An expansion to the CHEOPS mission official pipeline

A. Silva
Mestrado Integrado em Engenharia Física
Departamento de Física e Astronomia
2019

Orientador
Prof. Dr. Nuno Santos, Faculdade de Ciências

Coorientador
Prof. Dr. Sérgio Sousa, Faculdade de Ciências

U.PORTO

FACULDADE DE CIÊNCIAS
UNIVERSIDADE DO PORTO

Todas as correções determinadas pelo júri, e só essas, foram efetuadas.

O Presidente do Júri,

Porto, _____/_____/_____

UNIVERSIDADE DO PORTO

MASTER'S THESIS

An expansion to the CHEOPS mission official pipeline

Author:
André SILVA

Supervisor:
Nuno SANTOS

Co-supervisor:
Sérgio SOUSA

A thesis submitted in fulfilment of the requirements
for the degree of MSc. Engineering Physics

at the

Faculdade de Ciências da Universidade do Porto
Departamento de Física e Astronomia

October 16, 2019

" Maybe nothing in this world happens by accident. As everything happens for a reason, our destiny slowly takes form. "

Silvers Rayleigh

Acknowledgements

I would like to start by thanking my supervisors, Nuno Santos and Sérgio Sousa for all the help that they have provided during this thesis and the time they spent helping me with the problems that I found along the way. And, alongside them, every member of the Instituto de Astrofísica, IA, with special mentions for Susana Barros, João Faria, João Camacho, Jarle Brinchmann and Olivier Demangeon for helping me when I needed and always being available to discuss ideas. The last person from the institute whom I want to thanks is Pedro Silva, for all the interesting discussions about our theses, all the rants we made when things didn't work as expected and for accompanying me and Nóbrega during the all-nighters, when such times were needed.

I would also like to thank my parents, grandparents and the rest of my family, that also encouraged me and were always there when I needed. Not only them, but also my friends, who are too many to individually mention, except the most important, my love, Diana. I am forever gratefull for having all of them in my life, and I hope that it continues that way.

Lastly, I would like to thank the Instituto de Astrofísica e Ciências do Espaço (IA) and FCT, for the Scientific Initiation Studentship with the reference A2019-04-BIC which was funded in the context of the project "Towards the precise characterization of Earth-like exoplanets" with financial support provided by the FCT/MCTES (refª IF/00028/2014/CP1215/CT0002)". This work was also supported by FCT/MCTES through national funds and by FEDER - Fundo Europeu de Desenvolvimento Regional through COMPETE2020 - Programa Operacional Competitividade e Internacionalização by these grants: UID/FIS/04434/2019; PTDC/FIS-AST/32113/2017 & POCI-01-0145-FEDER-032113; PTDC /FIS-AST/28953/2017 & POCI-01-0145-FEDER-028953.

UNIVERSIDADE DO PORTO

Abstract

Faculdade de Ciências da Universidade do Porto
Departamento de Física e Astronomia

MSc. Engineering Physics

An expansion to the CHEOPS mission official pipeline

by André SILVA

With the CHEOPS mission launch date set for later this year, in October/November, the official Data Reduction Pipeline (DRP) has been unveiled. In it, the data reduction procedures are applied, alongside with aperture photometry for the target star. However, there is still some untaped potential on the data provided by this mission.

In typical observational conditions, one might find stars in the background of the CCD's images that, due to the satellite rotational movement, are not in fixed positions, but instead move through the CCD.

We believe that is possible to extract data from those stars, albeit the mission being only focused on the targeted star. Thus, in this work, we have built ARCHI, an upgrade to the official DRP. This pipeline is capable of applying the aperture photometry method on all stars and, making use of a background grid, we can improve the noise in the light curves. Furthermore, we tested the viability of using Gaussian Processes to model the noise in the Light Curves, reduce the noise found in them and, at the same time, also determine parameters from the planet that is transiting the star.

In this work we made use of three simulated data sets, and found that ARCHI's light curves had less noise than the ones produced by the DRP and that it is possible to detect transits in the background stars. The application of the Gaussian Processes, allowed us to build models that closely resembled the simulated light curves, and reduce the noise in them.

Keywords: CHEOPS; Gaussian Processes; Exoplanets; Photometry; Planetary systems; Space Mission; Instrumentation.

UNIVERSIDADE DO PORTO

Resumo

Faculdade de Ciências da Universidade do Porto
Departamento de Física e Astronomia

Mestrado Integrado em Engenharia Física

Uma expansão para a pipeline de tratamento de dados da missão CHEOPS

por André SILVA

Com o lançamento da missão CHEOPS marcado para o final deste ano, em Outubro/-Novembro, a "Official Data Reduction Pipeline" (DRP) foi publicada. O objectivo desta pipeline passa pela redução dos dados provenientes da missão, bem como a aplicação de fotometria de abertura sobre a estrela alvo. Contudo, esta abordagem deixa de fora bastante informação presente nas imagens da missão.

Em condições de observação típicas podem existir outras estrelas no fundo da imagem que, devido ao movimento rotacional do satélite, também se encontrarão a movimentar ao longo do tempo. Tendo em conta isto, acreditamos que seja possível extrair informação destas estrelas, embora a DRP estaja focada unicamente no estudo do alvo. Neste trabalho foi desenvolvida uma nova pipeline, ARCHI, que irá funcionar em cima da DRP. O objetivo de ARCHI passa por aplicar fotometria de abertura em todas as estrelas e, com a utilização de uma grelha de fundo, conseguimos reduzir o ruído existente nas curvas de luz. Por último, iremos também averiguar a viabilidade da utilização de Processos Gaussianos para determinar parâmetros dos planetas, bem como modelar o ruído e consequentemente, removê-lo das curvas.

Neste trabalho foram estudados três conjuntos de dados simulados, tendo sido verificado que o ARCHI produz curvas de luz com menos ruído que a DRP e também que os métodos desenvolvidos permitem obter curvas de luz vindas das estrelas de fundo. Os Processos Gaussianos mostraram-se capazes de retornar modelos bastante semelhantes aos dados injetados nas simulações, bem como são capazes de reduzir o ruído existente nas curvas de luz.

Palavras Chave: CHEOPS; Processos Gaussianos; Exoplanetas; Fotometria; Sistemas planetários; Missão espacial; Instrumentação.

Contents

List of Figures

Glossary

CHEOPS	CHaracterising ExOPlanet Satellite
CCD	Charge Coupled Device
DRP	CHEOPS's Data Reduction Pipeline
ARCHI	An expansion foR the CHeops mission pipelIne
GP	Gaussian Process
RA	Right ascension
DEC	Declination
ppm	parts per million
LC	Light curve
CDPP	Combined Differential Photometry precision
SavGol filter	Savitsky-Golay filter

Chapter 1

Introduction

In this project we propose to expand the functionality of the CHEOPS mission official data reduction pipeline (DRP), to maximize the scientific gains from its operation. As we shall see during this Chapter, the DRP can only extract light curves with high precision from the targeted star, without analyzing all others that may be present in the field of observation. Furthermore, due to CHEOPS's mission design, the satellite is rotating throughout the observations and, consequently, so do the stars. The first goal of this work is to be able to extract light curves with the highest precision possible, from the moving stars, in order to find possible candidates for further studies. Our second goal, is to also estimate the noise introduced by the rotations and determine planetary parameters from the obtained light curves.

In this Chapter, we will expose the motivation behind the search of other planets and introduce some of the key concepts applied throughout the thesis.

In Chapter 2 we have a in-depth explanation of the developed pipeline: "An expansion foR the CHeops mission pipelIne - ARCHI", in which all of the different components for the photometric portion are discussed and, when possible, compared against the official pipeline. Following, we have Chapter 3, where we will approach the thematic of Gaussian processes and their role in the pipeline for the estimation of planetary parameters.

Afterwards, in Chapter 4, we have a small discussion on the implementation of the previously described modules, alongside a (basic) memory consumption analysis of the algorithm.

Lastly, in Chapter 5, we present the obtained results and compare them against themselves to find the impact of each component on the final light curve, followed by a discussion on a sinusoidal signal, with period equal to half of the satellite's rotational period, found on some of the light curves of the background stars. The Gaussian processes are then also discussed, alongside their results and improvements that can be made upon them. To finalize Chapter 5 we shall also compare ARCHI's light curves against the ones from the DRP.

1.1 The hunt for exoplanets

The humans have long since started to look into the sky, trying to unveil the mysteries of the cosmos, starting with Galileo in 1609, with him seeing, for the first time in human history, the surface of the moon using a telescope. Since then, the human race has been on a quest to explore the unknown, in hopes of finding other habitable planets and/or life.

It was in 1995 that Michel Mayor and Didier Queloz published their findings of an exoplanet orbiting a solar-like star, *51 Peg*, which was a groundbreaking achievement [1]. To accomplish it, small Doppler shifts were observed in the spectrum of a star, i.e. the star was oscillating around a point, alongside with another object.

However, there were doubts of the validity of this discovery, due to the characteristics of the (supposed at the time) planet. Another explanations were provided and community was somewhat split on this topic. In the following years, further studies solidified the finding of this planet, and others were also found using the same method [2].

The interest for exoplanets started to grow in a meaningful way right before the turn of the millennium and, in 2010 alone, they were the topic of more than 1000 published papers [3]. This rise in popularity was not only evident among the scientific communities, but also within the general public, through programs that allowed a citizen to be a part of various projects.

In the early stages of exoplanet detection, until right before the turn of the millennium, the only efficient method was through observations of small variations of radial velocities. However, that changed with the detection of planetary transits, which is now the most efficient method that one can use, due to the results from the Kepler mission [4].

Contrasting earlier days, we now have specific missions dedicated to it, such as Kepler [5], TESS [6], TRAPPIST [7], and others planned for the near future, such as PLATO [8].

The search for habitable planets is a never ending quest, with many uncertainties. Currently, we are searching for other habitable worlds using as a basis what we already know, i.e. Earth. We have created a set of criteria that must be met for a discovered planet to be classified as habitable [9], thus giving a new goal for the search of exoplanets.

1.2 CHaracterising ExOPlanet Satellite - CHEOPS mission

CHEOPS mission is the first dedicated mission to better characterize planetary transits with ultra-high precision photometry on stars known to have transiting planets. It is one of ESA's mission, set to launch later this year, in October/November.

To accomplish the goal of ultra-high precision photometry, this mission is required to reach a photometric precision of 20 ppm for Earth sized planets transiting G5 dwarf stars with V-band magnitudes in the [6 - 9] mag range over 6 hours of integration time. For larger planets, Neptune-size, in transiting K-type dwarf stars it should reach 85 ppm over 3 hours of integration time [10].

The stars to be studied will come from both ground-based surveys, space-based surveys, such as TESS, and the community proposals for the available open time, thus revealing synergy with other, already existing, missions. However, it's important to note that the available targets are limited by the Sun, Moon and Earth itself, as seen in Figure 1.1, due to both the scientific requirements for the mission and its low altitude orbit.

The typical orbit during the mission operation is a circular Sun-synchronous orbit, at an altitude of 800 km and a rotational period of approximately 100 minutes. The spacecraft is nadir locked and thus it will always be rotating around Earth, pointing towards the targeted direction. This orbit configuration results in the rotation of the background stars, i.e. those that are not the target, instead of them being fixed in place.

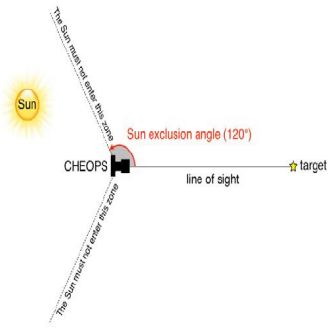


FIGURE 1.1: Illustration of the pointing constraints. Image taken from [10].

The South Atlantic Anomaly, SAA, is a region in which the inner Van Allen radiation belt is closest to the Earth's surface, and thus the satellite receives higher doses of radiation [11]. In practical terms, this leads to sudden spikes in the light curves [12]. To avoid data

contamination due to this phenomenon, all images captured by CHEOPS during it will be discarded, which in turn leads to the possibility of finding time lapses within the light curves.

The mission makes use of a frame transfer CCD with a 1024x1024 pixel grid but, only a part of the image is used. The default area of interest is a 200x200 pixel region, that can be centered on the desired area of the detector [10], which is used to save bandwidth during the data transfer from the spacecraft to Earth. To the entire region of the CCD the name of *FullArray* is given, while the smaller portion of the CCD is named *SubArray*.

The CCD's focal plane was also made in such a way that we get a large Point Spread Function (PSF), i.e. a large bi-dimensional profile that represents the light distribution, thus facilitating the Flat Field correction. Due to jitter of the spacecraft, the PSF will not always occupy the same CCD pixels but instead will shift between the nearby ones, thus introducing noise due to pixel-to-pixel variations. In some cases, to avoid saturation, the sub array can be made of a stack of images with shorter exposure time [13], which will then be used to create a "final" image.

During a visit we can have exposure times in the range of 1 to 60 seconds. Each image, will actually be a combination of individual exposures, whose number is automatically set taking into account the desired exposure time [14]. Due to limitations on the amount of data that can be transfered, all of the individual exposures are co-added, pixel by pixel, on board and then sent to Earth. However, to avoid loosing too much information on the target star, we have *imagettes* for each individual exposure. They are a small circular cutout of the image, containing only the PSF of the target star and are used in the Data Reduction Pipeline, henceforth referred to as DRP, for correction purposes [14].

For further details on the missions scientific requirements and payloads, one should refer to [10, 14, 15] and for information on the outputs of the pipeline, and the data inside each file, we refer to [16]. Lastly, since the mission is yet to see its first light, all obtained data sets are simulated, using CHEOPSim [17], the official simulation tool.

1.2.1 Official Data Reduction Pipeline

Taking into account that this project was built as an extension for the CHEOPS mission DRP, we must understand what it can do. However, to have a deeper understanding of this pipeline, as well as the photometric analysis, one should refer to the article in which it is introduced [13]. It's important to keep in mind that the DRP is still under development

and thus we sometimes find small problems in the data storage procedures, as we shall see later on, in Chapter 5.

The DRP can be thought of as a collection of 3 different modules, that are applied in a sequential order, as presented bellow:

1. *Calibration*: Corrects the instrumental response, by attempting to remove:
    - Bias and readout noise;
    - Non-linearity of the CCD;
    - Gain;
    - Dark current;
    - Flat Field.
2. *Correction*: Corrects environmental effects, such as:
    - Smear correction;
    - Bad pixels;
    - Background.
3. *Photometry*: Extracts the targeted star's light curve from the images.

Since a deeper analysis of the first two modules is outside the scope of this document, we only listed their contributions, without further discussion.

We can now look into the *Photometry* module with greater detail, to understand the methodologies in use. The flux from the target star is calculated with a circular binary mask. In order to maintain the same number of pixels in the mask, it is created once and, afterwards, shifted with an anti aliased shifting algorithm. If we used a non-constant mask, then we would be introducing photometric noise in the light curve. The mask aperture is optimized through the maximization of the signal to noise ratio. This pipeline, when used always outputs 4 different light curves:

- *DEFAULT*: Uses a mask with a default radius of 33 pixels;
- *OPTIMAL*: The mask uses the optimized radius;
- *RINF*: The mask has a radius equal to 80 % of the default mask size;
- *RSUP*: The mask has a radius equal to 120 % of the default mask size.

In order to always have the mask centered over the central star, an iterative Gaussian apodization method is used to estimate the star's position in each image. This algorithm starts by removing, from the images, contributions from nearby stars and from pixels changing due to the jitter, for a mask placed on the point provided by on-board software. Afterwards, the center of light, from this corrected image, is calculated, and a new mask is centered in it. This process then repeats until a convergence criterium is met, which tends to occurs fairly quick, under 20 iterations. This process is capable of estimating positions with errors as low as $2x10^{-3}$ pix, as reported in [13].

The calculation of uncertainties in the light curve, is made through Equation 1.1:

$$\begin{aligned} Err &= \sqrt{Flux + bg + N_{pix} * N_{stack} * (gain * ron)^2 + dark * t_{exposure} * N_{pix}} \\ bg &= background * N_{pix} * t_{exposure} \end{aligned} \tag{1.1}$$

where Err is the uncertainty for a given point, Flux the corresponding point, N_{pix} the number of pixels inside the mask, dark is the dark current, ron is the read out noise, $t_{exposure}$ is the exposure time for each image, N_{stack} is the number of stacked images, *background* is the flux from background objects and, lastly, *gain* is the gain from the digital conversion process that occurs in the CCD.

1.3 Small introduction to astrophysics

In this section a small overview of the coordinate system in use, as well as the parameters used to characterize a planet's orbit shall be presented and discussed.

1.3.1 The Equatorial Coordinate system

As a way to properly track celestial objects, independently of the observers position and time of observation, one can make use of this coordinate system. Since the equatorial plane of Earth remains approximately stable during Earth's rotation, we can use it as a reference plane. We can measure the angular separation between the object and the reference plane, naming it **declination**, also referenced as δ or DEC.

However, we still need one other point of reference, to have two parameters to track the object. The vernal equinox, is where the sun crosses the celestial equator, and it can be used as a reference. If we measure the counterclockwise angular separation along the equator, we get the **right ascension**, or α or RA. One can also refer to Figure 1.2, in which

the coordinate system is schematized. Even though the reference points were said to be fixed, in reality they are not. Due to the precession of Earth's axis, the references suffer a shift of 0.01 degree per year. To counteract this, when using this system, one has to know the time at which the measurements were made, to be able to find the object's actual position.

The angular separations can be represented in terms of arcminutes (arcmin) and arcseconds (arcsec). As the name might suggest, one arcminute is equal to $\frac{1}{60}$ of an angle, while an arcseconds is equal to $\frac{1arcmin}{60}$.

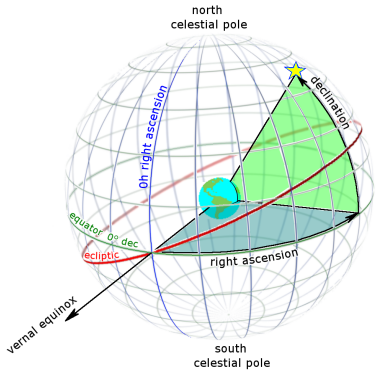


FIGURE 1.2: Schematic of the right ascension and declination for a star. Image taken from [18].

Now that the basics of the Equatorial system has been laid out, we can now calculate the distance between two points, which is not as trivial as it might seem. In the declination case there is no problem, since the distance between the two points is the same, independently of where they both are in the sphere.

However, that is not the case for differences in the right ascension. If we calculate the difference of RA between two points near the equator and near the poles, we find different values in both cases, with the bigger difference near the equator. Thus, to calculate this difference, we have to map the stellar sphere into a flat map. To do it, we can use of one the many techniques described in [19]. However, we shall calculate the distance using the *Great-circle distance*, with Equation 1.2, derived in Appendix A.

$$Diff_{RA} = \Delta RA * cos(Average_{DEC}) \tag{1.2}$$

1.3.2 Planetary orbits

We have three laws, that describe the motion of planets around its Sun, most known as Kepler's laws:

- First Law: The orbit of a planet is an ellipse, one focus of which is in the Sun;

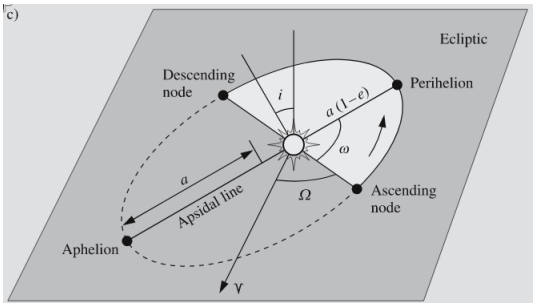


FIGURE 1.3: Illustration of the orbital elements, taken from [20].

- Second Law: A line segment joining a planet and the Sun sweeps out equal areas during equal intervals of time;
- Third Law: The square of the orbital period of a planet is directly proportional to the cube of the semi-major axis of its orbit.

Now that we know that the orbit is elliptical, we still need to know how to be able to describe it and how to locate the planet in it. Thus, we shall introduce six parameters, henceforth named Orbital elements, that are capable of fulfilling that role.

1. Semi-major axis - a: Half of the ellipse's major axis;
2. Eccentricity - e: Measures the deviation, of the object's orbit, from a perfect circle. For a value of 0 we have a circular orbit, for values in the interval]0,1[we have an elliptic orbit, for a value of 1 we have a parabolic escape orbit and, greater than 1, a hyperbola.
3. Inclination - i: obliquity of the orbital plane, relative to some fixed reference plane;
4. Longitude of the ascending node - Ω: Where the object crosses the ecliptic, from south to north, measured counterclockwise from the vernal equinox;
5. Argument of the perihelion - ω: direction of the perihelion (point in which the object is closest to the sun) , measured from the ascending node in the direction of motion;
6. Time of the perihelion - τ: time in which the object crosses the perihelion.

For a more visual comprehension of the different parameters, one can look at Figure 1.3, taken from [20].

Kepler's third law, can be written as Equation 1.3:

$$T^2 = \frac{4\pi^2}{G(M+m)} * a^3 \tag{1.3}$$

, where T is the planet's period, G is the gravitational constant, M is the star's mass, m the planet's mass and a is the semi-major axis. With this, we can determine the semi-major axis of the orbit, assuming that the orbital period and the star's mass are known. As an approximation, we can discard the planet's mass, since it is usually much smaller than the one from the star.

1.4 The Transit method

As mentioned in Chapter 1.1, the *transit* method is one of the most effective methods for exoplanet detection.

This method, when combined with observations of radial velocity, which let us determine the planet's mass, allows astronomers to estimate the radius of the planet. However, as we shall see, it's also needed accurate estimates of the mass, radius and limb darkening of the star, to properly analyze the transit.

1.4.1 Fundamentals

When a planet passes in front of a star, as schematized in Figure 1.4, the measured brightness of the star suffers a dip in its value due to the partial occultation of the stellar disk.

It's possible to make an educated guess for the order of magnitude of this dip, assuming that both the planet and the star have spherical shapes and that the planet's flux is negligible [21]:

$$\Delta F \approx \left(\frac{R_{planet}}{R_{star}}\right)^2 \tag{1.4}$$

where ΔF is the transit's depth.

From Equation 1.4 we can also see that an increase of this ratio is better for the detection of the transit in the light curve, since a larger fall in the flux's values is easier to detect than a smaller one. Another important detail that one can take from this Equation, is that the method can only provide precise information in cases in which the star's radius is known.

Although the detailed modeling of the signal produced by a transiting planet is complex, we can still describe and interpret it in simple terms. During the phase in which

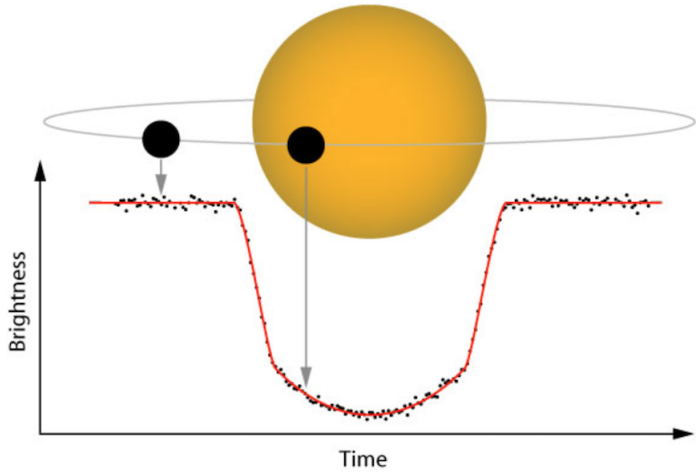


FIGURE 1.4: Schematic of dip in the flux, due to the passage of a planet in front of the star. Image taken from [22].

the planet is entering the "frontal" part of the star we can notice a steady decrease in the flux. This decrease reaches a minimum when the system detector - planet - star are completely aligned. After this halfway point, the flux starts to return to the baseline value. The duration of the transit, τ_F is defined as the amount of time that the entire planetary disk is in front of the star. As we shall see later on, with greater detail in Chapter 1.4.3.2, the star-disk, from our line of sight, is not uniform but, instead it is fainter near the edges of the star. This phenomenon, also referred to as limb darkening, impacts the light curves near the beginning and end of transit, when the planet is passing by those fainter areas, thus explaining the curved shape near the mid-transit area.

During the transit, we can also define the *impact parameter*, i.e. the minimal projected distance to the center of the stellar disk, as given by Equation 1.5.

$$b = \frac{a}{R_s} cos(inc) \tag{1.5}$$

Lastly, from [23] we know that the probability of finding a transit is given by Equation 1.6

$$p_{transit} = \left(\frac{R_{star} \pm R_{planet}}{a}\right)\left(\frac{1 + esin(\omega)}{1 - e^2}\right) \tag{1.6}$$

in which "+" accounts for grazing transits, i.e. when the planet is never entirely inside the star's disk, and "-" excludes them.

1.4.2 Problems

One of the worst problems that we can encounter are false positives, i.e. the transit may not be a result of a planet's passage, but instead be caused by another astrophysical, or instrumental, phenomena. As an example, eclipsing binaries, i.e. two stars rotating around their center of mass, that have their light diluted by a nearby bright star, or even eclipsing binaries observed directly [24] can mimic a planetary signal.

In order to validate the transit, we can check if it appears on several observations of the same star, since false positives have very low probabilities of occurring if several transiting signals are found [21].

Although this issue poses some serious problems, it falls outside the scope of this thesis. However, to better understand what causes them, one can refer to the many articles on the topic, e.g [25–27].

Another limitation of the photometric method is that it depends on the planet being aligned correctly with the observer, i.e. the orbital plane of the planet must be perpendicular, or almost perpendicular, to cross the stellar disk in our line of sight, as well as performing the observations at a time in which the transit occurs.

1.4.3 Noise sources

There are many noise sources that can negatively impact the obtained results, such as instrumentation noise, stellar activity, stellar granulation and oscillations [28]. In particular, stellar activity can lead to both modulation of the flux, as well as in-transit fluctuations that will posteriorly affect the determination of the transit's depth.

1.4.3.1 Stellar spots

Active stars typically have dark and/or bright spots on their surface that can induce changes to the spectral line shape [29] and thus hamper precise measurements.

These spots appear in places of the stellar photosphere in which intense concentrations of magnetic flux erupt from. With the star's rotation the spots are shifted until they eventually decay due to turbulent diffusion. Stars with higher rotational speeds tend to be more active and thus have larger spots, which take longer to decay due to diffusion [30].

From [31] we know that the dark spots can impact the light curve in two different ways: The spots not occulted by the planet lead to a decrease of the flux outside the

transit and, the occulted ones lead to an increased flux. So, star spots also introduce out of transit flux modulation over the timescale of stellar rotation period [32]. On the other hand, bright spots, also known as *faculae*, also affect the light curves in the same manner, but with reversed effect [29], i.e. when they are not occulted we see a flux increase. Aside from the changes in the flux level, there is also a possibility that they can induce transit-shaped alterations on the light curve.

Stellar spots can influence our measurements, such as the planet's radius [30], the host star limb-darkening coefficients [33], as well as the orbital inclination [31]. In [33] planet radius 4% smaller than the real one were estimated, as well as changes of approximately 4% on transit duration were found, due to the anomalies in the light curve.

Typically light curve normalization occurs with the data before and after the transit, which means that the spot contributions will enter the normalized light curve, thus affecting the profile shape or, in some cases, depth of the transit [34].

1.4.3.2 Limb darkening

As seen from our line of sight, the star-disk is not uniform, but is instead brighter in the middle and fainter near the edge [23]. Near the edges of the star, the line of sight enters at a small angle and we are only able to see light from the cooler higher layers, resulting in a solar disc that appear darker near the edges [20]. It lead to an increased difficulty in the determination of the starting and final point of the transit. This phenomenon, called limb darkening, has been a target of many studies and models of it have been created [35–38].

To properly understand and characterize this phenomenon, one must observe the intensity distribution over the stellar disk, which is doable for the Sun but not for Stars further away from Earth [39]. For those kind of stars, one must rely on indirect methods, i.e. determining the coefficients with the help of light curves, over which the models are fitted.

The effects of limb darkening on the light curves are evident in Figure 1.4, in which we see the a gradual decrease and, posteriorly, increase of the flux near the beginning and end of the transiting event. That behavior appears due to the non-uniform brightness of the star near its edges, i.e. when the planet is closer to the edges it blocks less flux than when it is near the center, thus leading to a non-uniform dip in the flux, with the shape of the dip depending on the limb darkening law determined for the star.

1.4.4 Aperture Photometry

Now that we understand the principle behind the method, we still need a way to apply it to images that come from a CCD. Aperture photometry can be condensed into two simple steps [40]:

1. Find the pixels that have light from the star and sum the counts in them;
2. Estimate the background level, i.e. what we would read if the star was not there and, afterwards, remove it from our data.

To perform the first point of this list, we must delimit a region in the sky that contains the greater part of the observed radiation from the object under study [41]. The most basic aperture tends to be a circular one, but others could be used, as we shall see later on this thesis.

