc. SPIE, 9143, 91434O

Padgett, C. & Kreutz-Delgado, K. 1997, IEEE Transactions on Aerospace Electronic Systems, 33, 202

Patel, J. & Read, C. 1996, Handbook of the Normal Distribution, Second Edition, Statistics: A Series of Textbooks and Monographs (Taylor & Francis), 288–322

Poyneer, L. A., Lafortune, K., & Awwal, A. 2003, Lawrence Livermore National Lab Document, Livermore

Press, W. H., Teukolsky, S. A., Vetterling, W. T., & Flannery, B. P. 2007, Numerical Recipes 3rd Edition: The Art of Scientific Computing, 3rd edn. (New York, NY, USA: Cambridge University Press)

Starodubtsev, S. a. & Usoskin, I. G. 2010, Astronomy Letters, 36, 438

Strasbourg Astronomical Data Center. 2015, NGC 5139, http://simbad.u-strasbg.fr/simbad/

Swordy, S. P. 2001, Space Science Reviews, 99, 85

Thomas, S. 2004, Proc. SPIE, 5490, 1238

Thompson, R. E., Larson, D. R., & Webb, W. W. 2002, Biophysical Journal, 82, 2775

Tinetti, G., Drossart, P., Eccleston, P., et al. 2015, Experimental Astronomy, 40, 329

Tong, Y. 1990, The multivariate normal distribution, Springer series in statistics (Springer-Verlag), 6–21

Toyozumi, H. & Ashley, M. C. B. 2013, Publications of the Astronomical Society of Australia, 22, 257

Transtrum, M. K. & Sethna, J. P. 2012, ArXiv e-prints

Tremsin, A. S., Vallerga, J. V., Siegmund, O. H. W., & Hull, J. S. 2003, Proc. SPIE, 5164, 113

Uhm, T.-K., Kim, J.-Y., & Youn, S.-K. 2008, Journal of the Korean Physical Society, 52, 160

van Assen, H., Egmont-Petersen, M., & Reiber, J. 2002, Image Processing, IEEE Transactions on, 11, 1379

Vyas, A., Roopashree, M., & Prasad, B. 2009a, Proceeding of the National Conference on Innovative Computational Intelligence and Security Systems, 400

Vyas, A., Roopashree, M., & Prasad, B. 2009b, Advances in Recent Technologies in Communication and Computing, International Conference on, 366

Vyas, A., Roopashree, M. B., & Budihala, R. P. 2010, Proc. SPIE, 7588, 06

Vyas, A., Roopashree, M. B., & Prasad, B. R. 2010, International Journal of Computer Applications, 1, 32

Vyas, A. & Vohnsen, B. 2013, Ophthalmic and Physiological Optics, 33, 434

Waldman, I. & Pascale, E. 2013, EChO Pointing Jitter Impact on Photometric Stability, http://sci.esa.int/science-e/www/object/doc.cfm?fobjectid=53425

Walker, G., Matthews, J., Kuschnig, R., et al. 2003, The Publications of the Astronomical Society of the Pacific, 115, 1023

Wernet, M. & Pline, A. 1993, Experiments in Fluids, 15, 295

Widenhorn, R., Blouke, M. M., Weber, A., Rest, A., & Bodegom, E. 2002, Proc. SPIE, 4669, 193

Winick, K. A. 1986, J. Opt. Soc. Am. A, 3, 1809