The second point is not as simple as it may appear to be. Many methodologies have already been developed to solve it, such as [42, 43]. This step is of utmost importance, since it will try to account for the contributions of thermal radiation, background emissions, cosmic radiation and contributions from the detector itself [44]. Fortunately, the CHEOPS mission official pipeline already takes care of this step, as we have seen in Chapter 1.2. Thus, this topic of background estimation, shall not be discussed in greater detail.

1.5 Overview of Bayesian statistics

Bayesian methods allows one to incorporate previous knowledge and opinions into the analysis, thus being more intuitive to us than frequentist approaches [45]. In a frequentist point of view, we assume an hypothesis as true and try to find the probability of validating it. On the other hand, in a Bayesian mindset, we observe the existing data, trying to assess the likelihood of the hypothesis being true.

If we go by a more formal approach, Bayesian inference is the process of fitting a probability model to a set of data and summarizing the result by a probability distribution on the parameters of the model and on unobserved quantities, such as predictions for new observations [46]. Within this framework, we specify a prior distribution, which is created with previous information available and our beliefs about the parameters before looking at the observations. The parameters of this distribution are called *Hyperparameters* and

thus we can distinguish them from parameters of the model for the underlying system under analysis.

If we have a model, we can also define a likelihood function, which is the probability that the observations follow from a given model evaluated at a given point in the model parameter space [47].

We can now calculate the posterior probability $Pr(H|D)$, for the hypothesis H given data D, with Equation 1.7, derived from Bayes' theorem. By using the prior probability of the hypothesis and data, Pr(H) and Pr(D), respectively, and the probability of data given the hypothesis, $Pr(D|H)$ we reached the probability distribution of the unobserved parameter, conditioned on the existing data.

$$Pr(H|D) = \frac{Pr(H)Pr(D|H)}{Pr(D)} \tag{1.7}$$

When comparing two different models, we have to take into account the marginal likelihood, which is obtained by integrating the posterior over the parameter space [47], giving us the probability of observing the data, given the model. Higher marginal likelihoods are connected with better models.

Typically, Bayesian inference is based on random samples from the posterior. However, if we are not able to sample directly from the posterior, we can estimate values. For that, we can use *Markov Chain Monte Carlo*, henceforth MCMC. Further details on the theory behind then method are left for other works, for example [48, 49], but a quick summary of the method, alongside a brief discussion on how to use them are presented in Chapter 3.1.3.

In practical terms, this method starts a chain, in a location inside the parameter space. Afterwards it proposes a new point, which is accepted or rejected, depending on the posterior density. The proposals are made in such a way that convergence for the posterior distribution is guaranteed, albeit in some situations it may take longer to achieve it [47], depending on the complexity of the model.

Chapter 2

The Photometric analysis

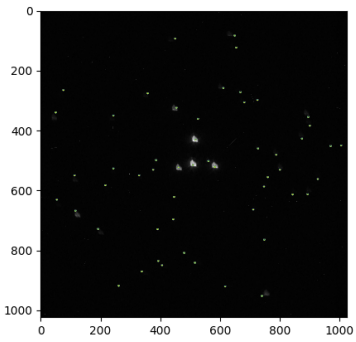
As we have mentioned, the official pipeline was built to extract the light curve of the central star, with the highest possible precision. The goal of this Chapter is to describe a method capable of detecting the other stars present in the field of observation, extracting the best possible light curves from them, which can then be used to search for planets in them, thus allowing us to extract more data from the CHEOPS's mission.

In this Chapter the methods used to determine the stars initial positions, the masks creation, the center tracking routines, the noise metric and a method to reduce the noise in the light curves shall be presented and discussed.

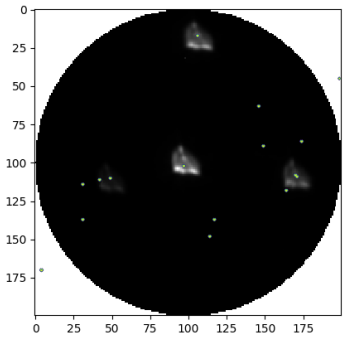
2.1 Initial detection of the stars

The first challenge we faced was how we can detect all the stars in the field. To accomplish this, two very different methods were devised and implemented - one based on the previously known *RA* and *DEC* of each star, and another built on top of image treatment techniques. For simplicity sake, the first shall be named as the **fits** method and the second as the **dynam** method.

Lastly, we shall set a convention to name each detected star: they are named with a zero-based numbering notation considering the distance, in pixels, from the star's centroid, in the first image, to the central point of the image. Thus, the lowest indexed star refers to the target and, the highest one to the furthest star from the target.



(A) Logarithm with base 10 of the first image, with all the estimated centroids, from the *StarCatalogue* file.



(B) First image of the data set, in the *subarray* area.

FIGURE 2.1: Initial star detection for all the stars using the *fits* method.

2.1.1 The fits method

As we have previously seen, during the DRP operation, the target star position, inside the CCD, is calculated. However, there is no direct information on where the background stars lie within the CCD.

Instead, from existing stellar catalogues, we know their right ascension and declination, alongside the scale of the images and the rotation angle of the satellite. If we calculate the difference of RA and DEC between the target star and each non-target star, by applying Equation 1.2 and convert it to pixels, we know how far away each star is from the center.

If the distance in pixels is known, then it's trivial to calculate the actual position of each star: we sum the distance to the central point coordinates. However, there is still a last detail that we have to take into account: the RA and DEC were calculated for a CCD with a *roll angle* of zero, which is not guaranteed to happen on the first image of the data set. Thus, we have to rotate the points by an angle of $360 - \Theta_{initial}$ to arrive at the correct centroids in the initial image, where $\Theta_{initial}$ is defined as the rotation angle of the satellite for the first image in the Data Set. For the central star, we can use the values from the official pipeline, taking into account the low errors reported on the centroid detection, in Chapter 2.5.

We can now test this method in a simulated data set, for both the *FullArray* and the *SubArray*. However, before discussing the effectiveness of the method, we can see that for both cases, Figure 2.1, we have more detected points than actual visible stars.

1 pixel is approximatelly 1 arcsec.

Those points appear due to the presence of very faint stars, almost imperceptible in the normal image. However, if we apply a logarithm with base 10 to the image, as in Figure 2.1a, we can barely see some of them.

In the *FullArray* image we see that near the edges, the estimated points start to deviate from the actual position of the star. However, near the center we do not see these differences from the expected locations. Such deviations may be caused by uncertainties introduced by both the rotation of the points, the calculation of the difference in RA, that makes use of small angle approximations and the approximations of the pixel scale. Those assumptions may thus not hold for those specific points, due to their distance from the center.

If we now look closely at the *SubArray* area, in Figure 2.1b, we also notice the existence of more estimated points than stars.

Thus, we have a pool of candidates from which we need to weed out the estimations for the visible stars. This is done by first checking if the determined coordinates are within our image region. The catalogue gives us the position of each star inside CHEOPS's *FullArray*, even those outside of the *SubArray* area, thus we need to validate if the coordinate in the x and y axis are within the value range of [0, 200]. If this first condition is met, we filter them by their magnitude, i.e, we limit the points of interest for those correspondent to a star with a magnitude inferior than 13 mag.

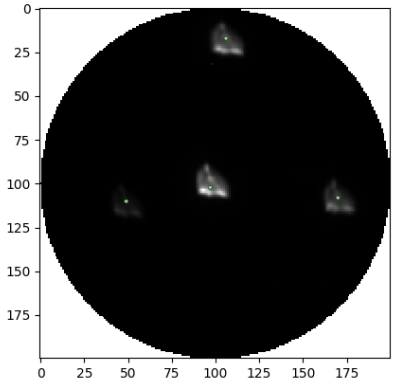


FIGURE 2.2: First image of the data set, with the filtered points from the *fits* method.

By applying the filters to the image we arrive at Figure 2.2, in which only the points of interest still remain, thus validating the method.

2.1.2 The dynam method

Even though the *fits* method yielded good results and managed to estimate the initial position of each star, it can still fail or give imprecise estimations.

Assuming that the default *SubArray*, with a size of 200x200 px is in use.

There are many approaches that one can use to track a moving object in images, but feature-based approaches tend to be more robust ones. This approach will be built around image moments, which we can think about as an weighted average of the pixel's intensities that can be used to determine the image's area, centroid and even some information on the image's orientation. A further discussion is beyond the scope of this project, but an interested reader can refer to [50].

Using both the zeroth and first degree moments we can estimate the centroid of any given shape, by applying Equation 2.1, in which X_c and Y_c are the coordinates of the image's centroid.

$$\begin{aligned} X_c &= \frac{m_{10}}{m_{00}} \\ Y_c &= \frac{m_{01}}{m_{00}} \end{aligned} \tag{2.1}$$

Such algorithm can be easily implemented using Python's *OpenCV* wrapper library, as seen in [51].

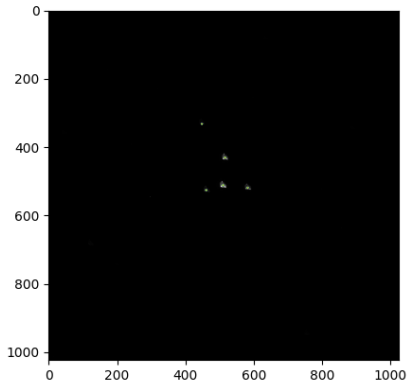
Before we can apply it, we have to perform some pre-processing steps to the images, so that they match the desired functions inputs.

In the first place, we must convert the data type of each pixel, to a more suitable format for the *OpenCV* library, that shall be used to the apply the image treatment techniques. We can convert them to various formats but, the easiest one to convert to, is from one in which each pixel has 16 bits, to one in which each pixel has 8 bits. It's straightforward to see that this data type conversion is accompanied by a reduction in the image resolution, since each pixel now holds less information.

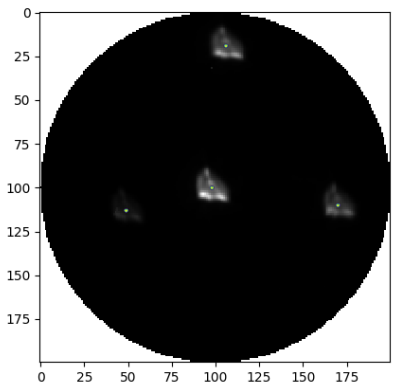
Now that we have our image in the desired data type, we proceed to apply a binary threshold to the image, as given by Equation 2.2, so that the stars are represented by a value of MaxValue and the background a value of zero, thus facilitating the next step in the process: finding the contours.

Both the contour detection and the moments calculation are handled by the *OpenCV* library. Lastly, we only have to worry about false positives in the contours. Due to small imperfections in the images, that may still pass through the data conversion type, we can have a detection of small contours. The contours returned by *OpenCV* are comprised of a list of coordinates, thar mark the detected edge. So, as a way to avoid the usage of

https://opencv.org/. Accessed: 28/8/2019.



(A) First image of the data set, with the points given by the *dynam* method.



(B) First image of the data set, in the *subarray* area.

FIGURE 2.3: Initial star detection for all the stars using the *dynam* method.

those false detections, the contours with less than 5 points are filtered out, since we do not expect to be able to detect and track (in a reliable way) such small objects.

$$I_{final}(x,y) = \begin{cases} MaxValue, & \text{if } I_{original}(x,y) > threshold. \\ 0, & \text{otherwise.} \end{cases} \tag{2.2}$$

Looking at Figure 2.3a, we immediately notice the lower number of determined points inside the image, when comparing against the *fits* method. From Figure 2.1a, we know that we have faint stars, outside the *SubArray* region, whose positions are easily given by the *fits* method, since it changes the position of the star in the sky to the expected position of that point, whitin the image.

However, when using the *dynam* method, we cannot find such faint stars, due to using a image with lower resolution, as discussed. This decrease in the resolution, alongside the threshold, are the most probable causes for the inability to detect the fainter stars.

If we now focus solely on the *SubArray* region, shown in Figure 2.3b, we see that the detected points appear to be within the expected region of the real centroid, thus allowing us to conclude that the method is working as expected.

Lastly, in the *SubArray* area, we managed to detect stars in the [9.21 mag; 11.1 mag] range, seen in Figure 2.3b. Furthermore, in the region outside the *SubArray* we detected a star with 12.3 mag. A possible way of being able to detect more of the fainter stars, could be by using the logarithm of the image, instead of the image, to apply the algorithm.

2.2 Star tracking

Following what was previously discussed, the satellite's CCD is not always in the same orientation but instead, it's rotating, alongside the satellite. To properly study the background stars, we need to be able to track them consistently between images. To do so, three different methods were devised: *static*, *offsets* and *dynam*, as we shall see from now on.

2.2.1 Static method

The most straightforward solution to this problem is to simply rotate the initial centers of each star by the change in the satellite's *roll angle* between two consecutive images. As the difference between consecutive images *roll angle* is not constant, but instead is sinusoidal, seen in Figure 2.15, we have to calculate the rotation angle for each image. The consequences of this behaviour shall be addressed later on, when comparing the three methods, in Section 2.5.3.

In order to calculate the updated point for a given image, P_{image}, we multiply the initial point, $P_{initial}$, by a rotation matrix with an angle, $\Delta\theta$, equal to the difference between the current image's *roll angle* and a previously chosen reference point, as depicted on Equation 2.3.

$$P_{image} = P_{initial} * \begin{bmatrix} cos(\Delta\theta) & sin(\Delta\theta) \\ -sin(\Delta\theta) & cos(\Delta\theta) \end{bmatrix} \tag{2.3}$$

If we set our reference as the previous image, we could be slowly introducing more and more errors, in a "snowball effect" that would slowly increment the difference between the estimated center and the real position. Other alternative could pass by using the rotation angle measured at the beginning of the visit, and calculate the changes taking the star's initial positions as the reference, in an attempt to minimize this effect.

The analysis of a small data set, with 300 images, reveals an average difference with a magnitude of 10^{-15} between the points obtained while using the initial point as the reference and the ones with the previous image as a reference, thus revealing that both of the options could be used. Thus, we shall set our reference as the angle for the first image in the data set.

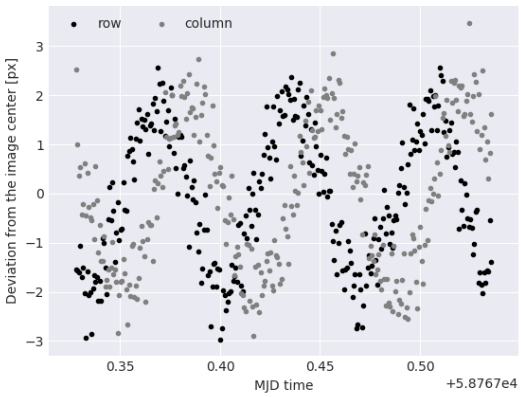


FIGURE 2.4: Deviations from the central point in the image, [100,100], for the target star.

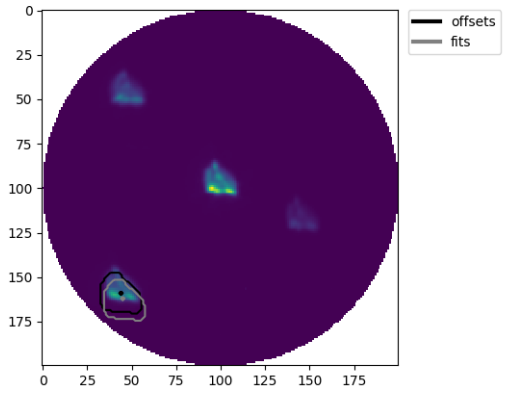


FIGURE 2.5: Comparison of the center obtained with the *static* method and the one corrected with the calculated *offsets* for one of the stars.

2.2.2 Offsets method

If we decide to see how the DRP's centroids change over time, Figure 2.4, we notice some significant changes. Since the *static* method uses the target position to rotate the background stars, in an attempt to improve it, we decided to use the changes in the DRP's centroids, to correct the estimated points.

So, for each image we calculate the change in the central star coordinates from the current image, in relation to the initial position, rotate the background stars centers, using the *static* method and, add the differences found in the central star coordinates to the background ones, in an attempt to correct the jitter.

However, when applying the corrections to the central star, it would almost mimic the DRP's points. So, for simplicity sake, we shall use the DRP's coordinates as the coordinates for the central star. This will also allow us to see how a lower error in the centroid's detections impact the overall quality of the light curves.

Lastly, in Figure 2.5, we can see the impact of the corrections on the points estimated with the *static* method. Even though the offsets corrections managed to improve the center detection, we can still see that the detected point is not yet in what we could call the optimal position. However, we can conclude that this improvement of the *static* method was successful despite not being the optimal alternative.

2.2.3 Dynamical method

Following the good results yielded by the **dynam** method for the initial detection of the stars, exposed in Chapter 2.1.2, one can use it to keep tracking the stars throughout the images.

In this iteration the centers are found with the same technique, although now with a small inconvenience: the association of each center with the corresponding star is not trivial. This time we can't use the distance to the center to number the stars. For example, a case in which two stars are at approximately equal distances from the center could lead to a situation in which the rotation was such that the distances "switched" and thus we could map the newly determined position to another star.

Thus, if we are not careful, the centers can switch between stars (in extreme cases). The solution to this problem passed by building a validation system, capable of estimating the next position for each star and, afterwards, matching the determined positions to the corresponding star. By applying the *static* star tracking method, we can predict the coordinates of each star, for the current image. After detecting each star position, we search the predictions, to find the closest prediction to the determined position, thus mapping a center to a star.

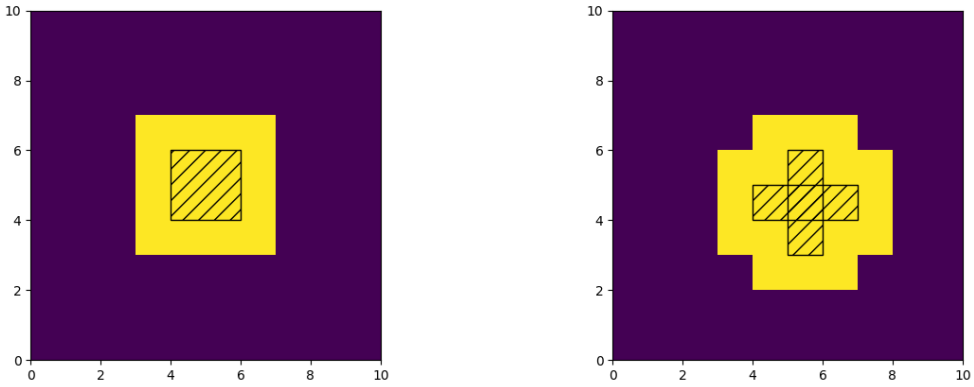
2.3 Masks

After discussing the initial detection of each star and the tracking methods, we shall now look into another essential piece of this pipeline: the masks. In this project, there are two different masks: one is a simple circular mask, similarly to the one used in the DRP, while the second one is based on the shape of the stars.

2.3.1 Circular

The first mask used was equal to the one of the official pipeline: a circle centered in each star's centroid. The circular mask was preferred over other basic geometric shapes due to the fact that the quasi-triangular shape of the star, seen in e.g Figure 2.16, could be easily accommodated inside a circle. The choice of a circular mask also brings the advantage of being straightforward to change the size of it, by just changing the circle's radius.

Other geometric shapes could be chosen such as a square mask or even a triangular one. If we went with a square mask, then it would span over more background area than



(A) Shape increase method applied to a small box (black box), that was increased by one pixel.

(B) Shape increase method applied to a "plus" sign (black box), that was increased by one pixel.

FIGURE 2.6: Examples of the shape increase method for two different structures.

the circular one, which was not ideal. Likewise, a triangular mask could offer a superior choice for the stars away from the edges of the image but, the closer we are to it, the worse fit the mask would be, since we would be extremely limited to expand in one of the directions and would probably result in a lot of masks out of bounds. Another problem that we could run into with a triangular mask, would be how to specify the size of each side and the type of triangle used.

2.3.2 Shape

After using image processing to both determine the initial position of each star and to follow the "moving" stars, we saw that we could also utilize the star's shape to create a mask. By using the process previously discussed, in Chapter 2.1.2, we can retrieve the shape of each star, thus allowing us to build a mask with it. In order to avoid having a mask with a varying number of pixels throughout the images, it is calculated using the first image of the data set.

However, a problem appears: we now need to find a way to change the mask's size, so it can be optimized. Unlike the circle mask, in which it's straightforward to increase its size, we now need to find a way to increase it and to quantify such increase.

To accomplish this, one can simply add layers of pixels around the shape, until the desired size is met. For example, an increase of 1, would add one layer of pixels around the entire mask, and so on.

We can see this technique in Figure 2.6a, where a layer of 1 pixel was added to a square. Although this method works as expected for basic shapes, it introduces some errors when we use shapes that do not have flat sides. Since the method adds a pixel around each pre-existing pixel, we see that the final shape can sometimes be different than the initial one. For example, in Figure 2.6b, we see a "plus" sign being increase by one. It's noticeable that the proportions are not exactly equal to the initial ones but, the difference should not be very problematic and most of the problems should be solved during the optimization process.

2.3.3 Shifting the masks

Now that we know the position of the star in each frame, having the mask accompany the movements is trivial: we need to calculate the changes in both axis in relation to the initial position and, afterwards, shift the initial mask by that amount.

With the masks moving, we may encounter cases in which the shift is such that the mask goes outside of the image boundaries, thus picking up "empty" space, as seen on Figure 2.7, for the rightmost star. Although the areas outside the image are typically small, they can still impact the overall quality of the light curve and introduce errors, mainly due to the fact that, in practice, a number of pixels "disappears" for that point in time.

The mask breaching the image boundaries is expected to occur for stars so close to the edge, as in this case, in which the mask's shape or small uncertainties in the tracking method can lead to such a situation.

Even though most of those edge cases are handled during the mask optimization process, discussed in Chapter 2.3.4, there are some that can not be avoided, e.g. if the base mask already leaves the image in some points.

2.3.4 Masks optimization

After determining the initial masks, they are either equal to the star's shape or a circle with a previously defined radius. However, we want to find the best mask for each star, i.e. the one that makes the best compromise of background-star area inside it. If the mask is big, it will pick up more background noise, and thus the light curve will have an higher flux level. On the other hand, if the mask is small, then some parts of the star will be outside of the region and thus the light curve will have more noise or even, in some cases, fail to have meaningful data inside it.

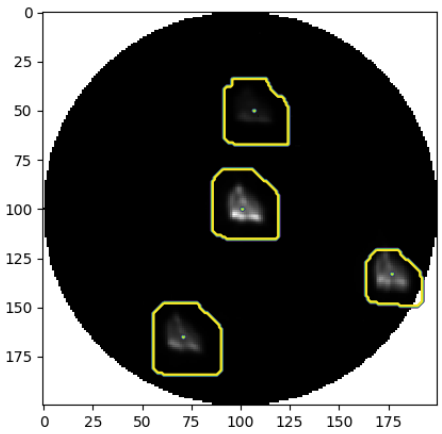
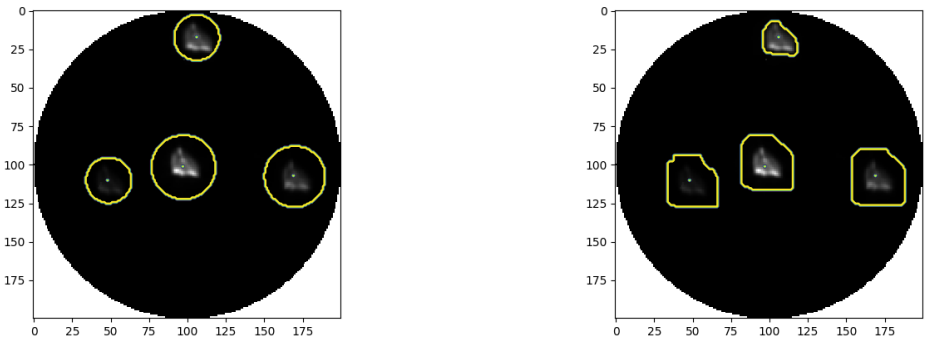


FIGURE 2.7: Edge case in which the mask is outside of the image region, due to mask shift process.



(A) Optimal *circular* mask.

(B) Optimal *shape* mask.

FIGURE 2.8: Optimized masks for both the *circular* and *shape* methods.

Similarly to the official pipeline, we shall find the mask size that minimizes the noise metric which, in this case, is the CDPP, discussed in Chapter 2.6.

In Figure 2.8 we can find the optimized masks for the 4 stars in the image. It's important to take note of the upper star for the *shape* method, Figure 2.8b, in which the mask is very close to the star's outer edge. This occurs due to the proximity to the border, that may lead to the mask leaving the image area. During the optimization process such masks are discarded when possible. Otherwise, in the edge case of having a star so near the border that it's impossible to not leave the image, then the lowest value possible is given to the mask, which tends to be a mask with a size of 1.

During the optimization process we expect the noise to decrease until it eventually hits a minimum value and, afterwards, starts increasing, as the mask starts covering a

greater portion of the sky.

However, if the mask is much smaller than the star, i.e. if the initial value for the mask size is unfit for the star's size, there is no guarantee that the noise follows this behaviour and thus the algorithm may converge to the wrong value. Further details on this optimization process and the challenges that it faces, are left for Chapter 4.2.2.

2.4 Background grid

Both the mask's shape and position are the key factors to achieve light curves with less noise. Ideally, we would like to have a image with more resolution than the one that comes out of *CHEOPS* CCD. Due to the usage of shape-based algorithms for the masks and star detection, alongside the integer conversions that must be made to convert positions into grid coordinates, the movements on the image are limited to the grid's nodes.

As an example, let us assume that we have estimated the star's centroid to be in the pixel coordinates [100.4, 100.4]. When working with images, we do not have a continuous surface and, instead, we should think of it as a equally spaced grid, in which each node represents a pixel. Thus, if we wish to link those coordinates with a location in the image, we must make some approximations. Once again thinking on the pixel grid, the distance between the grid node 100 and the 101, is all part of the same pixel and all points that fall within it are converted to the point [100,100]. If we could increase the number of nodes in our grid between any two pixels, we could approximate the coordinates to a node much closer to their actual value. One simply way way to accomplish the introduction of more nodes is to spread a pixel over a given number of pixels that would comprise a "block".

The goal of this technique is to spread the flux of a single point to a block of points, transforming a square with side of $N_{original}$ pixels in each side to one with $N_{increased}$ pixels in each side. Following this notation, we can define a scaling factor, given by Equation 2.4.

$$scaling\ factor = \frac{N_{increased}}{N_{original}} \tag{2.4}$$

However, if we transformed 1 pixel to a block of 9, we would have a 9 times increase in the flux of the entire image. So, we need to normalize the increased image, so that each block, when summed together has the value of the pixel that it is substituting. In practice, this means that the image is divided by the square of the scaling factor.

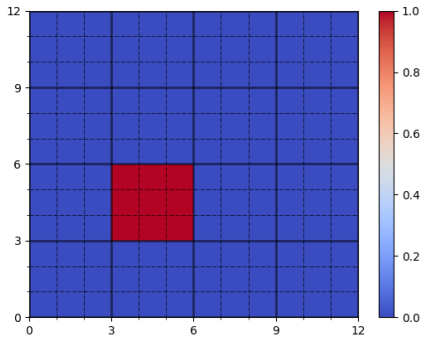


FIGURE 2.9: Example of a conversion between two grids, the first one with a side with 4 pixels and, the bigger one with 12.

Now that we can increase the images, we face a second challenge: we have to be able to transform the coordinates from the normal grid to the increased one.

Let us assume that we have a grid with a side of 3 pixels and wish to convert it to one in which the size has 6 pixels. So, each pixel is converted into a block of 4 pixels. If we now wish to convert pair of coordinates inside this normal grid to the bigger one, we face a problem: which one of the 4 pixels should be chosen for to represent it. We could define the center as the intersection of the 4 individual pixels but, to simplify, we can impose constraints to the grid size. If we always make sure that the background grid has an odd number of pixels, in each side, each block will have a well-defined center. Under those conditions, the point in the normal grid, $P_{initial}$ is converted to P_{final} through Equation 2.5:

$$P_{final} = \left\lfloor \frac{scaling\ factor}{2} \right\rfloor + P_{initial} * scaling\ factor \tag{2.5}$$

In Figure 2.9 we see an image with a square with a side of 4 pixels being converted into one with side of 12. In this example, the original image had 1 pixel with a non-zero value and, after the conversion, we have a block of 9 pixels with a non-zero value in its place.

After proving that the method is working as expected, we can apply it to one image and attempt to assess the quality of the conversion. To do so, we decided to apply a simple test: a circular mask, with radius of 20, was placed on the point [100,100] on the normal image and we calculated the flux that passed through it. Afterwards, we increased the image and the mask, with a grid of 600 points in each side, and recalculated the flux. Comparing both measurements, we find a difference of 3.8e-9, which is small enough to

not warrant concerns, thus, validating the capability of this method to increase both the masks and the images, without introducing significant changes on the final result.

Lastly, there is a detail that we must keep in mind: we can see the increase in the grid size, as an increase in the correlated points, i.e. if one pixel is transformed to 9, then those 9 pixels are equal to the first one, as well as all the errors associated with that pixel, even after normalizing the block of pixels, to make sure that we do not introduce flux into the images.

2.5 Errors in the centroid determination

Given the low errors in the detection of the central star, reported by the DRP [13], we shall now compare our results against the position of the centers used as input for the simulator.

However, before we can calculate them, we have to retrieve the "truth" values. In the simulation data we have the information for the center's position in each image. Looking at the simulation outputs, with the injected data, we see values for the PSF center, with a cadence of 1 second. Due to the spacecraft jitter during the rotation, the X and Y position of the PSF's center is not always the same and thus we have to calculate the mean value of those positions to reach a value that we shall take as the true position of the centroid.

Now that we have the real centers for the target star, we can calculate the errors for the initial detection and see how the three different star tracking methods impact the errors. Lastly, for the background stars, since we do not have any stored information, we shall only compare them amongst themselves.

2.5.1 Initial Detection

If we look at Figure 2.10, we can see the differences between the points obtained with the *dynam* method and with the *fits*. At a first glance, the points appear to be almost equal, with slight differences between them. However, when we look at Table 2.1 we see that it's not the case.

In Table 2.1, we see that *fits* method presents much lower errors than the *dynam* one. Remembering back from Chapter 2.1.1, the *fits* method, for the target star, uses the DRP's centroid estimation. Thus, we clearly see that the method applied in the official pipeline

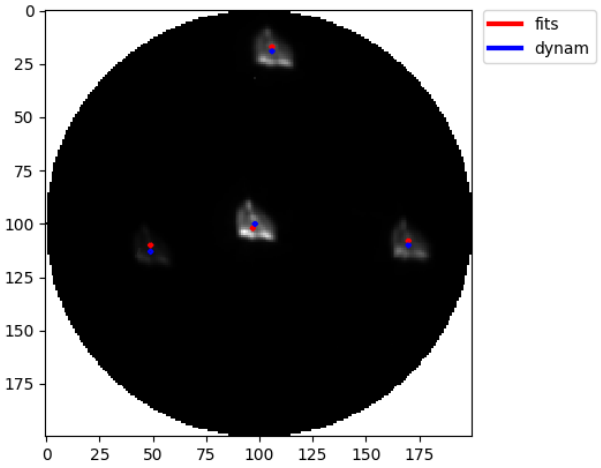


FIGURE 2.10: Comparison between the initial points determined with the *dynam* method, in blue, and the *fits* method, in red.

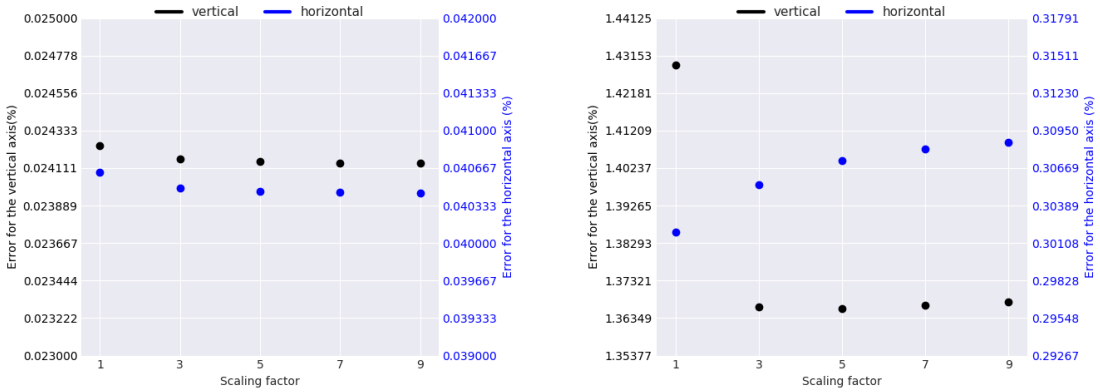
TABLE 2.1: Deviations, in pixels, for both initial detection methods, when comparing against the "truth" position of the centroid. The calculations were done for three different values of background grids.

Method	Background grid					
	0		600		1200	
	Delta X	Delta Y	Delta X	Delta Y	Delta X	Delta Y
fits	-0.0246	-0.0396	-0.0738	-0.1188	-0.1477	-0.2375
dynam	1.4508	-0.2942	4.1747	-0.8960	8.8566	-1.3070

produces the best estimations. To test the accuracy with the background grid active, we need to convert the true values with Equation 2.5.

In Figure 2.11 we can see the behaviour of the errors with the increase of the background grid. Starting with the *fits* initial detection method, we see that, the errors stay approximately equal for all background grid values. This result was expected, since the only differences introduced by the grids, in this method, is the scaling of the values calculated over the original image. If the estimated points and the true points increase by the same amount, we see no changes in the percentage of error.

However, that is not the case for the *dynam* method. At first, the increase of the grid size yields a significant reduction in the errors for the vertical axis. Unfortunately such is not the case for the horizontal axis, in which we can notice a gradual increase in the errors. Comparing the magnitude of the decrease and the increase, we see that the gains for the vertical axis far surpass the losses for the horizontal one.



(A) Using the *fits* initial detection mode.

(B) Using the *dynam* initial detection mode.

FIGURE 2.11: Errors, in the vertical and horizontal axis, for the first image in the data set. A comparison is made between the two presented methods, only for the target star.

We can also see that, as far as the error in the detection is concerned, using a background grid bigger than 600, i.e. a grid of 600x600, does not yield any significant improvements, but increases the computational burden. Later on, in Chapter 5, we shall see how the background grid actually impacts our light curves.

2.5.2 Tracking errors

Now that we have characterized the initial detection methods, we can look at the tracking methods. In order to keep the comparisons constant between the three methods, we shall always use the same initial detection method: the one with the lowest error - *fits* method.

The results of this test can be seen in Figure 2.12, in which we see the errors for all images, except the first one. The *offsets*, which for the central star is the DRP's estimations, yields the lowest, and almost constant, errors. Similarly, the *dynam* method also gives constant estimations of the central point, albeit with higher errors. We also notice that the deviations in this method are more related to the vertical axis, than with the horizontal one.

Lastly, the *static* method yields the worst results, as one would expect, since it applies no corrections on the target star's centroids. Furthermore, since the initial point is also subjected to the rotation applied by this method, it can easily "jump" between nearby pixels, thus introducing the visualized pattern.

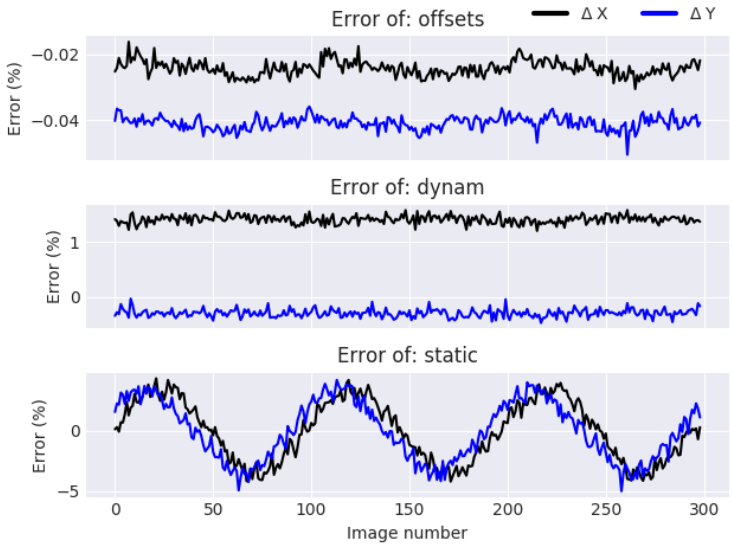


FIGURE 2.12: Evolution of the errors in the estimated centroid's position, for the three star detection methods. For the initial detection method we used the *fits* method, which is presents the lowest errors, as seen in Chapter 2.5.

2.5.3 Comparison between star tracking methods

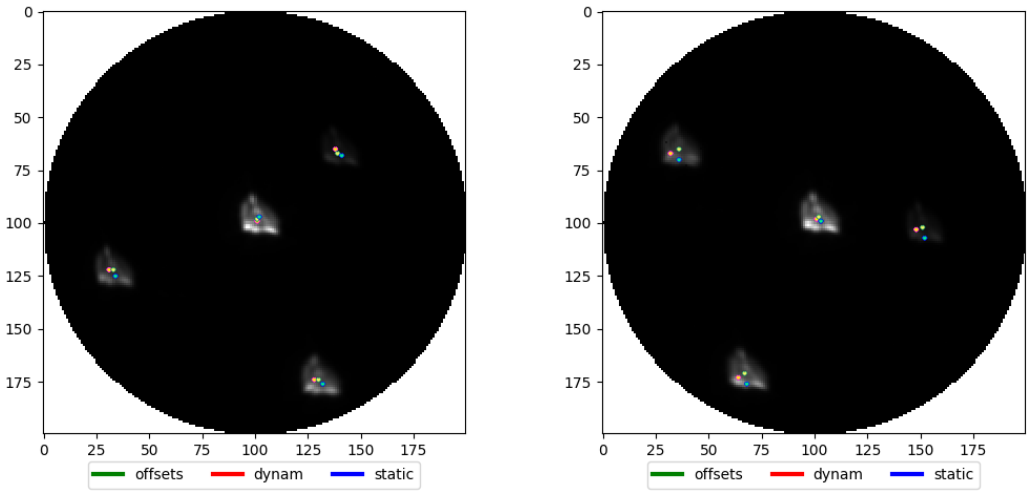
Now that we have seen how the target star detection and tracking is impacted by the corresponding methods, we still have to see the behaviour for the background stars. Unfortunately, we do not have information regarding their true position, thus prohibiting a proper analysis of the methods. However, we shall perform a more basic analysis: we shall compare the relative position of each estimated point inside the star's shape. This analysis will be performed over 2 different data sets.

Further along this thesis, in Chapter 5, the impact of each method on the produced light curves shall be better studied and thus allow a better comparison between them.

We can start by looking into a small data set, with only 300 images. We expect the *dynam* method to be able to consistently follow the stars. On the other hand, both the *static* and *offsets* methods are expected to suffer some deviations, due to relying on the rotation angle values to find the next value.

In Figure 2.13 we see the results yielded by each method for two different points in time. Starting with the earliest one, Figure 2.13a, we find small but almost irrelevant deviations between the positions estimated with each method.

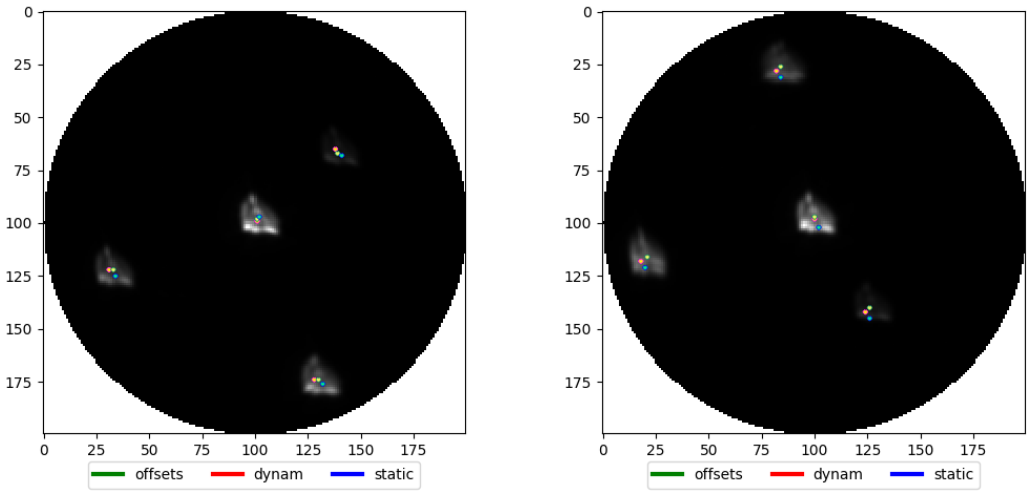
Contrasting this result, if we look at the centers at a later time, in Figure 2.13b, we now see that the points obtained with the *static* method are almost out of the stars, with the *offsets* slightly nearer to the center.



(A) Centroids estimated by each method, for the 40th image.

(B) Centroids estimated by each method, for the 55th image.

FIGURE 2.13: Centroids estimated by each method, for two different images: One in which the *static* method yields a good estimation and one, in which it yields a bad estimation. This analysis was done with a data set with 300 images.



(A) Centroids estimated by each method, for the 40th image.

(B) Centroids estimated by each method, for the 1940th image.

FIGURE 2.14: Centroids estimated by each method, for two different images: one near the beginning of the data set and, the other, near the last one. This analysis was done with a data set with 2000 images.

With this behavior in mind, we can now look into a bigger data set, with 2000 images. In Figure 2.14 we once again picked two images, one near the beginning of the data set, and one near the end. Albeit not as apparent, the same behaviour is found: for the first image, we notice the three estimations clumped together, although the *offsets* method presents some deviations. At a later stage, we find significant differences between the points, with the ones from the *static* method almost outside of the star.

After verifying the same phenomenon in two different data sets, we can got back to the earliest data set, in order to try to find a pattern capable of justifying it.

Looking into the *roll angle* differences between consecutive images, in Figure 2.15, we find a sinusoidal oscillation. If we compare the pattern with this oscillation, we see that the case in which the estimations are close together occurs when we are near the minimum of this signal and, when away from this region, we have a dispersion of the estimations.

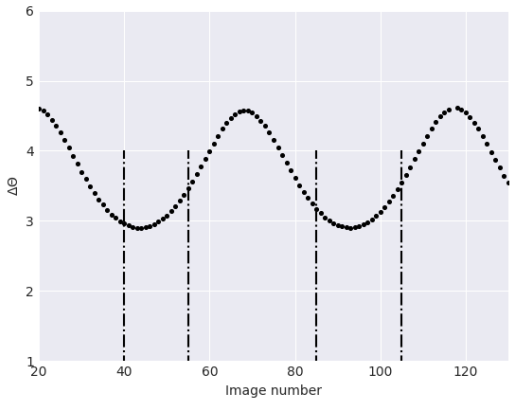
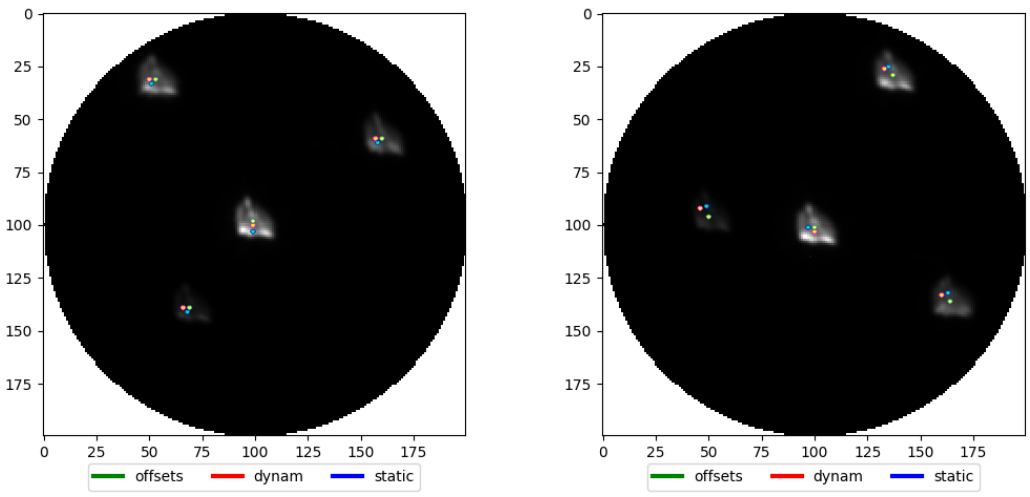


FIGURE 2.15: Differences between two consecutive *roll angles*. The black lines represent the points corresponding to the the images of Figure 2.13 and 2.16.

To test this hypothesis, we collected another two images, from the same dataset, Figure 2.16, that had *roll angle* differences under the same conditions and checked to see if the pattern still holds.

Unsurprisingly, we found this pattern, thus validating our hypothesis: Around the minimum, periodic, values of the *roll angle* differences, we find a region around which the differences are small and thus the rotation-based algorithms work properly. Outside of those valleys, we start to find deviations that not even the corrections applied on the *offsets* method can counteract.

There were no further attempts to characterize this region, since the tests could not be automated in a reliable way due to the fact that there is no clear criteria on what makes a good center determination. One possible way to understand it better could be comparing the rotation-based methods against the *dynam* one, and find where the differences between start to be noticeable.



(A) Centroids estimated by each method, for the 85th image.

(B) Centroids estimated by each method, for the 105th image.

FIGURE 2.16: Centroids estimated by each method, for two different images: One in which the *static* method yields a good estimation and one, in which it yields a bad estimation. The images are from the data set also shown in Figure 2.13, but at a later time.

2.6 Noise metric - CDPP

In order to estimate the noise in the light curves, an adaptation was made to the algorithm applied in NASA's Kepler mission: Combined Differential Photometry Precision, or CDPP. During the MSc's thesis of Pedro Silva [52], the algorithm was ported to be used for the CHEOPS mission, and it shall be used as the noise metric to compare the quality of the light curves obtained with the developed method against those given by the CHEOPS's mission official pipeline, DRP.

In Christiansen et al words [53]: "*A CDPP of 20 ppm for 3-hr transit duration indicates that a 3-hr transit of depth 20 parts per million (ppm) would be expected to have a signal-to-noise ratio (S/N) of 1, and hence produce a signal of strength 1 σ on average*", which is the ideal metric to quantify the noise existent on the light curve and, it can be interpreted as the effective white noise seen by a transit pulse [53]. In its calculation, the important factor is the near-term trend changes in brightness instead of the long-term ones [54].

For further details on the algorithm itself, one can refer to the works of [53, 55], where the algorithm is properly introduced and discussed. However, for a brief introduction on the methodology behind it, I will now refer to one of its many adaptations, in [56]:

- Start by passing a 2 day quadratic *Savitsky-Golay* (SavGol) filter to the flux, which should be more than enough to avoid fitting the transits in the Light Curves, as we shall see further below;

- Remove outliers outside 5 σ;
- Divide the data into chunks of data, which have the points correspondent to the integration time. Afterwards the standard deviation is calculated for each one of them;
- Take the median of the standard deviations and divide by the square root of the number of chunks, obtaining the desired photometric precision.

However, since all the referenced papers were made for the *Kepler* mission we had to introduce some changes to the window lengths in use. Similarly, regarding the *Savitsky-Golay* filter, its application results in a suppression of the white noise and thus, in *Kepler*'s case they have to scale the CDPP by a factor of 1.168 [57]. In *CHEOPS*'s case we are using the same ratios between the different parameters of the filters, thus having the same suppression of the expected rms from white Gaussian Noise, as reported in [57].

The goal behind the SavGol filter is different between the two missions. Whilst in the Kepler mission the transits are expected to be a small part of the light curve, on CHEOPS we expect them to be a more significant part of it. For the Kepler mission, the SavGol filter is made in such a way that the window has more points than the transit, but not enough to start fitting the astrophysical noise in the light curve. If the conditions are met, the filter should be able to pass through the transits, without fitting them, and thus the CDPP is calculated without removing them, which should not be impactful due to them being much smaller than the light curve. On the ported version, for the CHEOPS mission, this filter is now expected to fit the transit without fitting the noise present on the light curve, which introduces some constraints on the window sizes, as discussed in [52].

To calculate the CDPP with a time scale of 30 minutes we shall use the following values:

- Calculation window: 30 minutes - Integration time.
- winlen: 10 minutes - Size of the convolution window;
- Savgol filter window: 41 minutes - Should be two times the number of points that correspond to the integration time. However, it was determined, empirically, in [52] that this value was the optimal one for removing transits from smaller data sets, since 61 points started removing noise from the light curve.

At the time of writing, this implementation used the image number instead of the time correspondent to the image to calculate the noise. So, one can imagine that it will not fare well if the data sets have temporal gaps in them, or for bigger data sets we might find that the Savgol window is not good. In such cases, we shall switch to the DRP's implementation of CDPP

The official DRP reports the usage of a modified CDPP algorithm, which does not make use of the *Savitsky-Golay* and, consequently, does not take into account the possible transit in the light curve. So, the noise estimation is made over the noise and transit at the same time and, if possible, we would like to steer away from this metric whenever possible.

2.7 Uncertainties

In Chapter 1.2.1 we presented the equation to calculate the uncertainties in each point of the light curve.

As we have seen, in this Chapter, the utilization of the background grid will lead to an increase of the image size and, consequently, the mask size. For equation 1.1 we must calculate the number of pixels inside the image. Before we can apply it, we must retrieve the relevant parameters from the DRP's outputs. Even though most of them are straightforward to retrieve, the *background* and *dark* are not. Both of them suffer the same problem: in the DRP outputs, we can only find the values for the central star, without having the values outside of this region.

As discussed in Chapter 2.4, the increase of the grid leads to an increase in the correlated points. Furthermore, since all the parameters stored in the files are calculated over a 200x200 region, we will have a dimension mismatch and thus overestimate the uncertainties. As an example, when using a background grid of 1000, we will have a mask size much bigger than when we are not using a background grid. Thus, we must convert the number of pixels from the increased grid to the normal one, which can be accomplished by dividing the number of pixels in the mask by the square of the increasing factor, which gives a corrected mask size.

2.7.1 Background

The background stored in the DRP outputs, is given by Equation 2.6, where *background* is the background level, *mask_pixels* is the number of pixels in the mask used to calculate the light curve and *Exposure_time* is the exposure time of each image.

$$stored\_bg = background * Exposure\_time * mask\_pixels \tag{2.6}$$

With this knowledge, we want to extract the background per pixel, to calculate the uncertainties for our mask of choice. One aspect to take into account, would be the fact that the DRP's background calculation is made over the image outside a region delimited around the target star, i.e. it also factors the background stars. We could try to improve the background calculation with the masks and star tracking techniques so far described but, we would be introducing errors due to the movement throughout the CCD pixels and, some parts of the DRP would need to be re-implemented to allow us to work with images without background correction already applied on.

Thus, as an approximation, we shall assume that the background is valid for our case and, if we divide the stored_bg by the number of pixels inside the DRP's mask, we get an estimation of the background*Exposure_time.

2.7.2 Dark

Similarly to the background, the *dark* stored in the outputs is only calculated for the region near the central star, using the image outside the applied mask.

The actual values are stored in temporary files during the DRP operation and, afterwards, deleted. Thus, to have this information, we would have to recalculate it. Even though the *dark* value is not stored, in DRP's outputs we can find the dark component in the uncertainty calculation, given by Equation 2.7.

Since we know the radius of the circular mask used for our data set, we can simply divide this stored value by the number of pixels inside the mask, and thus have a rough estimation of the *dark*. However, DRP makes use of 4 different masks and, if we are not careful, nearby stars can impact its value. Thus, in an attempt to minimize the contaminations, we can calculate the median of the estimated *dark* for each one of the 4 DRP's apertures, and regard it as the *dark* value.

$$stored\_dark = dark * Exposure\_time * mask\_pixels \tag{2.7}$$

The downside of this method is that it assumes that we have an uniform dark, that is equal for both the target and the background stars, which may not hold as true. But, due to the difference in the orders of magnitude of the dark and of the flux level being so big, the small imperfections should have no meaningful impact in the estimated uncertainties.

Chapter 3

Gaussian processes

3.1 Introduction to Gaussian Processes

Before we begin this section it's important to note that this is only a (very) brief introduction of Gaussian processes, without the usual mathematical rigor that accompanies them and, for a more detailed description, one should refer to [58], a book written for "graduate students and researchers in machine learning at departments of Computer Science, Statistics and Applied Mathematics".

A Gaussian process is a generalization of the Gaussian probability distribution [58] and, unlike a probability distribution which describes scalars or vectors, a stochastic process controls the properties of functions. This process is completely specified by its mean function ($m(x)$) and covariance functions (kernel) and can be defined as:

$$f(x) \sim GP(m(x), k(x, x')) \tag{3.1}$$

where m(x) is the mean function and k(x,x′) is the kernel.

They are a very versatile tool, capable of working with any given number of inputs, and can model processes that are combinations of aperiodic, periodic and quasi-periodic signals, [47]. In astronomy, they have been used in numerous works to model systematics, e.g. [59, 60].

As we shall see further along, in Chapter 5, the light curves have noise that is introduced by the satellite's jitter and rotation, and thus we will attempt to model and remove it from the light curves.

3.1.1 Covariance function

The Covariance function, or Kernel, will control the behaviour of the GP, such the smoothness and (in)variance with translations.

Even though we can use any kernel that we wish, we have to take a consideration into account: it has to be a symmetric and positive semi-definite matrix [61]. So, in order to comply with these requirements, one must introduce a white noise term to the diagonal of our kernel (the noise is expected to be uncorrelated so we don't have to introduce the compensation on all elements).

We have many kernels to choose from but, since this thesis does not need a more detailed explanation and understanding of all the different kernels, one should once again refer to [58] if more information is needed.

Even with the existence of a great number of different kernels, they might still not be good enough to describe the desired behaviour. For example, if we are working with bi-dimensional data and we wish to use a kernel that has different behaviours over each dimension. In such cases, we can combine kernels, i.e. adding them or multiplying them either to other kernels or with a constant value.

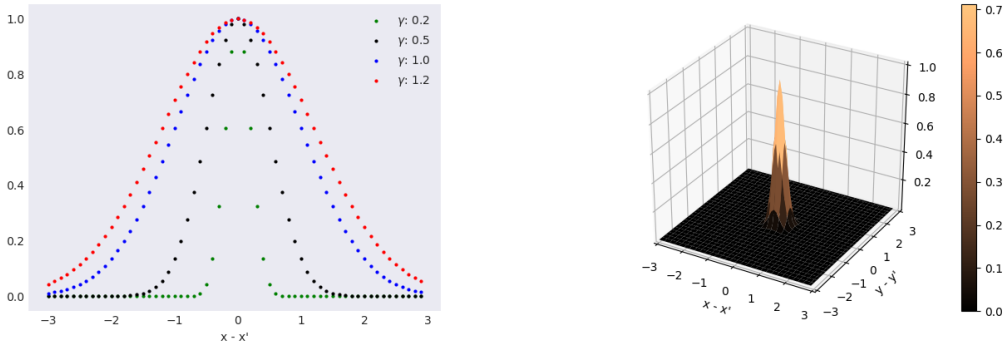
We can classify the standard kernels in one of two different ways, "stationary" or "non-stationary". As the name suggests, the first kind is invariant to translations, while the second kind depends on the input position. The stationary kernels are a function of "x - x' ", which means that equal translations induce equal changes in the kernel.

The Exponential squared kernel, depicted in Equation 3.2, is a stationary kernel and one of the most used in the kernel machines field [58], assuming a smooth and infinitely differentiable function [62]. Following [63], in which this kernel was also used to create a systematics model, we shall also use it.

$$ExpSquare(x - x') = \theta^2 * exp\left(-\frac{(x - x')^2}{2\gamma^2}\right) \tag{3.2}$$

This kernel has two hyperparameters, *theta* and γ. The first is the amplitude and will modulate the vertical axis changes, while the second is the length-scale, controlling the decay in the correlation between two points separated by x - x'.

If we now fix the amplitude of the kernel and just change the length-scale, as in Figure 3.1a, we see that smaller length-scales values will lead to a kernel capable of changing quickly, while higher length-scales will in turn signify a "slower" kernel, not as capable of keeping up with high frequency oscillations.



(A) Squared Exponential kernel for different length-scales, in the 1D case.

(B) Multiplication of two Squared Exponential kernels, for the 2D case.

FIGURE 3.1: Visualization of the Squared Exponential kernel.

As we have said, if we want a kernel that varies across more dimensions, we can simply multiply two kernels that only depend on one of the dimensions, as seen in Figure 3.1b, in which two Exponential Squared kernels with the same amplitude and length scale were combined. We can describe this bidimensional kernel through Equation 3.3 and, in practice, it tells us that the function value f(x,y) is expected to be close to f(x', y') if and only if both x and y are close to x' and y', respectivelly [64, 65].

$$k_{2D}(x, y, x', y') = k_x(x, x') * k_y(y, y') \tag{3.3}$$

3.1.2 Mean function

We can think of the mean function as a way to express our beliefs on the function before looking at data, and thus control forecasts in regions for which we have no data [66] or, in other words, it's our model. The most common mean function is a flat mean (read zero) function, which typically reflects the (lack of) knowledge on the function itself.

In our case, we shall apply a little trick and also assume a flat mean function. Before we delve deeper into it, we have to first create a reliable model of the expected light curve of a stable star with a transiting planet, as in Figure 1.4.

To solve this problem we can create this model with Python's *batman* package [67] and, assuming that we know some values of the star and the planet (Table 3.1), we can create a model of the transit, as seen in Figure 3.2.

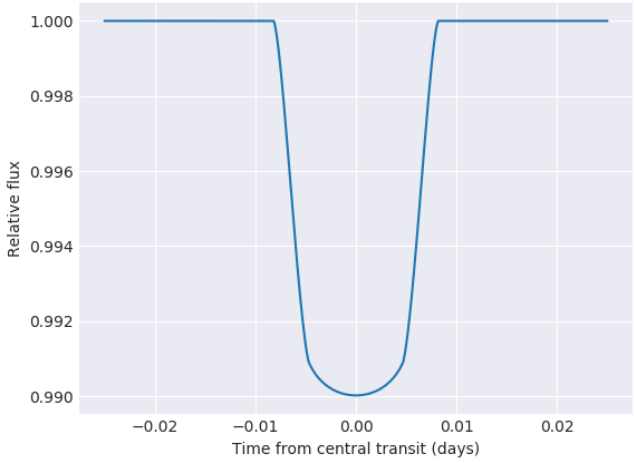


FIGURE 3.2: Example light curve, created with the batman package.

Parameter	Value
R_P/R_S	0.1
a/R_S	15
Inclination	87 deg

TABLE 3.1: Parameters used to create the model in Figure 3.2.

Now that we have a model of the expected light curve and the measured light curve, we can subtract one from another and, if the derived parameters are close enough to the real ones, we should have a value of almost zero for each point. We find an almost zero instead of a zero because we still have noise, which is not accounted by the theoretical model.

If we pay closer attention to Figure 3.2, we see that the model light curve is normalized, i.e. outside of the transit we have a value of 1, which is not the case for our light curves. Thus, we will need to also normalize our light curve, which is not as trivial as it might seem, as we shall see in Chapter 3.2.

3.1.3 Markov Chain Monte Carlo - MCMC

If we wish to sample and provide sampling approximations of the posterior probability density function, our best option is to use numerical methods, amongst which, the Markov Chain Monte Carlo (MCMC) method is widely used.

This method iteratively constructs a Markov Chain, using the posterior distribution as its equilibrium distribution, i.e. the distribution to which the chains will converge to [47]. A chain, that we can think of as a walker, is started from a point in the parameter space and proposes a new one, which is accepted or rejected, based on the posterior density ratios between the two locations.

Since we are using a finite chain, the starting location, in the parameter space, can affect how it converges. For example, if the chain starts far away from the posterior, then it

takes time before it starts sampling from the high-probability posterior space [47]. In order to diminish the effects of the starting position, we can discard a percentage of our chains. The portion of the chain that is discarded depends on the context of the problem. This practice of discarding a portion of the chains shall, henceforth, be referred to as "burn-in". Furthermore, typical approaches make use of many chains at the same time, thus allowing to start from different initial positions. There are many approaches for the creation of the initial values, as we shall see in Chapter 3.1.3.1.

A more in depth explanation of the algorithm is outside the scope of this text, and an interested reader should refer to [47] for a general and high level overview of the method , and [68] for specifics on *emcee*, a Python package that implements a MCMC sampler.

3.1.3.1 On the practical application of the MCMC

In order to use the MCMC method, we shall use the *emcee* Python package, that is widely used in astrophysics literature. Now, we will look over some of the key details that one should keep in mind when working with these methods.

In the first place, we need to initialize the walkers and, for that, we have (at least) three different approaches:

1. Start the walkers at a sampling of the prior;
2. Start the walkers around a point, in the parameter space, that is expected to be close to the maximum probability point;
3. Start from a sampling of the prior distribution and go trough a 'burn-in' phase in which the prior is continuously transformed into the posterior.

If the process is going well, the acceptance fraction, i.e. fraction of proposed steps that are accepted, should be in the [0.2 - 0.5] interval and, if it reaches a very low value, it's a clear indicator that we have a problem [68].

We should also run a large number of samplers, until performance problems start to arise, which leads to the compromise of using the greater of:

1. Smallest number of samplers for which the acceptance fraction during the burn-in is still acceptable.
2. Number of samples that we want in the end.

3.1.4 Problems of Gaussian processes

Although Gaussian processes are a powerful tool, they still have some associated drawbacks. The main problem passes by the computational burden that accompanies them. Those costs are derived from the size of the data set in use and from the complexity of our model. If we introduce more freedom in our model, through an increased number of parameters and hyperparameters, we will have a more complex model but, we will also increase the cost of the marginalization process.

The biggest bottleneck present in this process is the inversion of the covariance matrix, that has a $\mathcal{O}(N^3)$ time complexity, following the Big-O notation [47, 69]. For this thesis, the *george* Python library shall be used to implement the Gaussian Processes and, if the size of the data sets in use is such that we cannot apply the Gaussian Processes in a reasonable amount of time, then we could apply the GPs over parts of the light curve, as described in [60].

With the usage of a computational cluster, there was no need to implement this technique. However, it is an interesting approach, especially if we only want to fit the planet's parameters. If we also want to properly model the noise in the LC, we do not know if such approach could be viable, or how the results would compare to those obtained with the full light curve. It's also important to keep in mind that all the tested data sets had, at maximum, around 3000 points which we can still consider a reasonable amount of points.

Lastly, due to the high number of kernels available, as seen in e.g. [58], the task of choosing the right kernel for the problem in hand is not always trivial. The kernel choice is typically made through experience and intuition or, if we do not have them, then through a combination of the basic kernels [70].

3.2 Light curve normalization

3.2.1 Ideal case

In a ideal case, we would have no contribution of neither solar spots nor instrumental effects on the light curve (LC) and thus, the out-of-transit regions would be almost flat, similarly to the model of the light curve presented in Figure 3.2. Under those conditions, or near them, if we divided the light curve by the median of the out-of-transit region we should be able to obtain a normalized light curve.

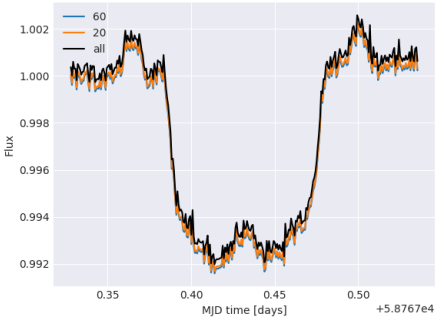


FIGURE 3.3: Light curve normalization, with division by the median of the flux. In blue, we performed a division by the mean of the first 60 points, in Orange, the mean was calculated over the first 20 points and, in Black, using the median of the entire light curve.

Even though this division is capable of normalizing the light curve, under such conditions, it fails in two different points: first, the number of points used to calculate the median influence the overall shape of the normalized LC, as seen in Figure 3.3. Secondly, we do not know, a priori, where the transit occurs within the LC and thus, we could consider it in the median, thus not fully normalizing the LC and instead introducing a, previously non-existing, trend on the LC.

3.2.2 Linear trend removal

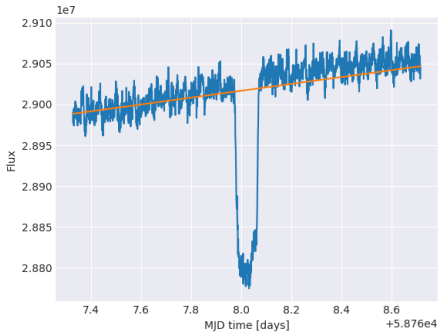
After seeing that the division by the median was not a good approach to the problem, we decided to try a different approach. As an approximation, we shall assume that the contribution from the stellar spots is linear throughout time, which is not necessarily true, especially if the spot stops contributing to the light curve during the observations.

Since we are attempting to remove a linear trend from our LC, we can choose two different regions in the light curve, outside of the transit, and fit a line that passes through them. Instead of using a single point, we use use two blocks of 10 points and perform a linear regression .

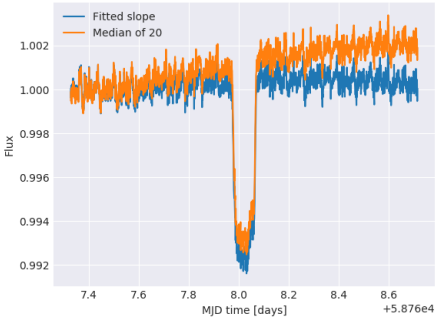
If we choose both edges of the light curve, present in Figure 3.4, fit a line to them and, posteriorly, we divide the LC by this line, we find that the normalized light curve is almost flat, as seen in 3.4b.

Now, that we have validated that a division by the fitted line yields normalized light curves, we have two different cases to take into account:

https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.linregress.html; Accessed: 14/8/2019



(A) Fitted line overlapped with the light curve.



(B) Comparison between normalized light curves. In Blue the one normalized with the fitted line and, in Orange, the one using the median of the first 20 points.

FIGURE 3.4: Normalization of the LC, with a linear fit outside the transit region.

1. The transit is away from the edges of the LC;

2. The transit is near the edges.

Unfortunately, there was no time to develop a transit detection technique and thus, at the time of writing, the choice of region from which the points for the linear fit are chosen, is manually defined. It follows that, if we are not careful when choosing the points, the normalization process will end up exacerbating the trend in the data, instead of removing it.

3.3 On the application of the Gaussian Processes

3.3.1 Choosing the kernel

As we have seen in Chapter 2 and will see in Chapter 5.1.2 the satellite's rotation is closely tied to many of the phenomenon seen in the light curves, so we shall use it as the input for our kernel. Instead of using the rotation angle, we could use the time of observations. However, the linearity of time does not describe well the circular nature of the rotation angle and the periodicity of the noise.

Furthermore, this technique, with the rotation angle of the satellite, has already been applied for the *Kepler* mission, showing good results [59]. However, in that case the satellite never completed a single lap around its center, since the spacecraft's thrusters are used

every few hours, returning the roll angle to its nominal value. In such case, the start position and the final one are not expected to be correlated, which simplifies the application of the Gaussian Processes, as we shall now see.

In CHEOPS case, as we have seen, the satellite is rotating around its line of sight, and thus the CCD is continuously rotating. Furthermore, not only do we have the transformation in the angle that occurs when the CCD's orientation return to its original orientation, i.e. it passes from $\theta = 360$ to $\theta = 0$, but also the ones induced by the SAA and/or Earth occultations, as we can see in Figure 3.5.

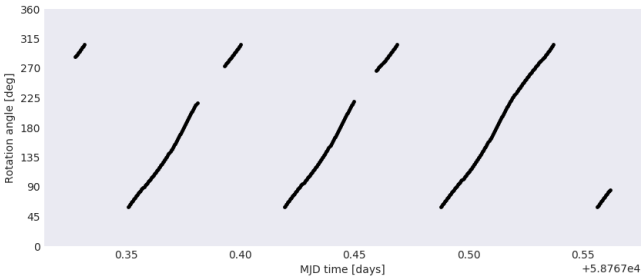


FIGURE 3.5: Rotation angle of the satellite for a Data Set with SAA induced temporal gaps.

We can classify our data as directional data, i.e. the noise found on the light curves depend on the satellite's orientation, at any given time. We can also refer to this kind of data as circular data, and it can be found in many fields, such as in political science [71], image processing [72], weather analysis [73], geology [74], astronomy [75], amongst others.

To tackle these kinds of problems with Gaussian Processes, one has to carefully choose an approach. The typical models, do not take into account the disk's geometry in their kernel and, in the literature we find multiple approaches to solve this problem, such as *polar GPs* [76] and *wrapped GPs* [77]. Even though both approaches reveal themselves promising, we did not have the time to implement and test either of them.

Instead, we shall make use of a similar approach, with a 2D kernel. If we think about a unitary circumference, we know that it's possible to specify a point in it, using the sine and the cosine of the angle between that point and our angle of reference.

As we have seen in Chapter 3.1.1 we can multiply two unidimensional kernels to create a 2D kernel, in which two functions values, f(x,y) and f(x',y') are only expected to be close if both x and y are close to x' and y', respectively. Since we expect the noise, in any given point, to be correlated with the noise from adjacent points, we can apply this

bi-dimensional kernel over the sine and cosine of the rotation angle, as in Equation 3.4, thus guaranteeing that the correlation is high between points with almost equal $sin(\theta)$ and $cos(\theta)$ or, in other words, when they are near each other.

$$k_{2D}(x, x') = \lambda * k_{sin}(x, x') * k_{cos}(x, x') \tag{3.4}$$

where λ is the amplitude Hyperparameter, k_{sin} the kernel whose input is the sine of the rotation angle and, k_{cos} has the cosine of the angle as its input.

Lastly, in order to take into account the white noise, we add the previously calculated uncertainties in the Light curve to the diagonal of our kernel, thus complying with the kernel limitations presented in Chapter 3.1.1.

3.3.2 Creation of our model

Now that we have chosen the kernel, we have to choose what are the parameters of our model. We decided to use the following:

- Planet's radius, R_P;
- Semi-major axis, a;
- Orbital inclination, inc;
- Time of the inferior conjunction, t_0.

The orbital period is not determined with the GPs, but instead with Equation 1.3. As previously mentioned, while working on a Bayesian setting, we can express our prior beliefs on the values of the parameters, i.e. constrain them to be within a given region in parameter space. Constraining the hyperparameters from the kernels is not as intuitive, and thus, we shall give them a broad range of values that they can take. The parameters used to create the transit model, however, are easier to constrain. In Table 3.2 we expose the default constraints that can be used for the Data Sets, introduced in Appendix C, studied during this thesis.

As we have said, the constraints applied on the kernel's hyperparameters are relaxed, with a wide range of possible values, whilst the ones from our model have a more well-defined bounds. The kernel's bounds are defined with the natural logarithm, due to *george* inner workings, that store them as the logarithm instead of the value.

TABLE 3.2: Bounding limits for both the kernel and the model parameters.

Parameter		Lower bound	Upper bound
Kernel	$\log(\gamma_{cos})$	-20	10
	$\log(\gamma_{sin})$	-20	10
	$\log(\lambda)$	-20	0
Model	R_P	0	0.2
	a	1	6
	inc	0	90
	t_0	Min(time)	Max(time)

Lastly, we just need to find the initial values of our model. As we have previously said, given enough time, every starting position will converge to the correct one. However, the better the starting position, the less time will be spent until convergence. In order to avoid having a parameter much larger than the rest, we decided to shift our time array, so that we measure the time from what we define as the central transit, thus t_0 initial guess will be a value of zero. To define the central transit, we decided to use a very basic approach. After normalizing the light curve, we find the point in time if which the the minimum value of the flux is reached, which should be near or very near the mid section a transit in the light curve.

3.3.3 Practical application of the GPs

Now that we have defined a model and a kernel for the GPs, we can apply them, as schematized in Figure 3.6. The first step is to normalize the LC and afterwards, initialize the MCMC chains.

Even though there are many different methods to create the initial position of each walker in the parameter space, as discussed in Chapter 3.1.3, we have decided to create a ball around the most likely values. To do so, we use a sample from *numpy*'s standard normal distribution , multiply it by 1e-8 and sum to what we believe that are the most likely parameter values, thus creating a distribution of values around our initial guess.

Now that we have the initial position, for each chain, we pass to the two burn-in stages and the production stage. Fundamentally, the three of them are equal, i.e. the same process occurs. In each stage, the sampler proposes parameters, that are tested to see if they lie within the delimited regions defined in Table 3.2. If they are, we need to set the new parameters in the GP and to recompute the kernel. Afterwards, we create a transit

https://docs.scipy.org/doc/numpy-1.15.1/reference/generated/numpy.random.randn.html. Accessed: 20/8/2019

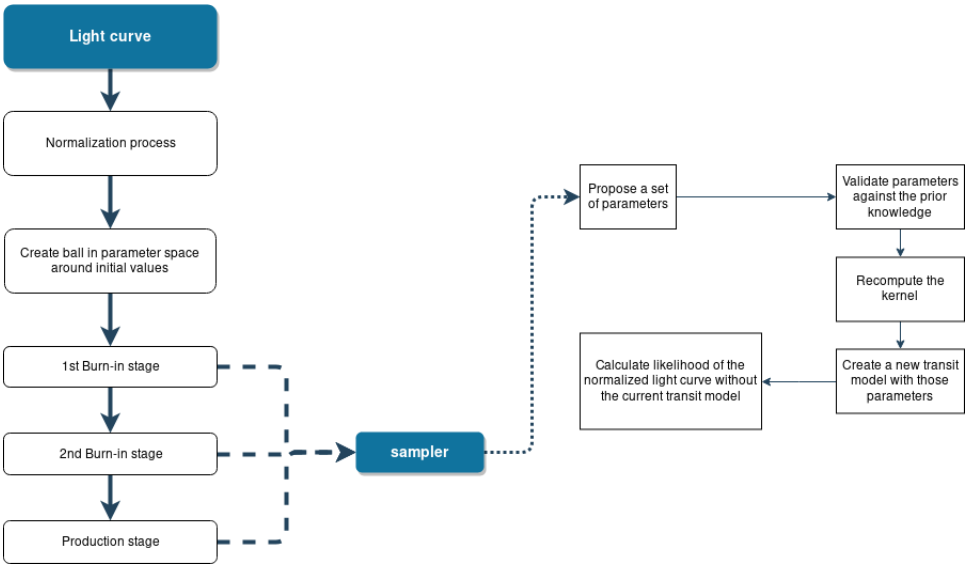


FIGURE 3.6: Schematic of the Gaussian Processes.

model with them and subtract them from the normalized light curve and, with this, we should be left with only the noise, for which we can then calculate the likelihood of it, given our model.

After the first burn-in stage we create a new ball in the parameter space, now around the position of the best walker. This burn-in stage is done in an attempt to guarantee that the walkers all started in reasonable values, in the second-burn in stage.

We will not work with the likelihood, but instead with its natural logarithm, which we shall henceforth refer to as *loglikelihood*. If any of the parameters is not within the region, then its loglikelihood takes the value of minus infinity ($-\infty$). Otherwise, we use the calculated value

The last step, after performing the three stages, is to sample our kernel, which should give us a good estimate of the noise, in each point. Instead of only performing a single sampling procedure, we shall take more samples and calculate the median value. Which can then be used to remove noise from the light curve.

3.3.4 How to present our results

After completing the production stage, we do not have a final value, but instead a distribution. To achieve a concrete value, alongside the correspondent uncertainties, we shall use the method recommended in the *emcee* documentation [78]: compute the *16th, 50th, and 84th percentiles of the samples in the marginalized distributions*. We shall take the 16th percentile value as lowest possible value for the parameter and, the 84th as the highest

possible value. The 50th percentile, or median, will be used as the final value and we can find the true values, for each data set, in Appendix C.

In order to assess the chain's convergence, we can plot each value taken by each individual walker, for each parameter, during the production stage.

Furthermore, to visualize correlations between parameters, we use *corner.py* [79, 80], which is a widely used package in astronomical literature, normally used to show results from MCMC processes. The MCMC routine, as we have seen, gives us a distribution of values, for each parameter. This package, *corner*, plots each parameter of the GP against all others. As an example, one can look at Figure 5.32.

The diagonal line in the plot represents the parameter plotted against itself. If the MCMC process went without problems, we expect to see a clear peak, in a Gaussian-like fashion. The other entries in the graph, easily allow us to search for correlations. If two parameters are correlated, we should see a line in their intersection, indicating that a change in one of them, induces a change on the other one. In this plot, we can also see how each value is distributed in the parameter space.

Chapter 4

An expansion foR the CHeops mission pipelIne - ARCHI

As we have seen so far, we have two main components in this pipeline: *photometry*, which aims to extract the light curves from all of the stars in the field and *Gaussian Processes* (GPs), which have the goal of correcting effects linked with the satellite's rotation and determining planet's parameters. In this section, we shall see how they are implemented and connected together.

In order to initialize all that is needed and to control the behaviour of the modules, we built two *Controllers*, that make an easy to use interface. At the end of this document, in Appendix E, a code snippet is given, showcasing ARCHI's user interface.

The *Controllers* are implemented independently, in such a way that we can use the GPs without needing to run the photometry by, in those cases, loading a output of a previous run. In order to maintain this modularity, custom classes were created to handle and store all the data necessary for correct functioning of both parts.

At the time of writing of this document, the code is not available publicly but, if it eventually changes to open-source, it will be listed under my *github* account [*], alongside all the relevant documentation. All code was written in *Python3* since the official pipeline was built with it and, as an expansion to it, the same language was used to avoid possible future compatibility issues.

[*] https://github.com/Kamuish

4.1 Data objects

The need to share information between modules required the creation of custom classes, capable of handling the data in a easy and organized way, so that we can access it whenever necessary.

We have three basic classes:

- *Star* class: stores all information of any given star. Such as, the name, centroid position for each image, the masks, the extracted light curve and the outputs of the Gaussian processes.
- *Masks*: The masks are stored inside this class, responsible for storing the information and performing all mask related operations, such as applying the shifting process, as described in Chapter 2.3.3. If the low memory mode is enabled, only the first and the current masks are stored, with all others being discarded after no longer being needed.
- *GP_Data*: Stores the outputs of the Gaussian Processes, which are then used to perform all the analysis and create all relevant figures.

In a higher level, we have a "master" class, named "Data". This class is responsible for the various critical tasks and has a public interface that is used to control the operations performed on the data, as schematized in Appendix B.1.1. A discussion on all functionalities are beyond the scope of this document, but the most important functions are:

- *load_parameters*: Loads all the required data from the provided *fits* files, then creates a *Star* object for each detected centroid and, finally, creates the chosen mask.
- *load_from_txt*: Loads the data from a text file created by a previous photometric extraction. This function allows us to run the GPs without first running the photometry module. It also loads the minimal necessary information that the GPs use and expect to have in this object, essentially mimicking the normal *photometry* functioning.
- *update_stars*: Performs a step in the photometry routine for the specified image. Before exiting this function, checks to see if the updated mask overlaps the region that is not part of the image. If we find a overlap, then a flag of *out of bounds* is set for the relevant star, indicating that the mask in use is not appropriate.

We also have a class-level error flag, that each method checks before running. If it's set, then the method is not executed and a value of -1 is returned.

The main purpose of this class is to be passed to both *Controllers*, inside which it is used to get relevant information and, filled with the results.

4.2 Photometric controller

4.2.1 Main functionalities

As previously stated, this controller is used as a simple interface to the "outside world", that allows us to optimize the photometric algorithm and run it. A schematic of the user-facing interface is given in Appendix B.2.

When this class is instantiated, all the passed parameters are validated, to check if the paths to the needed files exist, check all numeric inputs to see if they are concordant with the expected values and, lastly, validates the configurations for the mask type, initial detect mode and star tracking mode.

As we shall see in Chapter 5, the best configurations for the target star, may not be the same for the background stars, and vice-versa. So, we might have cases in which we wish to have a combination of mask type - initial detection - tracking methods active for the target star and another for the background ones

Thus, all those parameters allow to pass either a global configuration, applied for all stars or, specify different values for the central star and for the background ones, as seen in Figure 4.1 and explained in Chapter 4.5.

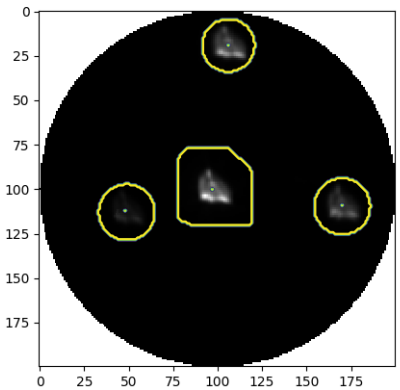


FIGURE 4.1: Usage of both masks at the same time. The *shape* mask for the target star and the *circle* mask for the background stars.

After validating the parameters, if the optimization keyword is set then the process launches immediately, as described in Chapter 4.2.2 . It's possible to disable this optimization, by passing a specific keyword that's disabled by default. The optimization can also be triggered at any time through a specific function. Afterwards, the user simply has to call the **run** function so the typical photometry process runs.

This routine, sequentially applies all the discussed steps until now on each image. For a visual aid, one can refer to Figure 4.2, in which a broad schematic of the entire process is shown.

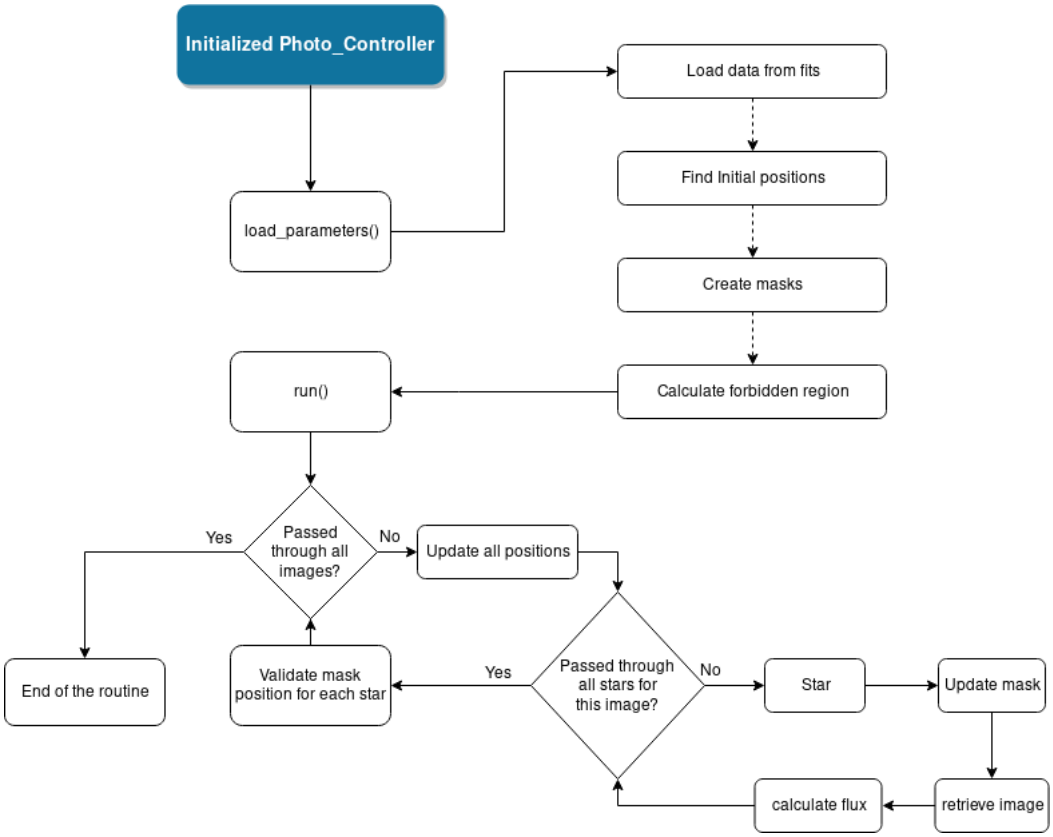


FIGURE 4.2: High level schematic of the photometric routine.

This sequence starts by loading the necessary data from the *fits* files, followed by the initial detection of the stars, depending on the chosen method(s) to be used. Right afterwards, we create the masks and an array with the positions outside the image, so we know in which areas the masks cannot enter. After all of the initialization steps are completed, we pass to the main routine where, for each star in each image, we update the star's position, shift the mask, and, lastly, calculate the flux that passes through it.

The background grid is applied behind the scenes, when requesting an image. However, we can quickly run into a problem: If we increase the images, it's obvious to see

that the memory footprint would increase accordingly. So, if we are not careful, we can quickly run out of memory. Thus, to contain this problem, a low memory mode was devised: When requesting the i-th image in the data set, we remove all images that occurred more than 2 images ago, i.e. at a given point in time, j, we only have in memory the images from j-2 to the end of the data set. The impact and costs of this approach shall be seen in Chapter 4.3.

During this process the *Data* object, discussed in Chapter 4.1, is filled with the various outputs of the different stages.

4.2.2 Optimization Process

The optimization process is one of the most important factors for achieving low noise on the extracted light curves, since it allows us to tweak the mask size. The process is quite simple, since it consists in running the algorithm with different masks sizes, and searching the one that minimizes the CDPP. Through the configuration file, the user has to pass a list with two values: the minimum mask size to be tested and the maximum one.

To speed up this process, it was implemented in a concurrent way, allowing to have multiple factors being tested at once. However, due to Python's *Global Interpreter Lock*, or GIL, the threads do not work in a truly concurrent way but, instead, at any given time only one thread is active and running. It's possible to bypass this lock, by using the *multiprocessing* module, to spawn different processes that will run the algorithm, with the downside of each process being independent from the rest, i.e. in practice it launches multiple instances of python, each running the desired function.

Now that we have a framework upon which we can work, we have to distribute the factors to be tested between all of the spawned processes, run the algorithm and send back the results back to the parent process, which will then merge all the information together and analyze it.

The distribution of values is made so that the workload is evenly distributed by all processes. This can be easily accomplished with a built in function in the *numpy* module, that can split a list into a specific number of different lists .

If the CDPP is deemed invalid, i.e. if the mask went outside of the image region, then we attribute an arbitrarily high noise level, such as 2e7, so it's certain that any valid values will have a far lower noise and thus a valid mask is chosen.

*https://docs.scipy.org/doc/numpy/reference/generated/numpy.split.html - Accessed: 3/09/2019

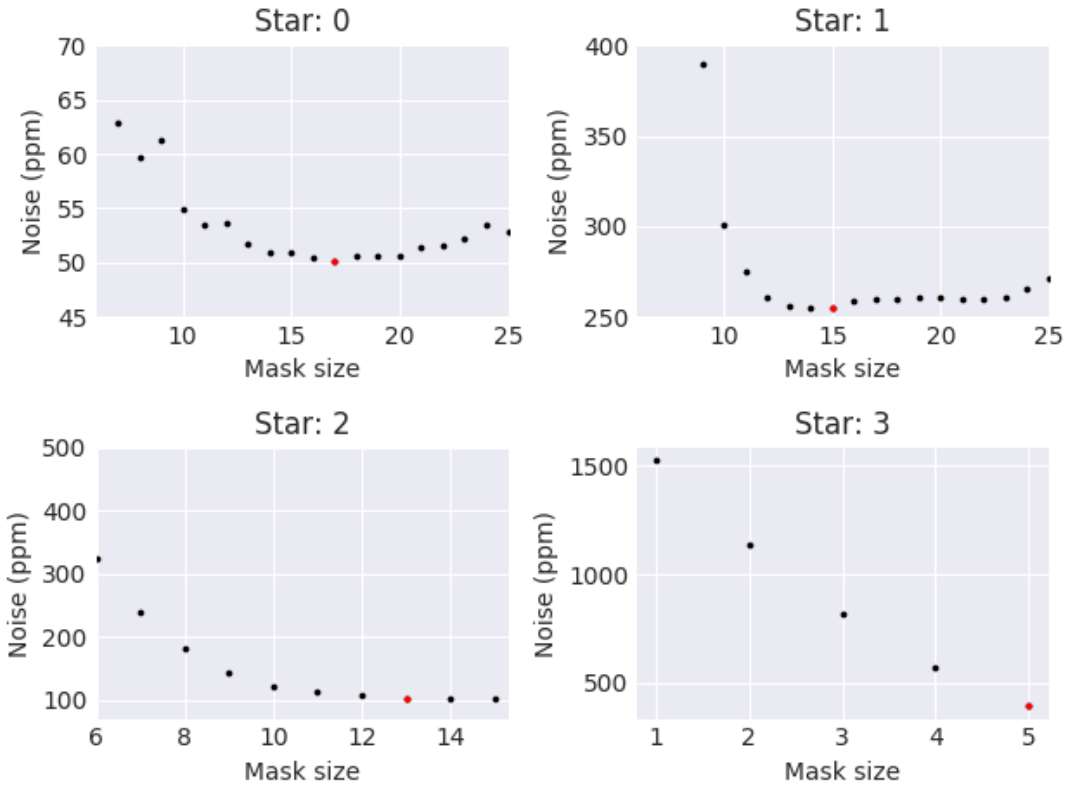


FIGURE 4.3: Evolution of the noise for different mask sizes, during the optimization process, using a *shape* mask with *dynam* initial detection method, *dynam* tracking method and a background grid of 600. The black points represent the value for each mask, while the red ones are the optimal mask size. For Star 2 and 3 we have less values since higher sized masks would leave the image region and thus are not valid.

During early stages of the implementation we noticed that if the configuration values were not the most appropriate ones, the maximum value would be set as the optimal size, even if the noise continued to show a downwards trend past the upper limit of this interval. To avoid these cases, if any star has a mask size that lies within a tolerance range of the maximum value, the search for the optimal size shall continue, now with a lower limit of the previous maximum value and an upper limit of two times the previous maximum. This repetition has a user-defined maximum number, but we found that limiting it to 5 times is enough to find the best sizes for all the stars, when considering background grids smaller or equal to 1800 points.

As a way to also save computational resources, all stars whose optimal mask size lies further away from the upper edge are disabled, i.e. there are no calculations made for that specific star other than the initial configurations.

In Figure 4.3 we see the evolution with the *shape* mask size for all the stars in a simulated data set. The optimization was made over a initial value range of [1-10]. Since the

first two masks found optimal values near the upper edge of this region, the routine was repeated, to see if better values existed further ahead, which was true for Star 0, finding an optimal mask with size of 17. For Star 1 the same was not true, but it still validated the size 15 as the best one. Regarding Star 2 and 3 we easily notice that we have less values, result of being nearer to the image edge and thus bigger sizes are more likely to pass over it. In all stars we notice the previously discussed trend of noise decrease until a minimum value is found, with a slight increase from that point on. In order to avoid recomputing the optimal value each time that we wish to run ARCHI over the same data set, the optimal values alongside the configuration parameters are stored locally. Afterwards, and without any user input, they will be loaded, if the active configuration has already been optimized or, if not, an error is raised, indicating the need to run the optimization routine before proceeding.

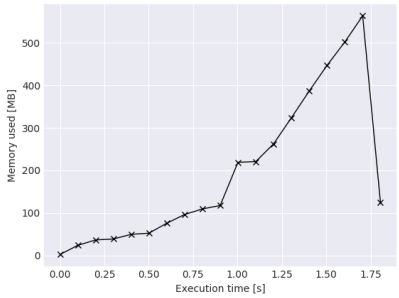
As we have discussed, the design of the *shape* mask does not allow for partial increases in the size. However, that's not the case for the *circle* mask, whose radius can be increased in fractions. So, for this mask, after finding the optimal value, a second optimization step is launched, now searching the values within 1 unit from the optimal one, in steps of 0.1 units. We have found that gains from using smaller steps were not enough to justify the increase in the computational cost.

During the entire optimization process, we attempt to save as much computational resources as possible, by disabling stars and enabling the *low-memory* mode, in order to store only the essential information in memory. As another measure, all checks to the parameters validity are also bypassed during this stage.

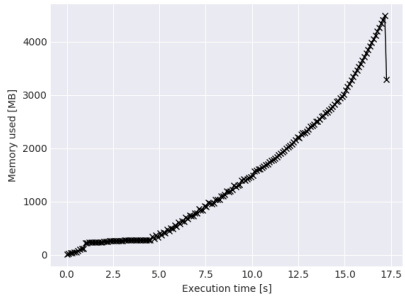
4.3 Benchmarks

Now that the photometric portion of the code was discussed, we shall take a closer look at the computational resources used by it. Since we can't perform memory-based benchmarks on the computational cluster, we shall try to create a general formula and, afterwards, compare it against the results yielded by a memory profiler. All of the results were obtained on a Intel® Core™ i7 6700HQ Processor, with 16 GB of ram.

For this estimation we shall only take into account the objects that will have the greatest impact in memory, which are the images and all the arrays with the same size as them.



(A) Memory consumption for the algorithm in the low memory mode, without a background grid.



(B) Memory consumption for the algorithm in the normal memory mode, using a background grid of 600 points.

FIGURE 4.4: Memory consumption of the algorithm using the normal memory mode. Both tests were made for 4 stars, in a data set with 300 images.

Normal mode

In the normal mode, we have N images, each being a 200x200 array. Since in this mode all masks are stored in memory, we will have the same number of masks, and a single array with the size of an image to store the regions in which we cannot have masks in.

If we use the *nbytes* property from a numpy array, we know that a 200x200 array uses 0.32 MB of memory. If we are using a background grid, each array will have $(scaling\_factor)^2$ more points, which will have an equal impact on the memory size of the array.

So, in Equation 4.1 we have an estimation of the maximum memory in use during the routine, for all stars.

$$Mem_{max} = (1 + N_{stars}) * N_{Images} * 0.32 * (scaling\_factor)^2 \tag{4.1}$$

where $1 + N_{stars}$ correspond to the N_{stars} masks and the corresponding image.

We then tested this memory model for a data set with 300 images and 4 stars in total, as seen in Figure 4.4. By calculating the estimated maximum memory values, with Equation 4.1, we arrive at an estimate of 480 MB for the case without grid, in Figure 4.4a and 4320 MB for the background grid of 600 points, in Figure 4.4b, which are concordant with the graphs.

With these results, we find a worrisome increase in the memory consumption, which can be problematic if we want to run the optimization process with more than a few processes at the same time, or with bigger background grids.

Low memory mode

As a way to mitigate this memory problem, a low memory mode was implemented. While it is active only two masks are stored: the first one, so it can be shifted and the most recent one, so it can be used to perform the photometry.

Taking this into account, we can update the memory model from Equation 4.1, to the one in Equation 4.2:

$$Mem_{max} = (N_{Images} - 2) * 0.32 + 2 * (1 + N_{stars}) * 0.32 * (scaling\_factor)^2 \tag{4.2}$$

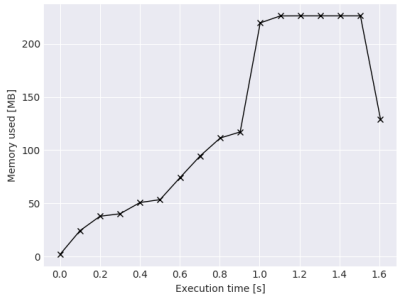
where $N_{Images} - 2$ is the number of non-increased images and $2 * (1 + N_{stars})$ is the number of the masks with increased size and the two images, that also have the increased size.

The peak of memory consumption occurs when we have $N_{Images} - 2$ with the original size and 2 with the increased size. This occurs at the start of the data set and, afterwards, the consumptions should be always decreasing. However, we find that memory consumption stabilizes after reaching the maximum value. When removing an element from a *numpy* array we can do it in (at least) two different ways: we either slice the first element of the array or use *numpy*'s *delete* function . The later, does not actually remove an element, but instead creates a new array without the given values, which as a significant cost associated with it, when considering the computational time. The former, in a practical fashion, removes the element from the array in an efficient way, but the memory is not freed. So, we decided to value more the gains in computational time over the small gains in memory.

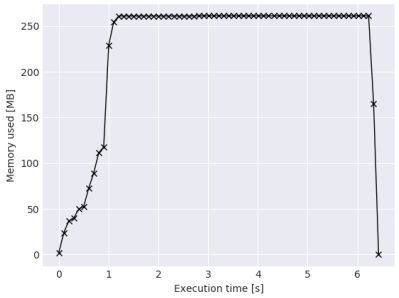
If we test the model, we get an estimation of 99 MB for the run without the background grid and an estimation of 125 MB for a grid of 600 points. However, when we look at the memory profile, in Figure 4.5a and 4.5b we find, similarly to the normal runs, a difference of approximately 120 MB between the estimates and the profiler results, which arises due to the variables used to store all other relevant information, that are independent of the grid's size.

Although the model is not perfect, the different between the profiles from the normal memory mode and the low memory mode are significant and improve the optimization process.

*https://docs.scipy.org/doc/numpy-1.15.1/reference/generated/numpy.delete.html. Accessed on: 29/8/2019



(A) Memory consumption for the algorithm in a low memory mode, without a background grid.



(B) Memory consumption for the algorithm in a low memory mode, using a background grid of 600 points.

FIGURE 4.5: Memory consumption of the algorithm using the low memory mode. The units are MiB, which is almost a one to one conversion to MegaBytes. Both tests were made for 4 stars, in a data set with 300 images.

4.4 Gaussian Processes controller

Following the *Photo_Controller* design principles, we created a simple interface, schematized in Appendix B.3, that accepts a *Data* object as input and, with a single command, lets the user apply the GPs. This application can be done over all stars or, over a specific one.

Before being able to run the GPs, we have to give it some information about the star:

- Mass;
- Radius;
- Limb darkening type;
- Limb darkening coefficients.

with the limb darkening information following what was specified in the documentation of the *batman* module [67]. Furthermore, if the expected values of the fitted parameters are known, we can also provide them, and errors are calculated and marked on the *corner* plot, thus allowing us to see if the true values for each parameter are within the distributions.

4.5 Inputs

Now that we have seen how the different parts of ARCHI work individually, we will now see how we can configure it. The configuration is made through a *yaml* file, that is loaded

with the *Photo_Controller* initialization. A properly commented version of this file is given in Appendix E.1 and, in here we will see, with a closer look, some of the parameters.

As previously mentioned, the target star and the background ones can have different configurations from one another. To do so, when specifying the desired mask, initial detection mode or star tracking mode, two different valid values can be put there, separated by a plus sign, e.g. "shape+circle" would produce a shape mask for the target star and a circular one for the background stars.

Ideally, when using this kind of configuration, we could be able to search the file with the previously calculated mask sizes, optimized for each method, and search there for the values for the target and background stars. Unfortunately, due to time constraints it was not possible to do so and thus, even if the different components have already been optimized individually, the optimization routine must still be called for the chosen combination.

Furthermore, ARCHI can load all of the necessary data from the DRP's output folder, without having to manually enter the paths for the files that are used. Instead, and assuming that the DRP's folder structure stays the same, it's only needed to provide the path to the folder that contains those outputs. When any of the controllers is initialized, the configuration values pass through a validation stage, in which the inputted values are compared against the expected ones. If any discrepancies are found, then ARCHI's execution halts and it returns to the user a list of the wrong parameters.

4.6 Outputs

The last module in the ARCHI project, is the data storage one. This module accepts a **Data** object as input, assuming that it has any kind of data inside it. Before we can do data-specific processing, we have to make sure that the we have an organized folder structure, as in Figure 4.6 to store all the graphs.

Most of the time the code should be executed in the computational cluster, *Supernova*, especially if the GPs are going to be used or background grids with more than 600 points need to be optimized. The workload manager used in this cluster, SLURM [81], makes use of *jobs* to submit code, that then waits to be executed at a convenient time. Each *job* is attributed an unique identifier, *jobID*, that allow us to distinguish between two different jobs. This unique identifier will allow us to create a new folder inside which we shall store the outputs.

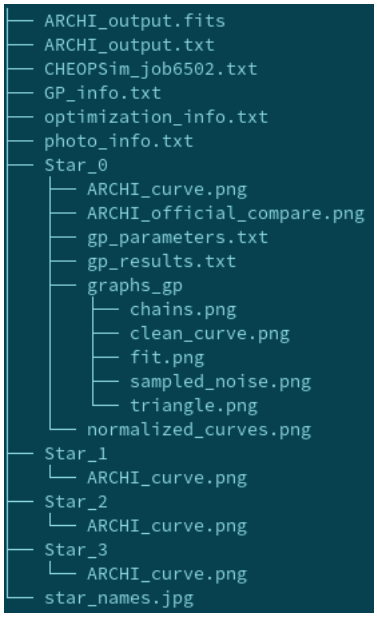


FIGURE 4.6: Folder structure used to store all the data extracted from the **Data** object.

In this directory, we find the name of the folder that comes out of the official pipeline, "CHEOPSim_job6502.txt". In it, we find informations of the chosen light curve (from the 4 available) to compare our data against. Alongside this information, we also have the *mjd* time corresponding to each point and the *roll angle* of the satellite.

It's also on this folder level that the outputs of the pipeline will be stored, under the name "ARCHI_output" followed by the correct file extension. In the "optimization_info" we have the noise, for each star and each mask size. Take note that in this file the sizes are not by numerical order, but instead by the order from which they leave the various processes used during ARCHI's optimization routine. The last text file, "photo_info.txt" stores informations related to the run, as well as the noise of each curve. Furthermore, we have a image, named "star_names.jpg", that is used to map each star to the corresponding name.

To simplify the representation of the structure, we only included data on the target star. If more were in use, the data inside any star_ folder would mimic the one seen inside Star_0. In this folder we have outputs from both photometry and Gaussian processes. In 'ARCHI_curve' we have the light curve from the corresponding star. In 'ARCHI_official_compare' we compare the ARCHI light curve against the one extracted from the CHEOPS official pipeline.

Afterwards we see a 'gp_parameters.txt', that has the estimated values for each parameter, alongside the uncertainties for each of them and, if the true values were provided, a

calculation of the error, and a "gp_results.txt" in which the corrected light curve and the noise samples are stored. Lastly, we have a folder - "graphs_gp" - to store all images that result from the Gaussian Processes:

- chains.png - Chains for the production stage;
- fit.png - Normalized light curve, overlaped by the best fitted model;
- sampled_noise.png - Median of the noise samples from the GP;
- clean_curve.png: Normalized light curve without the noise that was sampled from the GP;
- triangle.png - *corner* plot with all the parameters from the model and from the Kernel;

Chapter 5

Discussion

5.1 Photometric comparisons

Now that the implemented methods have been discussed, we have to characterize them and see how they fare against each other and against the results from the DRP.

At first, we will analyze the so-called "normal" runs, i.e. without using the background grid. Afterwards we shall introduce them, and see how they impact the light curves.

Due to the sheer number of possible combinations of mask type - initial detection - center tracking - background grid, the graphics and tables are mostly presented in Appendix D, whilst in this Chapter only some of them will be looked into.

In this Chapter, we shall study three different Data Sets:

- Data Set A: features a single transiting planet in the central, main target star, and will be mostly used to compare ARCHI's methods against themselves;
- Data Set B: has transits in both the target star and Star 1 and will be used to compare ARCHI's curves against the ones from the DRP;
- Data Set C: features transits in the target star, Star 1 and Star 2 and shall be used to validate transit detection on the background stars.

For more information on the Data Sets, Appendix C has a compilation of the star's parameters alongside the inject planets.

This analysis will consist in the comparison between the light curves, which is afterwards complemented with an analysis of the CDPP and the uncertainties in each light

curve. Unless explicitly stated that we are using the DRP's CDPP algorithm, we shall make use of the one ported from *Kepler* mission, always with a 30 minute time scale.

In order to differentiate the combinations of initial detection modes and star tracking modes, we shall use the convention $< \textit{Initial detection} > - < \textit{Star Track} >$ to refer to them, where "Initial detection" refers to the methods presented in Chapter 2.1 and "Star Track" to the ones presented in Chapter 2.2.

As discussed beforehand, the data is simulated using the CHEOPS official simulator tool, and thus the Data Sets emulate typical conditions. Furthermore, the data sets were crafted with a different number of planets and planet parameters, in an attempt to try to characterize the methods in a broad set of conditions.

5.1.1 Data set A

5.1.1.1 Normal runs

Our analysis will start with the *shape* mask, in Figure 5.1, where we see that, for the target star, there are minimal differences in the shape of the light curves, with the most major ones being in the flux level.

However, when comparing the noise of each light curve, we find big discrepancies. As one can see in Table 5.1, the *static* method yields the worst results for the target star. Both the *dynam* and *offsets* method are capable of presenting similar curves, assuming that the initial detection methods are equal. It's noteworthy that the *dynam-offsets* combination manages a 2 ppm reduction when comparing against the *dynam-dynam*.

Regarding the mask size, we see that there isn't much difference between the methods, although smaller sizes are preferred, since they pick up less background noise.

Looking globally at all light curves, in Figure 5.1, and the mask's size, in Table 5.1, is clear what the effect of the mask size is over the flux level. Smaller masks result in smaller flux levels, as seen with the *dynam-dynam* case: For Star 0,1 and 2 this mask used the smallest grid and, consequently has the lower flux level. If we look at the other end of the spectra, bigger masks are always the curves with the highest flux.

For the background stars we find that the *dynam* star tracking method surpasses the other two, by a wide margin, not only in the noise, but also in the mask size, with the sole exception of Star 3, as we shall see later on.

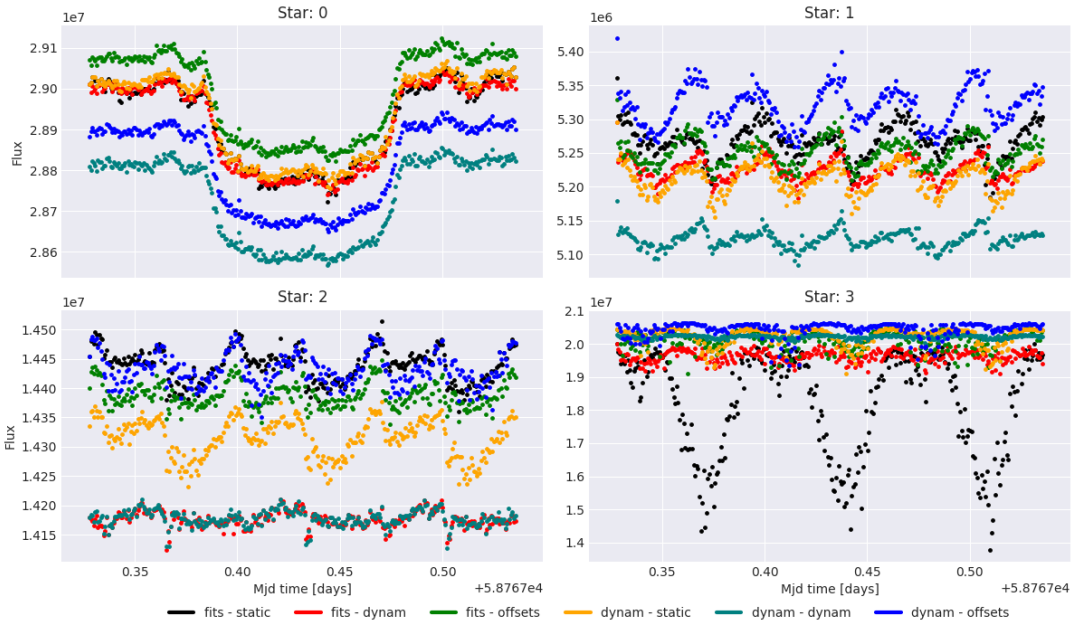


FIGURE 5.1: Light curves obtained with all the combinations of methods, using a *shape* mask and a background grid of zero.

TABLE 5.1: Table with the noise, in ppm, for all the Light curves (seen in Figure 5.1), using a background grid of 0 and a shape mask. In green we have the masks with the lowest noise and the corresponding sizes. Contrastingly, in red we have the masks with the highest noises and their sizes.

Methods		Noise (ppm)				Mask's size			
Initial	Track	0	1	2	3	0	1	2	3
fits	static	83.8	331.8	173.9	3797.7	14.0	10.0	10.0	1.0
fits	dynam	63.8	273.5	122.2	1138.9	14.0	8.0	6.0	1.0
fits	offsets	56.9	292.7	174.6	1873.5	15.0	9.0	9.0	3.0
dynam	static	73.9	310.9	166.1	793.5	14.0	8.0	8.0	3.0
dynam	dynam	54.8	251.7	112.7	281.1	11.0	5.0	6.0	2.0
dynam	offsets	52.6	335.5	199.6	518.3	12.0	12.0	10.0	4.0

If we look closely at the results from the *static* and *offsets* star tracking methods, we find a sinusoidal behaviour. This periodic pattern is also seen on the *dynam* method, although not as noticeable.

Interestingly, in Star 2 we can clearly see that the background signal has different periods, depending on the star tracking method applied. The *dynam* star tracking method has a signal with a period roughly two times greater than the other ones. Since the period is equal for the *offsets* and *static* star tracking method, we can attribute this difference to the centroids oscillations during the images, as we have seen in Chapter 2.5.3.

Remembering back from Chapter 2.5.3, the methods based on rotating the points, suffered from periods in which the estimations fell far away from the star's center. Thus,

since the *dynam* method is not dependent on the rotation angle, we can assume that it's affected by less systematics caused by the rotation. The root causes behind this signal will be explored further along this work, in Chapter 5.1.2.

Returning to the comparison of the applied methods, we see that with a *dynam-dynam* combination will allow the usage of smaller masks, thus picking up less background and reducing uncertainties. For Star 2, the initial detection has no impact on the mask size when using a *dynam* star tracking method.

The last star in this data set, Star 3, shows the biggest differences between the methods. In the first place, it's important to note that due to the closeness to the image's edge, the mask optimization process was restrained, leading to the creation of a sub-optimal mask. Since there is no workaround for this limitation, we will now see how the configurations impact the mask size, noise and overall shape of the constrained light curve.

A first glance at the light curves lets us see that the combination *fits-static* yields the worst results, with the easily seen dips in the flux. Similarly, the *dynam-static* combination also shows a slight dip in the flux, although less perceptible. The difference between these two curves stems from the fact that the initial detection methods, for both of them, are different, and thus estimate different positions for the centroid. Due to the nature of the *static* method, this small difference will propagate throughout the images and result in the seen difference. Similarly, the mask size is also different for both methods, with the *fits-static* only allowing a mask of size of 1 , while the *dynam-static* allows a size of 3.

Contrarily to what was said for the other stars, for Star 3 a smaller mask is not necessarily the best one. As we have seen, the *static* method estimates points that oscillate within the star. This effect, coupled with the distance from this star to the image's edge, are enough for the mask to go outside the bounds. Taking this into account, one could expect that the best curve would then be the one that maximized the mask size, which is not entirely accurate. The biggest mask is produced by the *offsets* method that, as we have seen before, is also impacted by the rotation angle of the satellite. The impact can be such that even though a bigger mask is in use, a significant portion of the PSF still remains outside the mask.

Globally speaking, the *dynam* initial detection method yields lower noises than the *fits*, presenting a considerable difference. Regarding the tracking method, we see that the static one is the worst, for the target star. We also see that the corrections introduced by

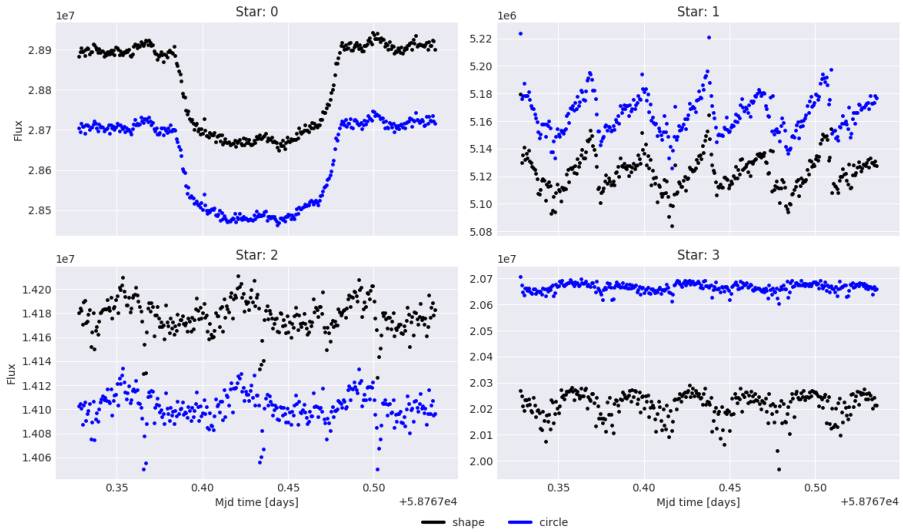


FIGURE 5.2: Comparison, between the light curves from the *circle* and *shape* mask, using the best method from the light curves.

TABLE 5.2: Comparison, between the light curves from the *circle* and *shape* mask, using the method combination that minimizes the noise, for each star.

Mask type	Noise (ppm)			
	Star 0	Star 1	Star 2	Star 3
shape	52.6	251.7	112.7	281.1
circle	52.4	250.0	107.8	102.0

the *offsets* method are not working as well as expected, since the noise is, in some cases, worse than the one obtained with the *static* method.

For the *circle* mask, in Figure D.2, we once again find the same relative results amongst the possible combinations, with similar noise levels and light curve's shape, including the periodic signal on Star 1 and 2. However, it's noteworthy that the *fits* initial detection method is preferred for the background stars, whilst the best combination possible for the target star remains the same.

Now that we have seen that both masks exhibit almost the same behaviour, we can see how they fare against each other. In Figure 5.2 and Table 5.2, we have a comparison between both masks, using the best method combination for each one of them.

When comparing the two masks, we find almost minimal differences in the noise, except for the outermost star, in which the *circle* mask yields less than half of the *shape* mask's noise.

Since both curves have different flux values, it's best to look at the relative uncertainties, i.e. the uncertainties divided by the flux, so that we know how much of the signal they represent. In Figure 5.3, we see that despite the lower flux level of the *shape* mask,

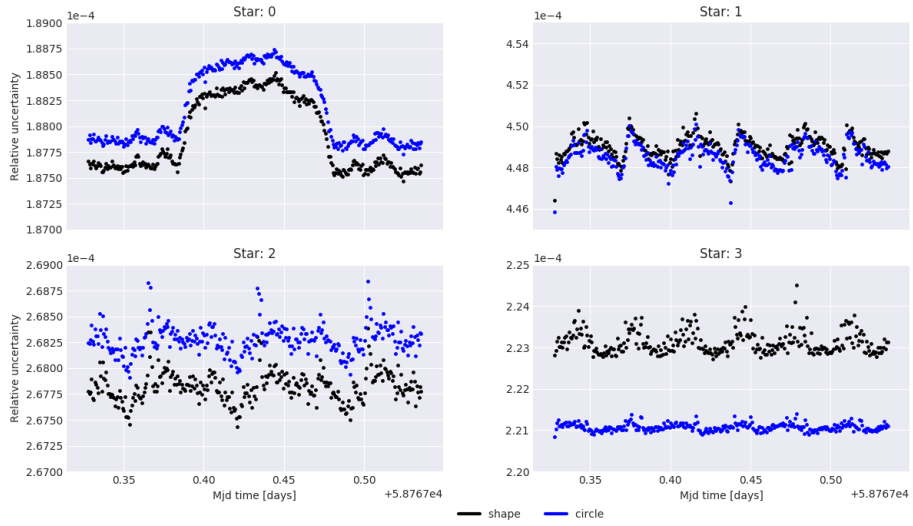


FIGURE 5.3: Comparison of the uncertainties, divided by the flux, for best method combination when using a *circle* mask and a *shape* mask.

it has higher uncertainties, than the *shape* mask, for the target star. For Star 1 and Star 2, the uncertainties in each method are quite similar, despite the difference in the flux level. From the target star, we can also see that the uncertainties are dominated by photon noise, since we can still see the shape of the transit.

5.1.1.2 Background grids

Having seen how the different methods affected the Light Curves, without the background grid, we will now see if it is capable of improving the photometric precision.

For this Data set, we will compare 5 different background grids: 600; 1000; 1400; 1800 and 2200. The tables with the noise, in ppm, obtained for each method are provided in Appendix D.1, alongside the curves for a grid of 600, using both masks.

Figure D.1 and D.3 show us that a background grid of 600 does not produce obvious alterations on the curves, although the noises do not show the same.

In Figure 5.4 we see that the usage of the smallest possible grid, of 600, translated into a decrease in the noise level, for most combinations of initial detection and star tracking techniques applied in the target star.

Despite the positive evolution of the noise for all *dynam* and *offsets* star tracking methods, we notice that the *static* actually worsens with bigger grids. This effect may be related to the fact that in the bigger grids the centroid's coordinates can oscillate more between pixels and thus the mask is worsened by the rotation applied in each frame.

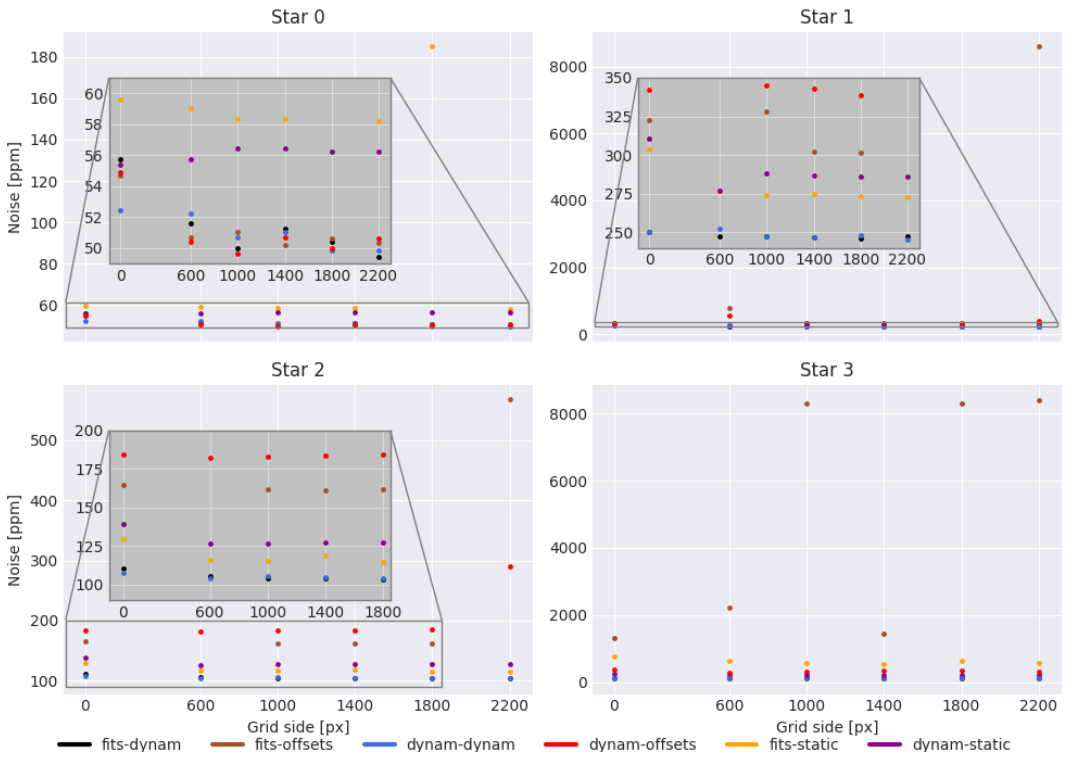


FIGURE 5.4: Behavior of each combination of initial detection and star tracking method with the increase of the background grid, using a *circle* mask.

It's also noteworthy that this method yields a sharp increase in the noise level for the target star, when using a grid of 1800, which may be caused by a possible failure of the optimization process, since the same behaviour is not observed for the grid with 2200 points.

For the background stars, the background grid has almost no effect on the noise level and, in some cases, bigger grids result in tracking problems, as seen with the *static* method for the target star. The best method, for these stars, is the *dynam*, that consistently yields the lowest noise levels. The rotation based methods, i.e. *static* and *offsets*, both present higher noise levels and, under some circumstances sudden increases, as seen with the *offsets* method for Star 3.

Focusing back on the target star, we notice that after the background grid passes the size 1000, we find both negative and positive gains in relation to the previous grid. The oscillations on the noise level may be an indicator that we have reached the lowest possible noise using the developed techniques and thus the noise is converging to a value in the 47 to 51 ppm region.

The fact that no improvements are seen for the background stars may indicate that due to their non-static position, they are not impacted by the background grid and that a

normal image is enough to reach the maximum, or near maximum, precision.

In the normal grid, when converting from the determined centroid's position to a grid coordinate, we loose some information and thus, two different tracking methods could give the same result due to the approximations. With bigger grids, even though there are still approximations, the conversion will be more accurate, thus impacting the rotation-based algorithms. This effect, is the most likely cause of the increased noise found for the rotation based methods, with the increase of the background grid.

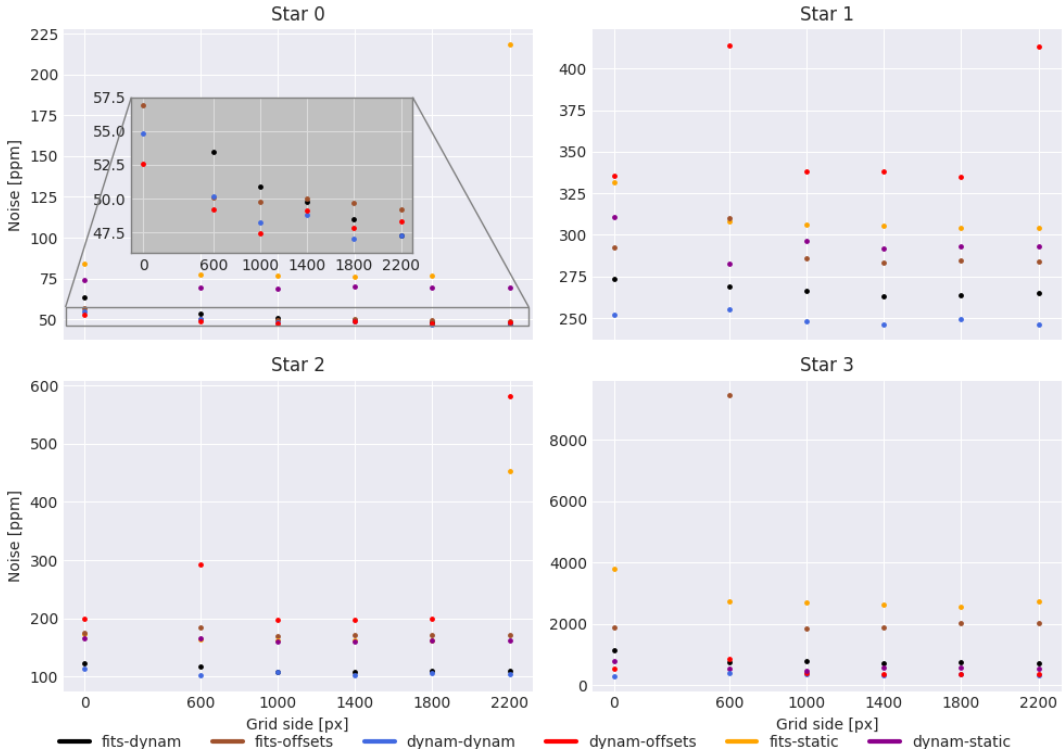


FIGURE 5.5: Behavior of each combination of initial detection and star tracking method with the increase of the background grid, using a *shape* mask.

Similarly to the *circle* mask, the *shape* mask shows the same patterns, Figure 5.5, thus also validating the hypothesis that a minimal noise plateau exists. Albeit it is still possible to find some of the noise spikes for the rotation based methods, they are not as frequent as before.

The introduction of the background grid has an overall positive effect on the light curves, as seen in 5.6. The best combination of initial detection and star tracking methods allows a decrease in noise, for most stars. However, Star 3 does not show the same trend, but instead the noise increases with the background grid. As referenced before, the closeness to the image's edge is a possible explanation for this effect.

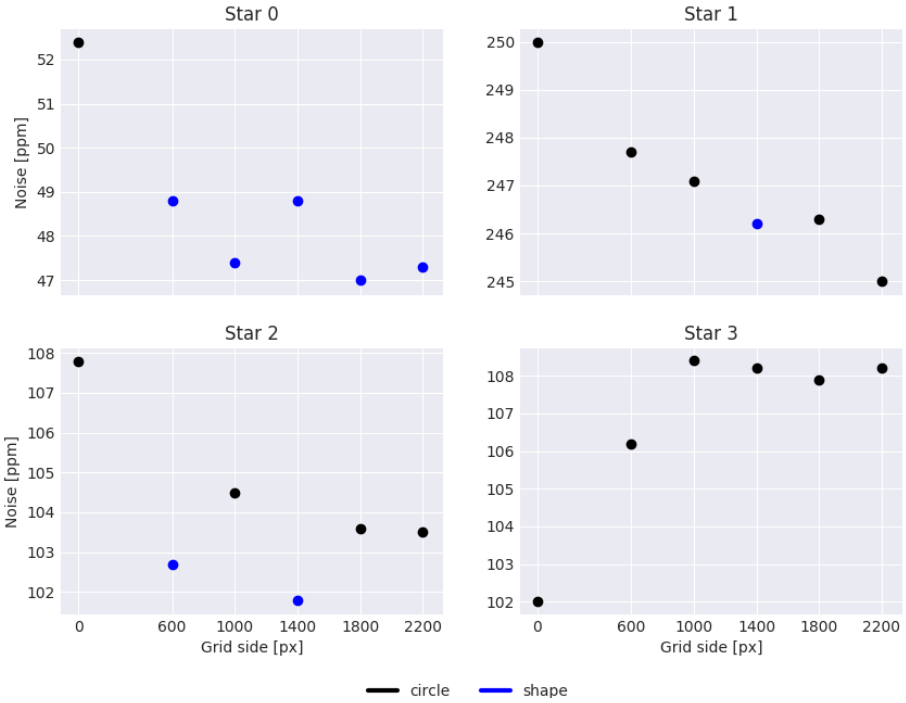


FIGURE 5.6: Evolution of the achievable minimum noise with the increase of the background grid, for Data Set A.

While the *shape* mask is the best for the target star, representing a slight improvement when comparing against the *circle* one, the *circle* mask is the best for the background stars.

Lastly, we can also see how the uncertainties behave for different sizes of background grid. For this test, we shall use a *shape* mask, with a *dynam* initial detection method and a *dynam* star tracking method.

In Figure 5.7 we find that, for the target star, only one grid achieves a decrease in the uncertainties, relative to the flux level. With the sole exception of the grid with a size of 1400, all others increase the uncertainties by a small amount. It's noteworthy that the grid of 1400 manages to introduce a significant decrease, when comparing against all others.

For the bigger grids, 1800 and 2200 we find a slight increase, when comparing against the case without a background grid. Since those differences are small, they could be introduced by the approximations made when calculating them, as described in Chapter 2.7.

For the background stars, we do not find noteworthy differences between the grids, except for Star 1, in which the largest grid increases, by a small amount, the uncertainties.

When comparing the uncertainties from the background stars against the target, we find that, in the background stars the uncertainties are larger, in relation to the signal. This

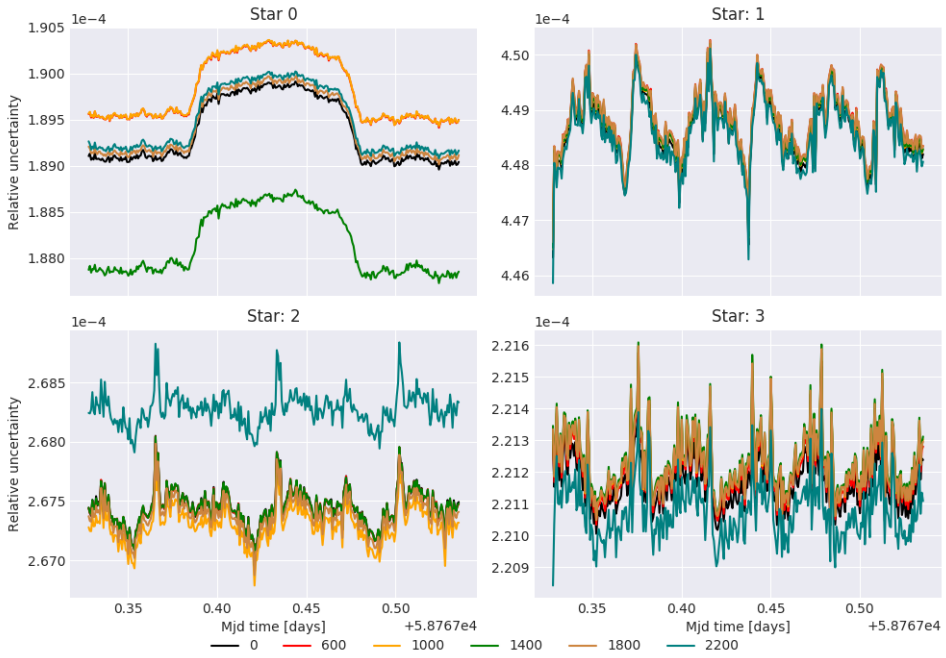


FIGURE 5.7: Evolution of the uncertainties, relative to the flux level of the corresponding LC, with the increase of the background grid.

may be due to them being fainter, and thus the signal is weaker, or due to the fact that the values used in the calculations are estimates and thus are not entirely correct for the stars in the background.

5.1.1.3 Comparison against the DRP

Now that we have compared all of ARCHI's internal options, we can compare the best configurations, against the *OPTIMAL* curve from the DRP. We notice that without using a background grid, ARCHI can achieve a slightly lower noise level than the DRP, as seen in Figure 5.8. After introducing a background grid of 1800, which is the one that both minimizes the noise, we achieve a light curve with a noise level approximately 5 ppm lower.

Unfortunately, due to a bug on the DRP, this data set did not store the calculated uncertainties, thus limiting the comparison between ARCHI and the DRP. For the other data sets, this bug was not present and we will be able to see which pipeline yields the lowest uncertainties.

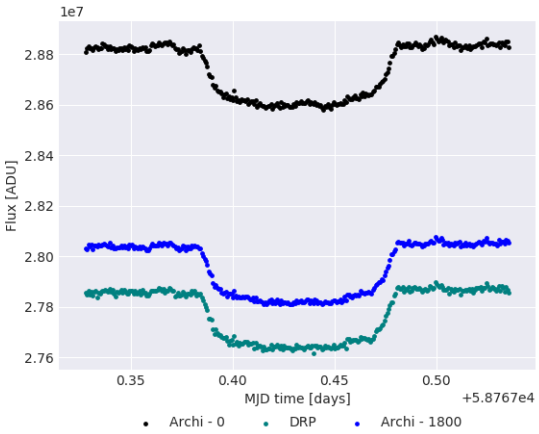


FIGURE 5.8: Comparison between DRP's *OPTIMAL* light curve and ARCHI's light curve, without using the background grid and with a background grid of 1800.

CDPP type		DRP	*Kepler*
ARCHI	0	199.1	52.6
	1800	196.8	47.6
DRP		200.8	52.9

TABLE 5.3: Noise calculation using the DRP's CDPP algorithm, for both the DRP's OPTIMAL light curve and the best ARCHI's light curve, using DRP's CDPP algorithm and the CDPP algorithm from the *Kepler* mission.

Lastly, we can compare the DRP's CDPP algorithm against the one ported from the *Kepler* mission. In Table 5.3 we notice that both with and without the background grid, that the ARCHI produced light curves present lower noise values. The DRP's version also shows the decreasing trend in the noise, albeit smaller for larger time scales.

It's fair to conclude that ARCHI's performance for the target star is comparable to the one from the DRP, when utilizing the DRP's implementation of the CDPP algorithm. Even when applying the ported DRP from the *Kepler* mission, the differences are not as evident, but ARCHI still show a lower noise level.

5.1.1.4 A closer look into the CDPP

As referenced before, in Chapter 2.6, the application of the CDPP algorithm to the light curves removes it's shape. In the target star's case the *Savitsky-Golay* filter will remove the transit, to calculate the noise.

In Figure 5.9 we see that the SavGol filter can reliable remove the transit from the light curve, thus allowing us to only characterize the noise present in it. For the background stars, we do not have transits, but once again we find, in Figure 5.10, that the light curve shape is removed with the filter.

Although the sinusoidal signal is properly removed, the noise value does not correspond to the data in the signal. As we shall see later on, in Chapter 5.1.2, this signal is created by contributions from the target star during the satellite's rotation. Although this

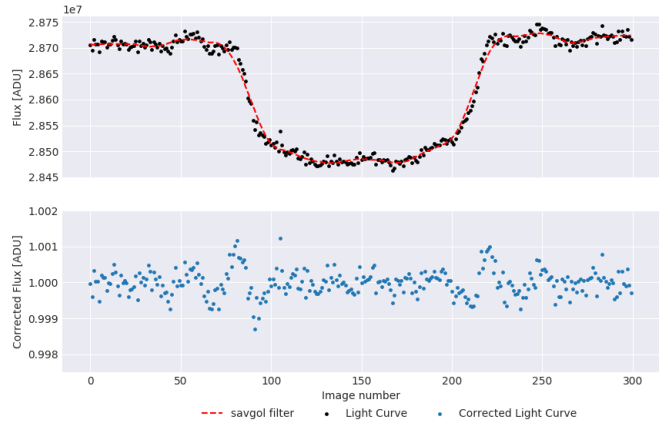


FIGURE 5.9: SavGol filter applied on the light curve from the target star, for Data Set A.

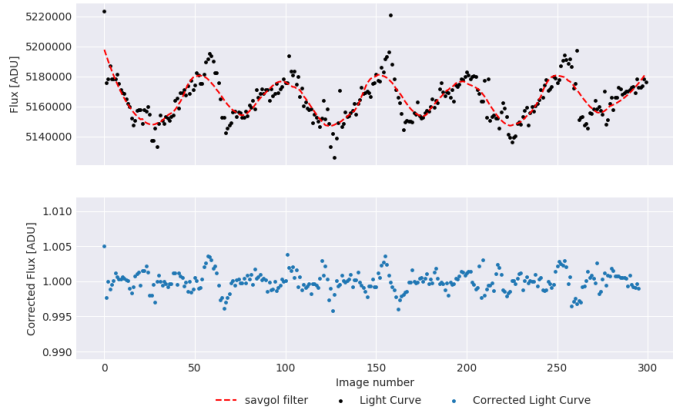


FIGURE 5.10: SavGol filter applied on the light curve from Star 1, in Data Set A.

TABLE 5.4: Comparison between the CDPP with the SavGol filter and the CDPP implemented in the DRP, for Data Set A, using a *shape* mask, a *dynam-dynam* combination and without a background grid.

	Star 1	Star 2	Star 3
SavGol	251.66	112.7	281.11
DRP	415.48	162.65	429.29

CDPP value is still a valid measure of the light curve quality, it does not correspond to the reality since we are removing meaningful contribution in the signal.

If we now compare this version of the CDPP against the one from the official pipeline, as in Table 5.4, we see that the DRP's version of the CDPP algorithm yields far higher noises, as expected.

5.1.2 The sinusoidal signal from the background stars

As we are working with the images corrected by the DRP, we expected to see white noise or, if they existed, transits, in the background stars. Instead, we have found a well defined signal in them. On top of the existence of this signal, we see different behaviours for the different stars, with different periods and amplitudes in all of them.

The analysis applied in this section will use a *shape* mask, with the initial detection being made with the *dynam* method, and the star tracking with also the *dynam* method. In order to better study the signal in question, we decided to use Data Set A, since it's the one that has less gaps in time, thus allowing us to better see the signal evolution with both the observation time and the satellite's rotation.

Taking into account the background stars movement throughout the observation, we know that after a certain period of time, approximately 100 minutes, the stars complete a lap around the target star. Since in this data set the background stars give $\pm$ 3 laps around the target one, we know that the cause behind the peaks occurs in each lap, thus indicating a problem that may arise from some imperfection in the data correction procedure applied by the DRP.

Furthermore, if we fit a sinusoid to the light curves, we find two different periods, as shown in Figure 5.15. For star 1 and 3, we find a period of 0.03 days, which is roughly 50 min and, for Star 2, a period of 0.07 days, or roughly 100 min. From Chapter 1.2 we know that the satellite takes about 100 min to complete a full revolution. Thus, we have a signal that repeats every complete turn of the CCD, and two others that can be seen twice during a single turn.

5.1.2.1 Relative rotation

In order to start understanding the origin of the signal, we can study each star during it's first lap, and attempt to see if the peaks are found for the same locations within the CCD. By defining the origin of our angles, $\theta = 0$, as the vertical line of the central star, as schematized in Figure 5.11, we can set a starting location to start analyzing the flux.

Since only one of the stars start in the zero location and taking into account that we wish to compare the flux of each star in the same angle, measured from our reference, we have to wait until each star reaches that point.

Looking the the light curves from the first lap, in Figure 5.12, we notice that Star 1 has peaks for $\theta = 90$ and $\theta = 250$. Star 2, also shows its only peak for theta near 250, although

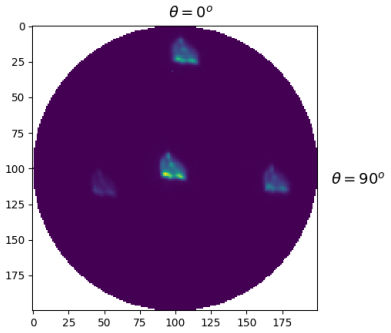


FIGURE 5.11: Orientation of the angles for study the stars in the same points.

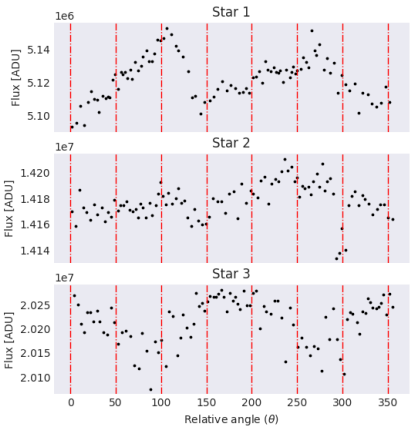


FIGURE 5.12: Light curves for all of the background stars, during the first lap around the central star.

not as noticeable. However, we find that the singular peak from Star 2 is concordant with the one from Star 1. Contrarily to both other stars, in Star 3, the flux hits its minimum values when the two others are in their maximum value.

Since this analysis did not bring us nearer an explanation, we shall now explore multiple alternatives. We shall start by looking at CCD wide corrections applied by the DRP: *background*, *Flat Field* or even the corrections for bad pixels. Afterwards, we will look into the possibility of it being explained by contaminations from the target star.

5.1.2.2 Contributions from the background

The first, and most obvious answer to this problem, would be some leftover contributions from the background, that were not removed during the DRP's operation. Extracting the background level, as in Chapter 2.7.1, we can compare it against the light curves levels, as in Figure 5.13.

If we start by looking at background values, we see two different peaks, with different amplitudes. At the time of this analysis, the DRP's background estimation is made using an aperture, outside which an histogram is calculated and, afterwards, a Gaussian is fitted, in order to estimate the background contribution per pixel. Since in this data set, we have a close-by star, near the target star, there is a realistic chance that this star is, periodically, weaving inside and outside this aperture, and thus influencing the background per pixel. There is also a chance that the peaks are explained by the simulated stray light, although there is no way to prove it with the current data sets. In the future, when such

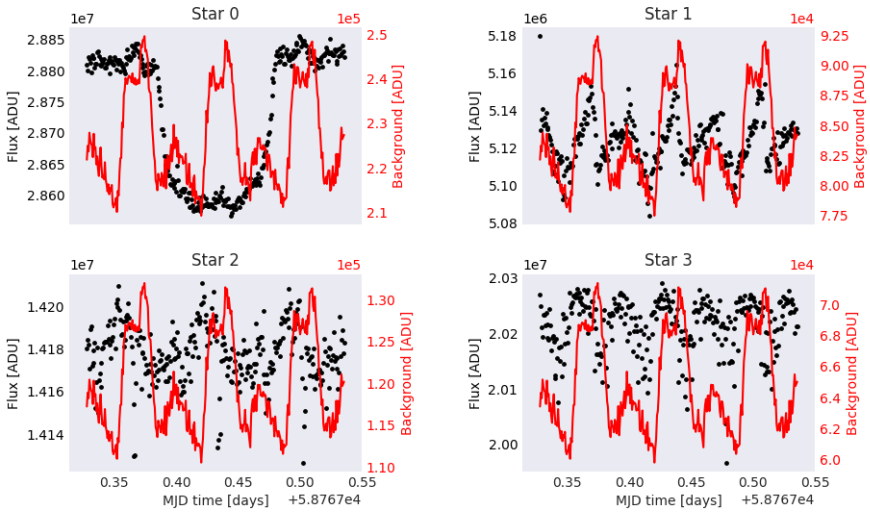


FIGURE 5.13: Comparison between the light curves and the background level, for each star. In black we have the light curve and, in red, the background level.

information is added to the simulated data sets, we can run more extensive tests. A more in-depth study of the background and the applied corrections could be made but, it's outside the scope of this thesis.

For Star 1 and 3 we can find a strong correlation between the background and the periodical oscillations, albeit not as much for the second star. However, near the secondary peaks, we can still see some anomalies in the light curve. When comparing the scales, we notice that amplitude of the background signal is far lower than the amplitude of the oscillation.

Since the background correction was built with only the target star in mind, the background estimation takes into account the background stars and, as we have mentioned, there is the possibility of stars entering and exiting the aperture region throughout the images. Even though the oscillations are not explained by variations in the background, some of the noise is introduced with it, due to the under/over estimations introduced by the rotating background stars.

5.1.2.3 Flat Field

Since the oscillations are found in all of the background stars, we decided to check another correction applied over the entire image: the Flat Field. In here we shall apply the masks, determined for each star, over the Flat Field and sum the entirety of the Flat Field inside the mask. If we do it for all masks, i.e. the ones determined for each image, we can see

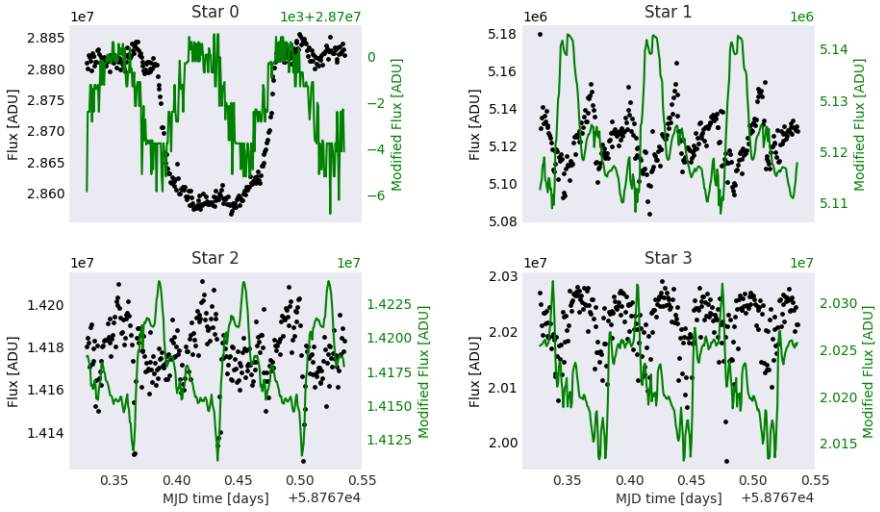


FIGURE 5.14: Comparison between the light curves and the Flat Field level, for each star. In black we have the light curve and, in green, the Flat Field, inside each mask and multiplied by the median of the LC.

how the Flat Field varies on the areas over which the background stars pass. As a way to quickly estimate the possible impact of the Flat Field, we can multiply it by the median of the light curve, to see if the variations are concordant with the periodic signal in the data.

Since we are using simulated data sets, we can use the truth values that were injected in the simulation.

When looking at the Flat Field we notice some similarities to the background shape, with some big peaks, interluded with small ones in between. Once again, it's hard to find a correlation between the two signals, since there is no clear pattern linking the variations in the Flat Field with the signals found on the light curves.

However, for Star 3 we can find a small correlation between the minimum values, despite the rest of the signal not matching as nicely.

Once again, it's not clear that the Flat Field is being applied correctly to the background stars, but since we have not found proofs of it's impact, we assume it to be negligible.

5.1.2.4 Bad pixels

After discarding the background and the Flat Field as the probable sources of the signal, we performed a study of the number of bad pixels inside each mask, during the observation period. We mainly searched for two types of bad pixels: "partially dead pixels" and

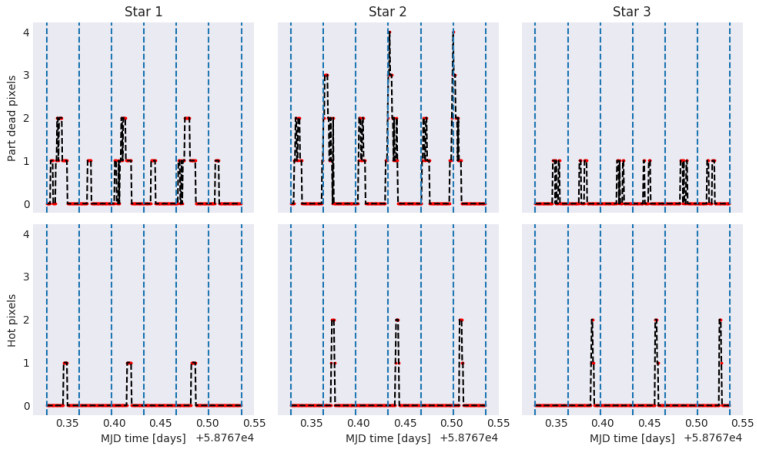


FIGURE 5.15: Number of bad pixels, for each star, inside the mask used for each image. The black curve is the number of bad pixels and, the blue lines were placed in equally spaced positions, separated by 50 points, i.e separated by half of the satellite's rotational period.

"hot pixels". We decided to exclude the telegraphic pixels, since none crosses the path of Star 1, thus invalidating it as the possible cause.

A first look at Figure 5.15 shows that the number of partially dead pixels is approximately periodic, with a peak every 50 images, which is roughly 50 minutes or, half of the satellite's orbital period, which is concordant with the found signal. The masks also cross the hot pixels, although only once per each lap, almost at the same time as the partially dead pixels.

Comparing the partially dead pixels throughout time and the light curve of each background star, Figure 5.16, we fail to find a meaningful correlation between the two graphs. Although some maximums are coincident, some are peaks of the light curve are found with the a minimum value of partially dead pixels inside the mask.

Without a strong correlation between the signals, we reach the conclusion that the dead pixel correction is working as expected, and that they have no significant impact on the light curves.

5.1.2.5 Cross contamination from the central star

Due to the size of the target star's PSF and due to the closeness of the nearby stars, we can hypothesize that during the rotation, the masks from the background stars enter the target's PSF, picking up flux and thus impacting the results.

As a preliminary test, we shifted the masks in such a way that they would only pick up background, without the stars. This shifted mask, henceforth called as "trail mask", had

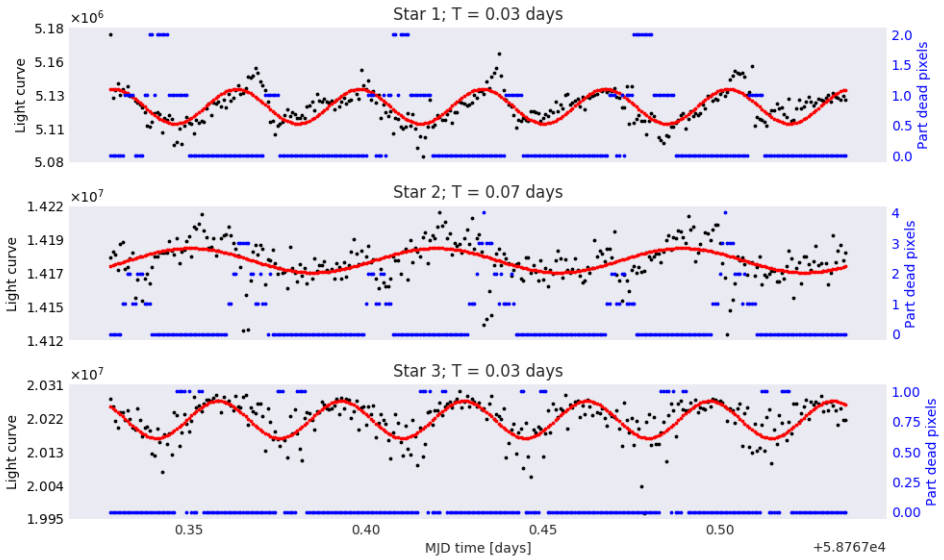


FIGURE 5.16: Number of bad pixels, for each star, inside the mask used for each image. The black curve is the light curve for the star, the red curve is a sinusoid fitted to the data and, in blue, the number of partially dead pixels.

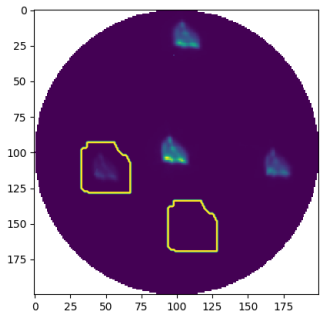


FIGURE 5.17: Example of a shifted mask, to study contaminations from the target star.

to be in different positions for each mask, since we want to avoid running into/passing near other stars. So, the only obvious region of the image available was the bottom, as exemplified on Figure 5.17.

Shifting the three masks to the empty area bellow the target star, as depicted in Figure 5.17, we can then study the background signal. Since there is not star, it's already know that the flux level will be different, but we are only interested in looking at differences in amplitude and the overall shape of the light curve.

It's clear, in Figure 5.18, that the signal picked up by the trail mask is concordant with the oscillation inside our light curves, thus strengthening the belief that the target star is contaminating the background stars. Furthermore, we see that for Star 1 and 2, that the

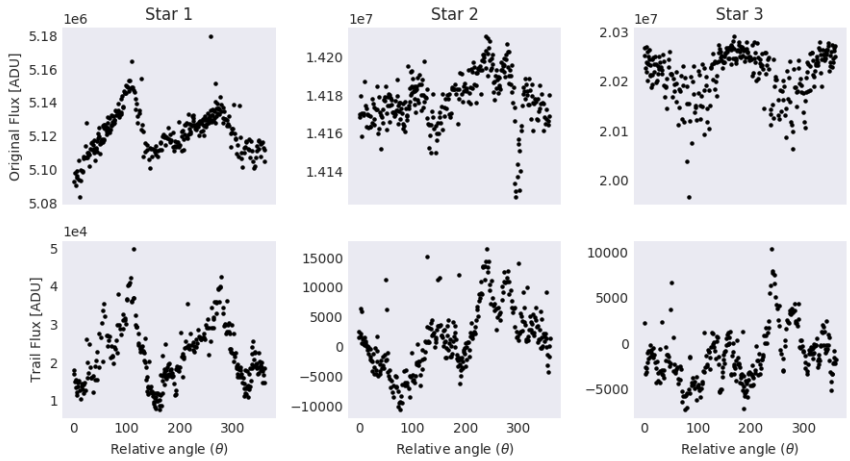
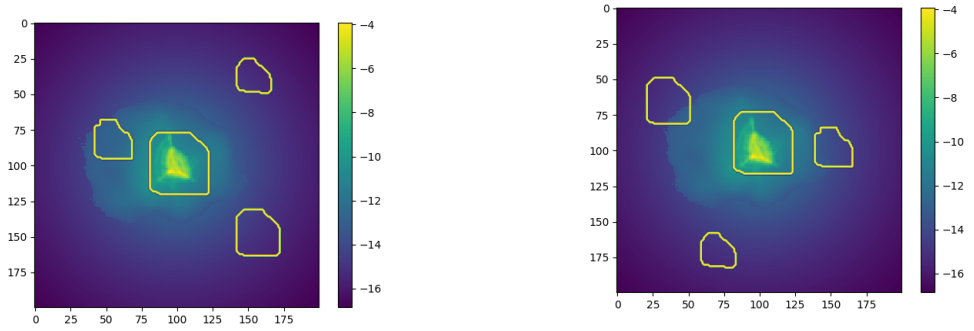


FIGURE 5.18: Flux calculated over each star mask phased so that it only picks up background.



(A) PSF of the target star with the entire mask of Star 1 inside it.

(B) PSF of the target star, with parts of Star 2 mask inside it, and the beginning of Star 1's second passage through the PSF.

FIGURE 5.19: Representation of the logarithm with base 10 of the central star's PSF, with the masks passing through it, in two different points in time.

oscillation's amplitudes diminish with the distance to the target star, thus also pointing to the existence of, at least, some contamination.

Although this hypothesis starts to seem more likely, we now have another unanswered question: If the central star is inducing alterations on the background stars, why do we only find them in a small part of the light curve?

Fortunately, since we are working with synthetic data, we have access to the true PSF used in the simulator, and can thus compare the mask's position against the PSF's shape.

As the PSF's shape is not uniformly distributed around the star's center, we have a region that extends further out. Star 1 and 2 managed to transverse this region, thus picking up flux from the central star. The closest star, due to the distance to the center,

also enters the shorter PSF's edge, thus explaining the period equal to half of the satellite's rotation period: in each lap around the target star, it enters twice the PSF of the target star.

Regarding the furthest star, Star 3, we see that it does not enter the PSF of the target star. Furthermore, against what was seen with the two other stars, the signal's amplitude does not decrease with the distance to the center, but instead increases, thus invalidating the contamination as the root cause.

So, for this star, we hypothesize that a combination of imperfections in the background and PSF corrections, alongside the mask limitation due to the closeness to the image's edge and jitter are the most likely explanations for what we see here.

Although we have proposed an explanation for this signal, it's a subjective analysis. It's entirely within reason that the background and the Flat Field are also responsible for some part of the signal found in the light curves, however they should be minimal and almost inconsequential.

5.1.3 Data set B

Now passing to our second data set, we can try to validate the previously taken conclusions for the best combination of methods, to minimize the noise. In Figure 5.20 we can see the extracted LCs, with a *shape* mask and a *dynam-dynam* combination. We find that, for the target star, the transit is shallow, but can still be seen. Regarding Star 2, which should have a transit in it, we are not able to see it in the light curve, which was expected, due to the planet's small size coupled with it being in a background star and thus the observational conditions are not the ideal ones.

The best mask for each star, as we can see in Figure 5.21, stays almost the same, with the outlier being Star 2. However, recalling from Data Set A, both masks presented quite similar noise values so, small changes in the conditions of the Data Set can dictate which is best.

When comparing the light curves, Figure 5.22b, we find that despite the DRP's LC presenting a lower flux level, it's the one that presents the highest relative uncertainties. Similarly to Data Set A, we find that the LC obtained without using the background grid has a lower uncertainty level, than the one obtained with a background grid of 1800.

Comparing ARCHI's light curve against the OPTIMAL one from the DRP, in Figure 5.22a, we notice that the noise for ARCHI's curve is half of the one in the optimal curve. This difference in the noise level was unexpected, since for Data Set A, we have only

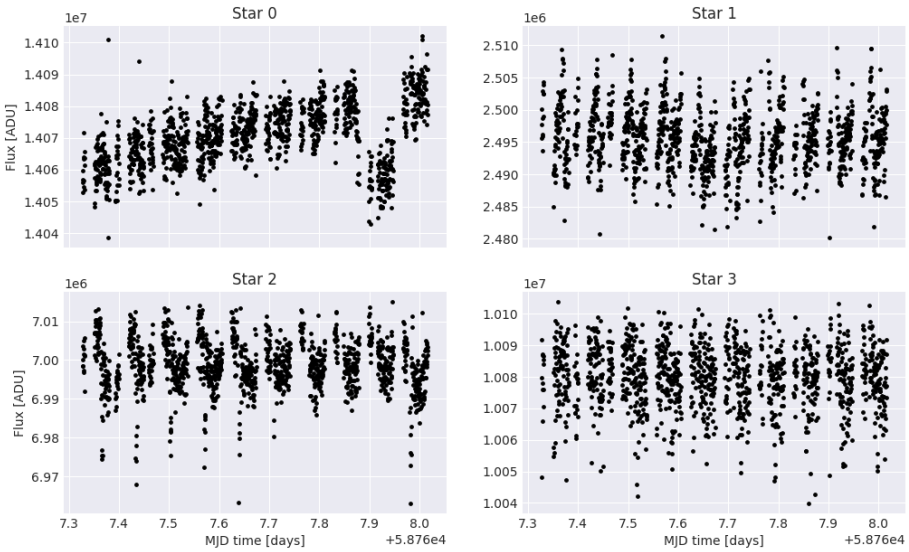


FIGURE 5.20: Light curves from the Data Set B, using a *shape* mask, a *dynam-dynam* combination and without a background grid.

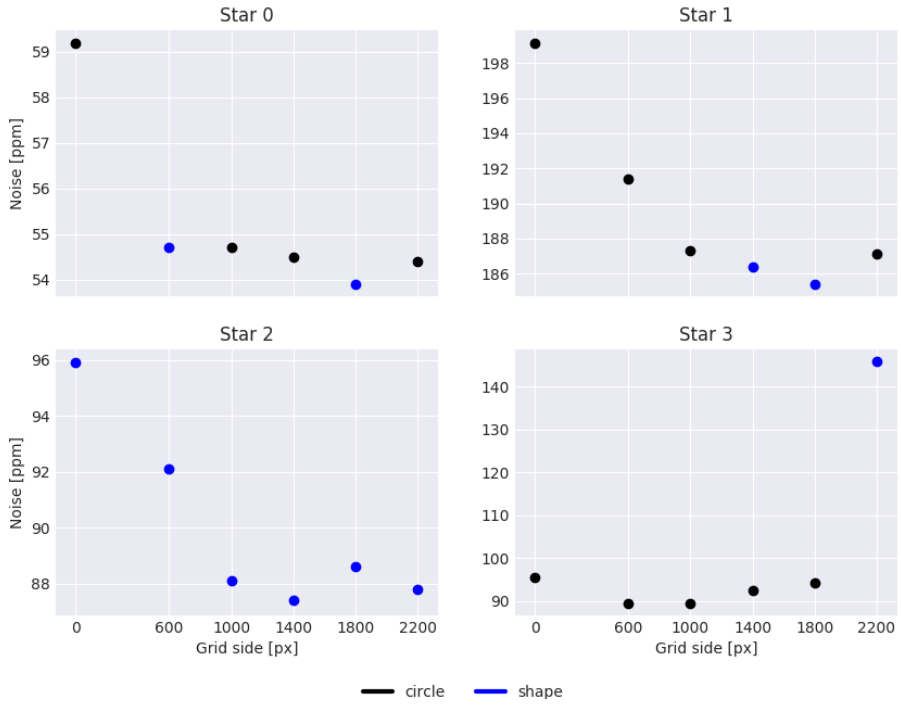
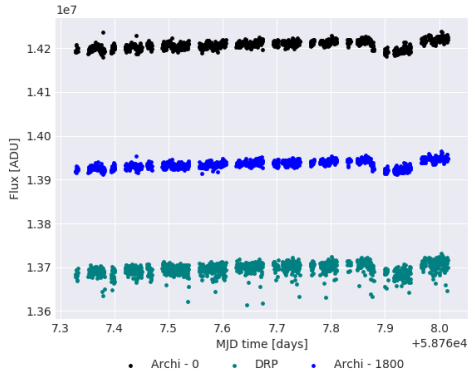
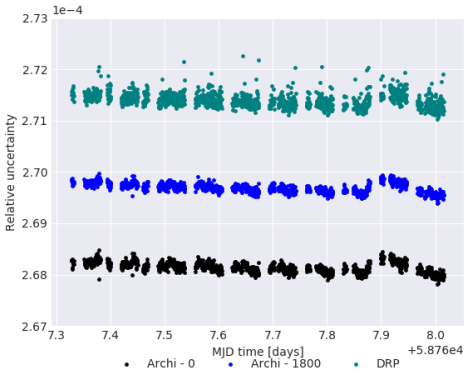


FIGURE 5.21: Evolution of the achievable minimum noise with the increase of the background grid, for Data Set B.

found a difference of approximately 5 ppm between the two LCs. Thus, to validate these results, we decided to apply the DRP's version of the CDPP that, as discussed before, calculates the noise in conjunction with the transit.



(A) Comparison between ARCHI's best light curve for the target star, against the DRP's OPTIMAL light curve. In green we have the *OPTIMAL* LC from the DRP, whilst in black and blue we have ARCHI's LCs without using a background grid and using one with a size of 1800, respectively.



(B) Comparison between ARCHI's light curve uncertainties with a grid of 0 and 1800 against the uncertainties from the DRP's *OPTIMAL* light curve. Once again, the uncertainties are divided by the flux level of the corresponding LC.

FIGURE 5.22: Comparison between ARCHI and the DRP for Data Set B. In here we find that ARCHI's curves have approximately half of the noise present on the ones from the DRP.

TABLE 5.5: Noise calculation using the DRP's CDPP algorithm, for both the DRP's OPTIMAL light curve and the best ARCHI's light curve, for Data Set B. The CDPP was calculated for a time scale of 30 minutes and 2.5 hours, while using the DRP's version and, with *Kepler's* versions we used the typical time scale of 30 minutes.

Pipeline		DRP		*Kepler*
Time scale		30 min	2.5 hours	30 min
ARCHI	0	51.44	24.33	59.2
	1800	44.68	20.71	53.9
DRP		111.57	48.36	129.0

When looking at Table 5.5, we notice that the noise without removing the transit from the data set is lower than when we remove it, which should not occur. It's not possible to have a lower noise metric when considering the transit and the noise, than when considering only the noise, thus proving our suspicions that the ported *Kepler*'s CDPP does not fare well with temporal gaps in the data.

Looking at Figure 5.23 we notice that the SavGol filter, once again, fits perfectly the transit, assuming an uniformly spaced time. However, this assumption by the CDPP algorithm, does not take into account the gaps introduced by either the SAA passage or Earth occultations. Thus, the application of this filter over the LC, may be removing or adding noise, depending on the values before and after the time gaps. Another explanation for these results can be the existence of the linear trend in the light curve, that has a greater impact in one of the algorithms. Since this discussion is outside the scope of this

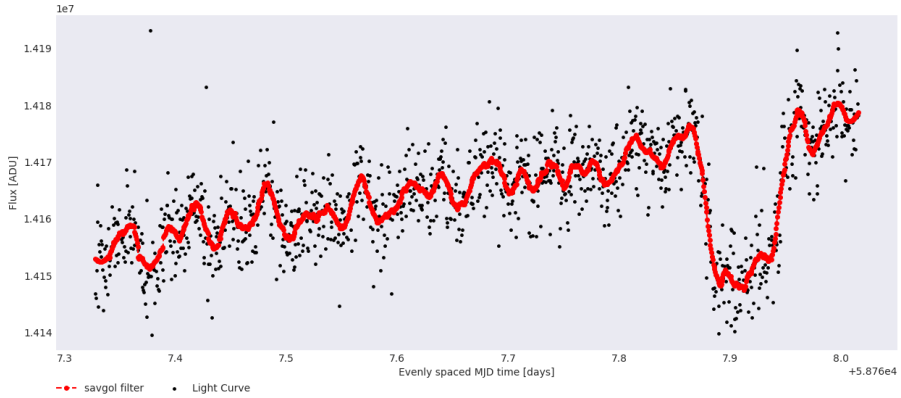


FIGURE 5.23: Comparison of the SavGol filter, assuming an evenly space MJD time, against a target light curve from Data Set B.

thesis, we shall use the DRP's CDPP to calculate the noise from now on. With this, we have to perform the optimization routine all over again, now with the new optimization metric.

With this new metric, in Figure 5.24 we find the same patterns as in Data Set A, with the noise decreasing with the size of the background grid, until a lower limit is met.

Strangely, we can also see the biggest grid for the outermost star, Star 3, also produces a far worse result than not using a background grid whatsoever. There is no clear motive for this to happen.

Although we can find differences in the overall noise level, we notice that the relative variation between the noise level for each grid still holds.

Comparing all of the possible combinations against themselves, Appendix D.2, we see that the noise levels follow the same pattern as found for Data Set A. The *offsets* and *dynam* methods present similar results for the target star while, for the background ones, the *dynam* star tracking method, once again, is the best option.

Regarding the best mask for each grid, we can notice some small differences when comparing against the previous data set. Foremost, it's important to note that only Star 3 prefers the *circle* mask while, for the other 3 stars, we notice an inverse pattern: where, previously, the *shape* mask was better, now the *circle* one is and vice-versa. Figure 5.25 shows us that the differences for the target Star and Star 2 are almost meaningless, although Star 2 presents an higher difference for the last grid. Star 1, in a global way, is the one that presents the biggest differences between the two masks.

If we pay a closer attention to Figure 5.21 we see that, for the background stars, the *shape* mask already is the best choice, for certain grids. This disparity between the data

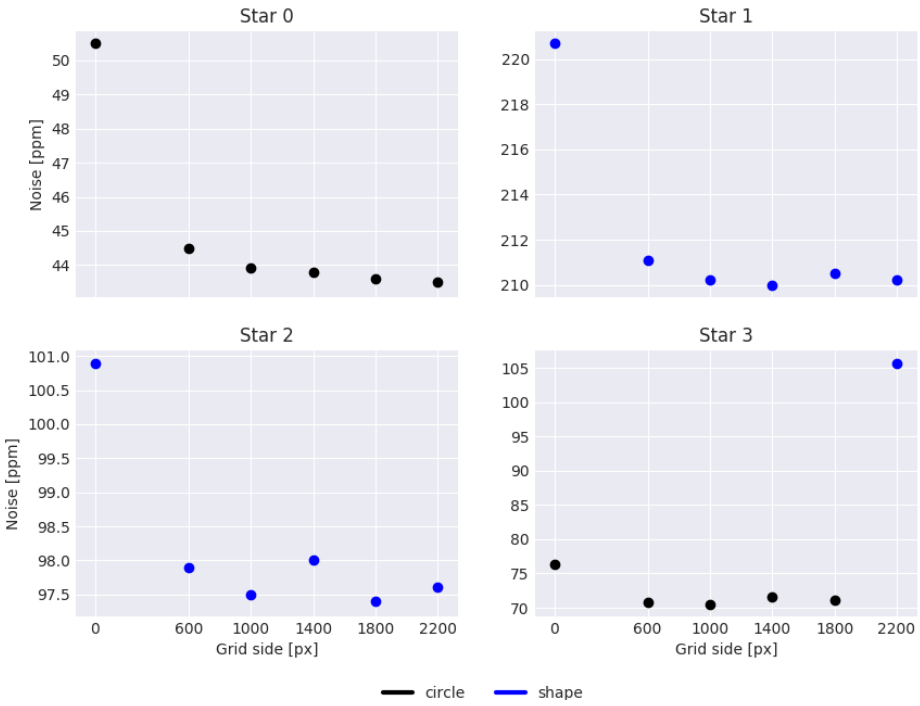


FIGURE 5.24: Evolution of the achievable minimum noise with the increase of the background grid, for Data Set B, with the DRP's CDPP algorithm.

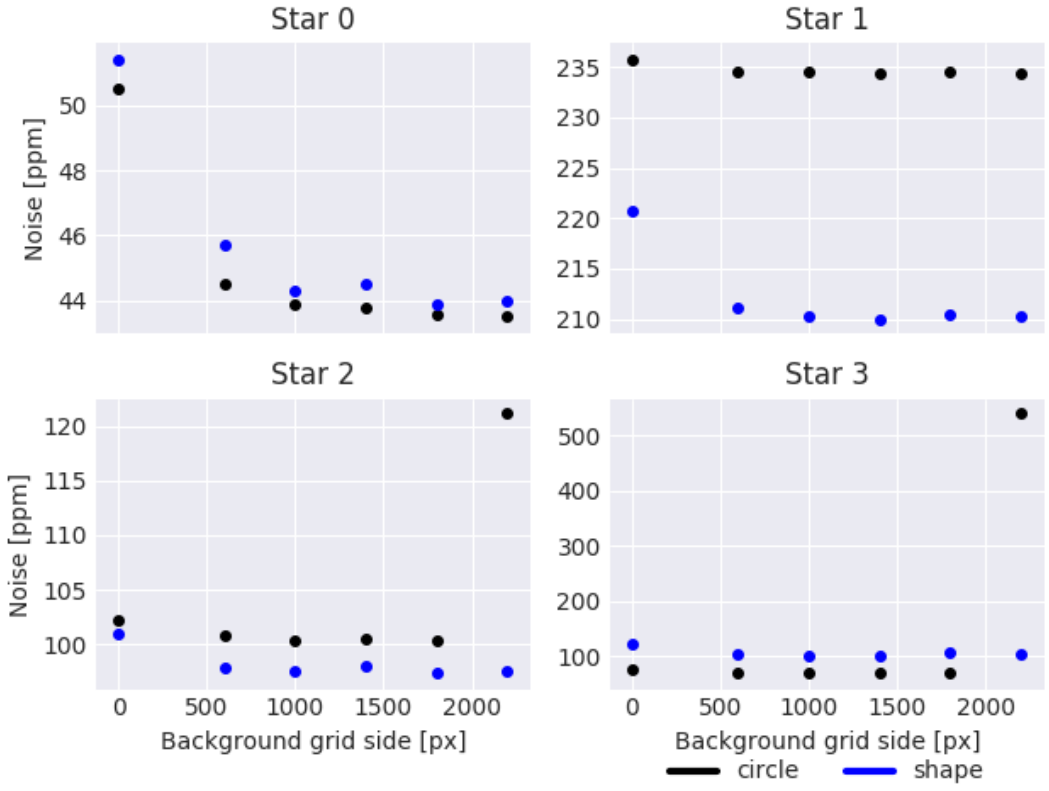
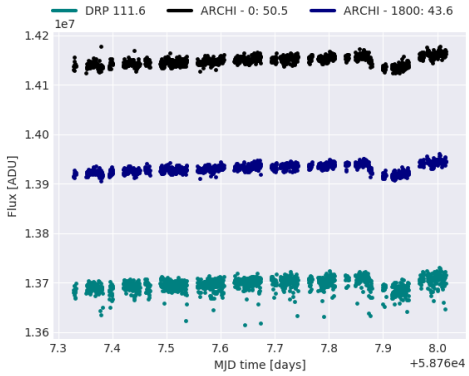


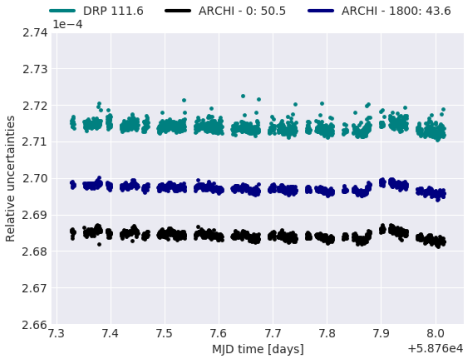
FIGURE 5.25: Comparison between the minimum noise achievable with the *circle* and *shape* mask, for Data Set B, with the DRP's CDPP algorithm.

sets can be attributed to the simulated noise, and even to the difference in sizes between them. Thus, it's recommended that when using ARCHI, both masks should be optimized, in order to find the best Light Curves.

Now that we optimized the light curves using a different method we find, in Figure 5.26, that we have almost no obvious differences between these and the ones obtained with the *Kepler's* CDPP algorithm, except a lower noise, due to the re-optimization process.



(A) Comparison between the DRP's OPTIMAL light curve, against ARCHI's best light curves without a grid and a grid of 1800, for Data Set B, using the DRP's CDPP algorithm.



(B) Comparison between ARCHI's light curve uncertainties with a grid of 0 and 1800 against the uncertainties from the DRP's *OPTIMAL* light curve.

FIGURE 5.26: Comparison between ARCHI and the DRP for Data Set B, while using DRP's CDPP algorithm.

It's noteworthy that ARCHI's light curves present approximately 55% of the DRP's noise without using a background grid and, when we introduce it, with a size of 1800, this noise reduction passes to approximately 61 %. On top of that, both of ARCHI's light curves present lower uncertainties, relative to the LC flux level, than the DRP.

5.1.3.1 Data Set C

The past two Data Sets have already allowed us to compare the methods against themselves and against the DRP. However, since we have not yet managed to find transits in the background stars, we shall use this one. However, take note that it will only be a very brief analysis to validate the light curves from the background stars, without a further analysis of the Data Set.

In Figure 5.27 we see that a bigger planet is capable of being detected in the background stars. The smallest one, is found in Star 1, with a radius equal to the one from

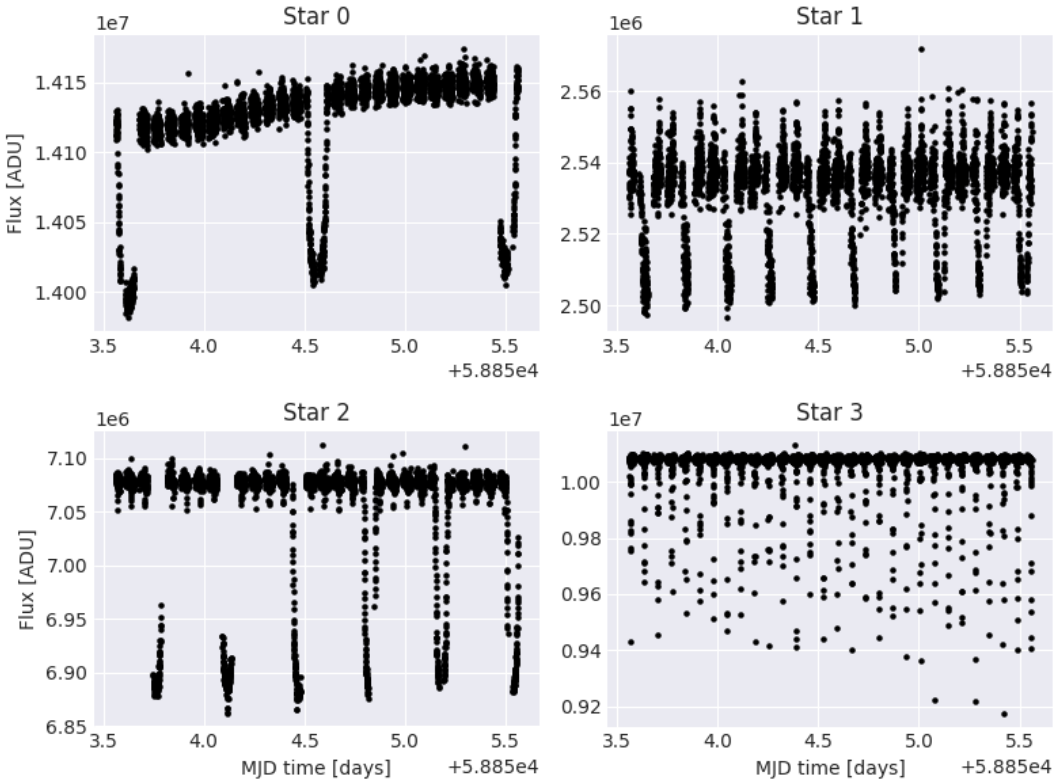


FIGURE 5.27: Light curves, from Data Set C, while using a *shape* mask, with a *dynam-dynam* method combination and without a background grid.

Jupiter. In Star 2, we see bigger dips in the flux than in the previous case, which is concordant with the fact that its radius is bigger. This Data Set allowed us to validate the capability of detecting transits in the background stars.

5.1.4 Overview of the results

The three studied data sets, albeit covering very different cases, still do not account for many of the possible cases that one might find when the mission sees its first light. As an example, we have not seen if a star on the outermost star is capable of being detected, despite the high noise level that accompanies that LC, due to the edge closeness. Further along in Chapter 5.3, we shall propose new steps, for the future, on how to better analyze the found results and how to improve ARCHI. However, despite the shortcomings of the presented analysis, we can still draw some conclusions.

Even though we did not have enough data to compare the *shape* mask against the *circle* mask, we have found that they are almost equivalent, in terms of noise in the LCs and in terms of relative uncertainties. For the central star, we found that our *dynam* method

and the one from the DRP's, used in the *offsets* method, also give almost equal results, depending on the background grid in use. On the background stars, the *dynam* star tracking method was the only that could track the stars, in a reliable way, independently of the background grid in use.

In terms of the relative uncertainties, the introduction of the background grid does not improve the methods and, in some cases, ends up worsening them. Similarly, we have failed to find a meaningful noise reduction in the background stars, when using the background grid, despite the good results for the target star.

Comparing our results against the DRP's *OPTIMAL* LC, with DRP's CDPP algorithm, we found small improvements in the first case, with an improvement of $\pm$ 1 ppm. For Data Set B, we have found an improvement of $\pm$ 60 ppm. The last Data Set, allowed us to conclude that it's possible to detect transits within the background curves.

Lastly, we have also proved that for stars near the target one, the PSF of the central star has a shape such that the background stars enter it, and we find contributions in their light curves.

5.2 The impact of the Gaussian processes

In this Section, we shall now explore the application of the GPS for the presented Light curves. Following the structure of the Photometry analysis, we shall start by comparing some of ARCHI's methods using Data Set A, with Data Set B we shall compare ARCHI against the DRP and, lastly, with Data Set C, we shall showcase the capability of the method adapting to a light curve with multiple transits in it.

In order to properly characterize the GPs, we shall compare the estimated light curve, for each combination, as well as the errors against the tabled values, taken from the CHEOPSim manual [17]. For Data Set A we have used 2000 runs for each of the stages, alongside 256 walkers. For Data Set B and C, due to their sizes, we used 128 walkers and 4000 runs for each stage.

5.2.1 Data Set A

Before we can start analyzing the outputs of the GPs, we have to see how the model created with the tabled values fare against the determined light curves and against one of the estimated LC models.

In Figure 5.28 we see that the transit model created with the truth values, does not follow neither ARCHI's curve nor the injected transit, thus making us doubt about the validity of what we shall consider as the "true" values. However, looking at the fitted model, the injected transit and the original LC we find that they are concordant with each other. In Appendix C we performed a linear interpolation to determine the star's radius and, the simulator may not use a linear interpolation, but instead choose the radius based on a different interpolation method. Thus, if we do not have the correct radius of the star, then the planet's radius, which is expressed in units of the stellar radius, will not be correct. Taking this into account, we shall refrain from calculating error values, and instead perform a more qualitative analysis of the results.

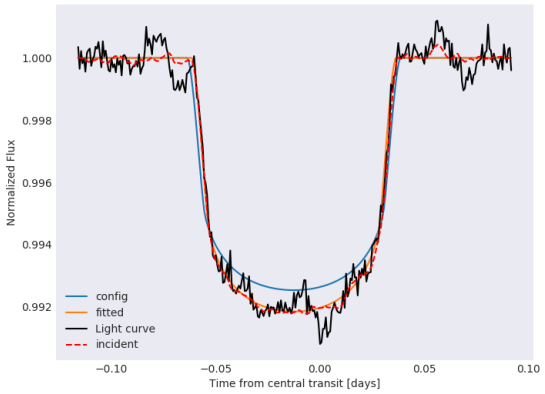


FIGURE 5.28: Comparison between one of ARCHI's LCs, with the injected transit and the model obtained with the tabled values. The injected transit was normalized using ARCHI's normalization process, described in Chapter 3.2.2. The red curve is the one injected by the simulator tool, the black one is one of ARCHI's light curves, in orange we have the fitted model for this LC and, lastly, the blue curve is the transit that we should be seeing, based on Table C.2.

Now choosing the best LC from the target star, i.e. the one with the lowest noise, for each background grid, we can see how the differences impact the fitted parameters. In Figure 5.29 we find similar values for the different grid sizes, except for the time of inferior conjunction, t_0. If we remember that this parameter is expressed in days, we notice that the from the lowest value to the highest one are $\pm$ 0.0159 days, which translate to roughly 23 minutes, which does not represent a great difference between them.

This difference stems from the fact that during the normalization process, we use the point with the lowest flux to center the time array to measure times from the mid-transit

point. Thus, for different combinations of methods, this lowest point is in different positions, and the t_0 parameter takes different values so that the mid-transit region occurs in the correct region.

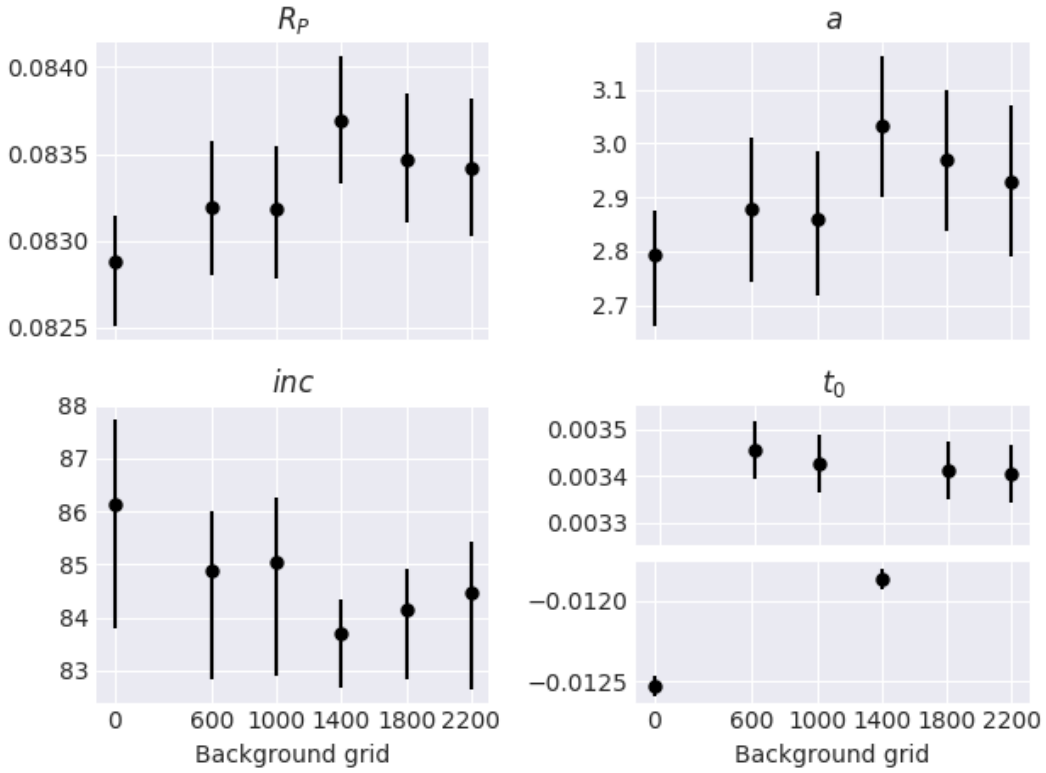
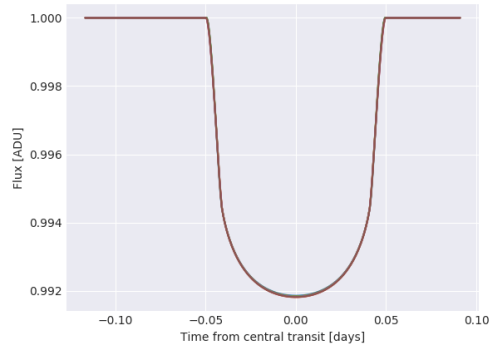


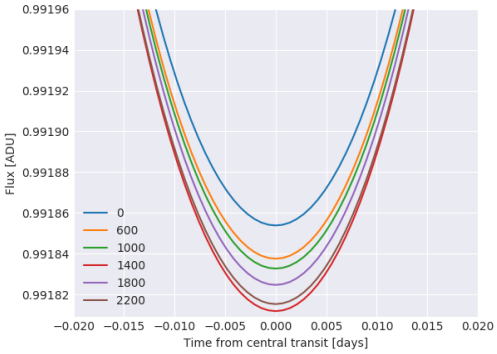
FIGURE 5.29: Fitted parameters, alongside the correspondent uncertainties, for the ARCHI's LC with the lowest noise for each of the previously studied background grids.

In order to compare fitted models, we shall set t_0 to zero, so that the transit's lowest point is always in the same place. In Figure 5.30 we cannot find any differences between the Light Curves. However, when zoomed in, we notice the slight differences that incur due to the different estimations for each parameter. With this, we can conclude that, in terms of the planet's parameters, all of the produced light curves are almost equivalent. If we had certainty of the true values, it would be possible to see if the best fit was concordant with the LC with the lowest noise.

We shall now turn our attentions to the analysis of a singular application of the GPs, for the light curve extracted with a *circle* mask, a *dynam-offsets* combination and without using a background grid. In the first place we have to make sure that convergence was achieved and, to do so, we start by looking at the values explored by the walkers during the production stage, in Figure 5.31.



(A) Comparison of all fitted LCs, for each background grid, for Data Set A.



(B) Zoom of the mid-transit region.

FIGURE 5.30: Comparison between all of the models created with the parameters fitted for each of studied background grids. For each background grid, we applied the GPs for the LC with the lowest noise level.

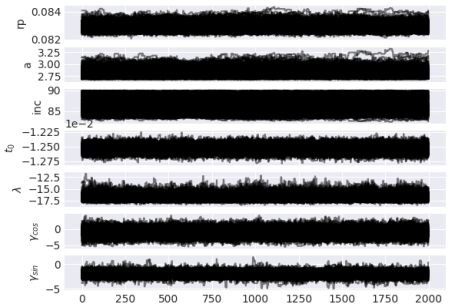


FIGURE 5.31: Values explored by the walkers in the production stage.

Looking at the chains allows us to have a preliminary idea of the state of convergence of the GP. In this case, we notice that all values are approximately stable during this stage, without any outliers. Furthermore, we notice that the value ranges are somewhat small, indicating the all the walkers were exploring a small space around a given point.

However, the corner plot in Figure 5.32, allows us to have a clearer idea, and search for correlations between parameters. Firstly starting with the planet's parameters we note that only the orbital inclinations deviate from the expected Gaussian shape. Remembering from Chapter 3.3.2, the orbital inclination upper limit was set to 90 degrees. This closeness to the upper limit, does not allow the walkers to freely explore around in this region, creating the sudden cut of the distribution. As expected, the "true" values do not match the fitted values but, once again, we shall disregard that, due to the difference found between the injected transit and the model created with those values.

It's also noteworthy that the kernel parameters, γ_{sin} and γ_{cos} converge to the same

value, which is to be expected in our domain, due to the equal periodicity of the sine and cosine.

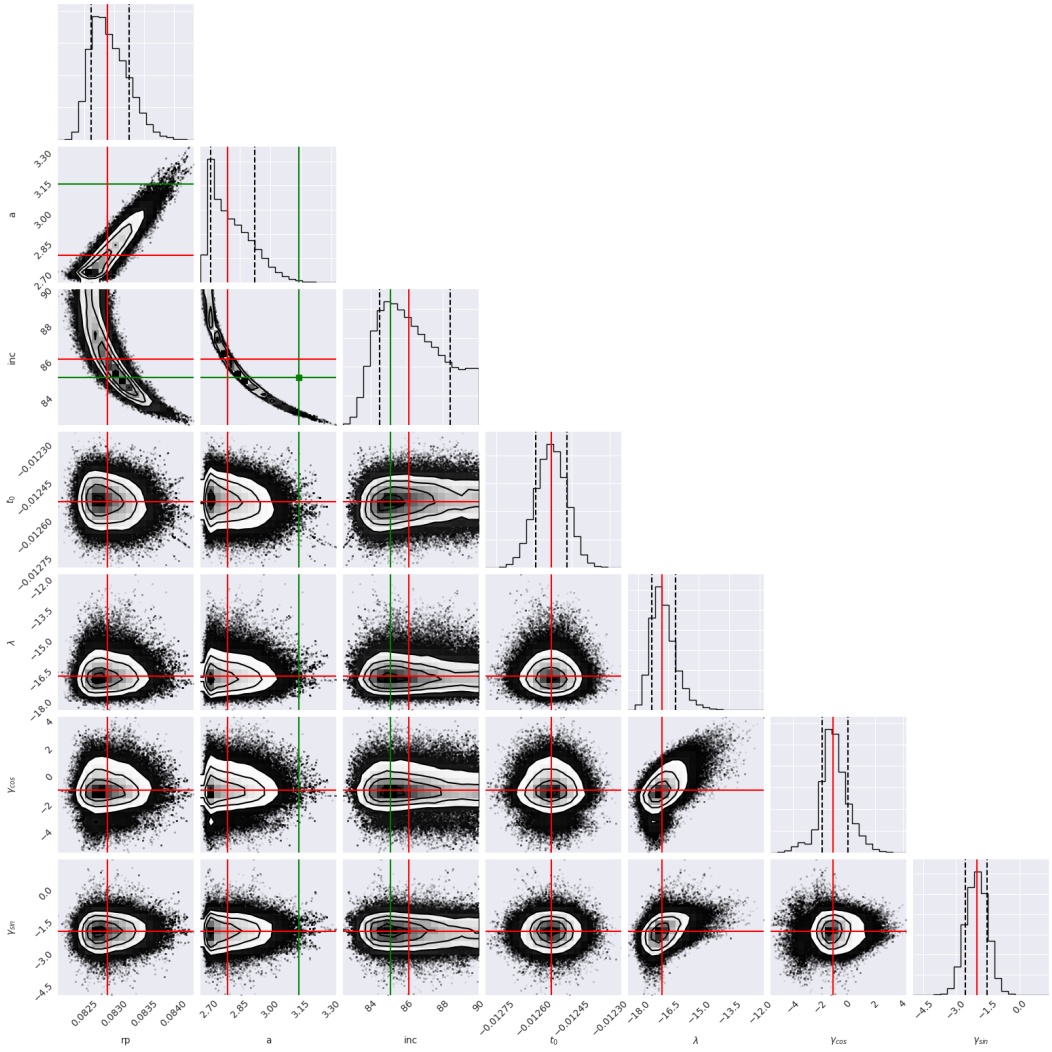
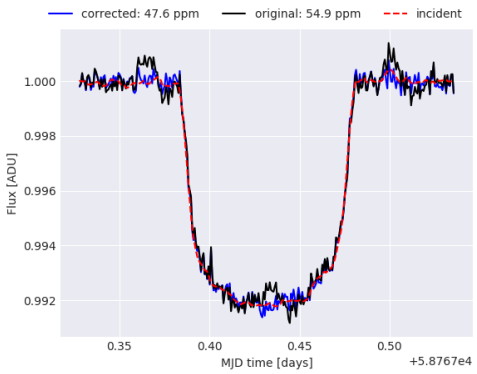


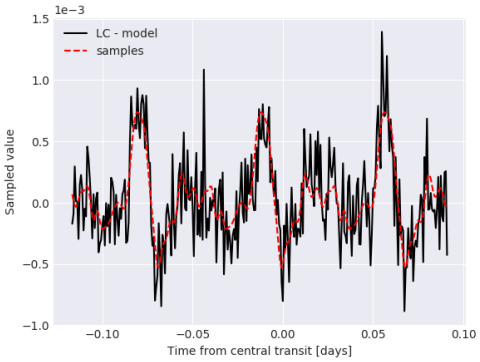
FIGURE 5.32: Corner plot of all of the GP's parameters. In red we can find the median of the distribution, in green the "true" values found in Appendix C, and the dashed lines represent the 6th and 84th quartile. This analysis was done for the light curve extracted using a *circle* mask, a *dynam-offsets* combination and without a background grid.

The last step with the GPs is to draw samples, and attempt to remove noise from the light curves. In order to accomplish this goal we sampled the kernel 100 times and calculated the median, for each position. Applying this step and removing the estimated noise from the original Light Curve, we arrive at Figure 5.33a, in which we see that the correction has managed to reduce the noise.

Although we have managed to decrease the noise, we can still see some of the small peaks found in the out-of-transit regions. Looking at the overall shape of the corrected



(A) Corrected light curve, from Data Set A.



(B) Median of 100 samples taken from the GP, compared against the normalized Light Curve without the fitted model.

FIGURE 5.33: Application of GPs to a LC obtained with a *circle* mask and a *dynam-offsets* method combination, for Data Set A.

TABLE 5.6: Impact of the Gaussian Processes, with the increase of the background grid. The GPs were used with a *circle* mask with a *dynam-offsets* combination.

Grid size	LC noise	Corrected LC noise
0	54.9	47.6
600	50.4	42.9
1000	49.6	42.3
1400	50.7	43.5
1800	50	43
2200	50.6	43.6

LC, we find that it now follows much closely the injected transit. If we look at the noise, we find that the overall shape of the samples are concordant with the noise fluctuations, without passing by all the points. Furthermore, we see that the correction applied to the large and periodic variations managed to remove them.

As previously mentioned, we do not have the DRP uncertainties for this Data Set and, since it's not possible to apply the GPs without them, we shall leave that part, once again, for Data Set B.

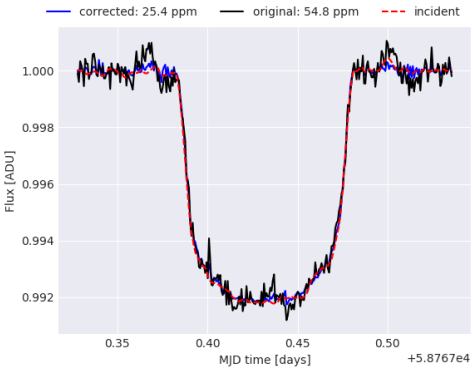
In Table 5.6 we see how the GPs impact the curve from each background grid, while using always the same configuration: a *circle* mask with a *dynam-offsets* method combination. The GPs, when applied over the background grids all yield very similar results, around 43 ppm. When using them, we have a decrease of around 6 ppm in the noise, which is also found for the curve without the background grid.

Lastly, if we look at other configurations from this Data Set, we find an unusual behavior. Depending on the chosen light curve, we find an over-fit of the kernel parameters,

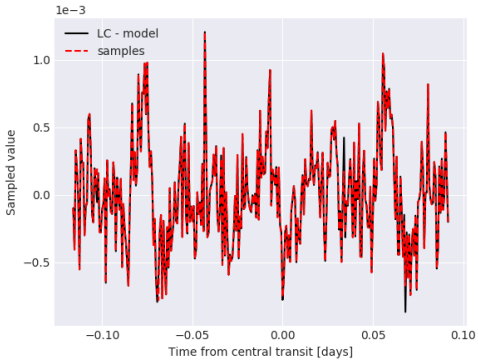
TABLE 5.7: Comparison between the results obtained with a *circle* mask and a *dynam-offsets* combination, against the ones obtained with a *shape* mask and a *dynam-dynam* combination-

LC	Original	Corrected
circle	54.9	47..6
shape	54.8	25.4

passing by all points of the noise values used as the inputs of the GP. This over-fit leads to GP following each point, instead of attempting to model the behavior.



(A) Corrected light curve, from Data Set A.



(B) Median of 100 samples taken from the GP, compared against the normalized Light Curve without the fitted model.

FIGURE 5.34: Application of GPs to a LC obtained with a *shape* mask and a *dynam-dynam* method combination, for Data Set A, without using a background grid. In this case, we have an over-fit of the kernel, estimating values almost equal to the given inputs.

In Figure 5.34b we see that the noise estimates by the GP are almost equal to the input values, thus revealing that instead of modeling the behavior, the kernel is such that closely follows the inputs. This is further evidenced by looking at the correspondent corner plot, in Figure 5.35.

Looking at Table 5.7 we see that the correction, for the LC obtained with the *shape* mask has the noise reduced to half of its original value. This can also be seen in Figure 5.34a, in which the corrected light curve presents almost no variations and closely follows the fitted model and Figure 5.34, where the samples match, almost completely, the given inputs.

The corner plot, Figure 5.35 and the estimated values in Table 5.8 reveal some information for which we cannot create a solid hypothesis to explain. In the first place, we see that the planet's parameters are close to the previous models. However, when we look at the kernels, we now see that the one applied over the sine has a length scale that is almost

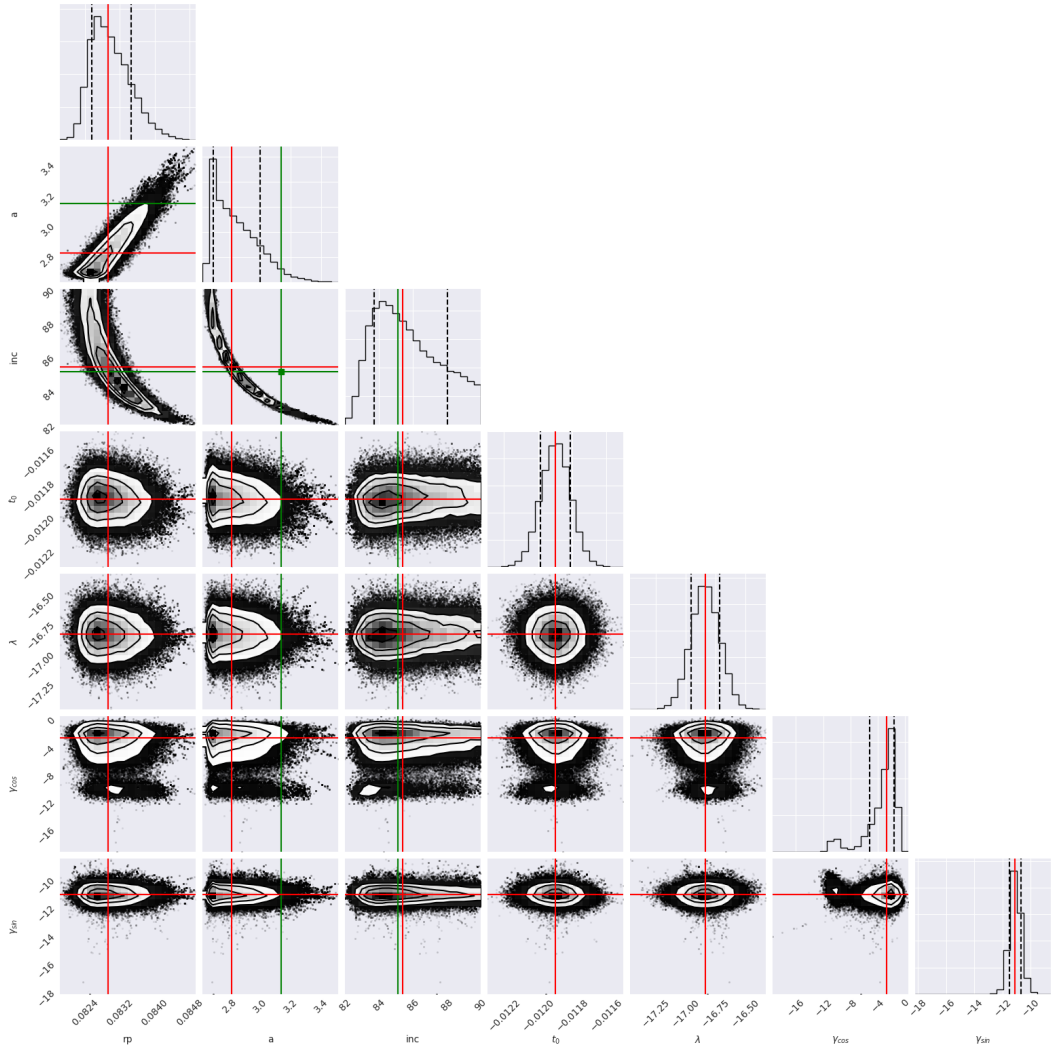


FIGURE 5.35: Corner plot of all of the GP's parameters. In red we can find the median of the distribution, in green the "true" values found in Appendix C, and the dashed lines represent the 6th and 84th quartile. This time, we used a light curve extracted with a *shape* mask, a *dynam-dynam* method combination and without using the background grid.

double of the one found with a *circle* mask. Furthermore, the cosine kernel also shows a secondary peak, behind the 16th percentile region, near the same values. Interestingly, using light curves produced by a *shape* mask with a *dynam-dynam* combination, but with the background grid enabled, do not show the same pattern and in them we, once again, find almost equal length scales on both kernels.

If we compare the planet parameters for both of these runs we find that the not only are the planet parameters quite similar, but also the uncertainties found with each method are very close to one another. With this, we can conclude that the quality of the planet's parameters is not tied with the kernel parameters.

TABLE 5.8: Comparison of the GP parameters for both cases under study. The "circle" entry is referent to the LC obtained with the *shape* mask and a *dynam-offsets* combinations. The "shape" refers to the LC obtained with the *shape* mask and a *dynam-dynam* method combination.

Parameter	Mask	
	Circle	Shape
rp	$0.0832^{+0.0004}_{-0.0004}$	$0.0829^{+0.0004}_{-0.0005}$
a	$2.8585^{+0.1352}_{-0.1433}$	$2.8166^{+0.1200}_{-0.1843}$
inc	$85.0165^{+1.2230}_{-2.3556}$	$85.4175^{+1.7005}_{-2.6235}$
λ	$-17.449^{+0.4144}_{-0.5594}$	$-16.8385^{+0.1186}_{-0.1220}$
γ_{cos}	$-0.9708^{+1.3473}_{-1.0333}$	$-2.6759^{+2.4713}_{-1.0354}$
γ_{sin}	$-2.7538^{+1.8492}_{-0.6127}$	$-11.1152^{+0.3787}_{-0.4103}$

However, we do not see the same when looking at the kernel hyperparameters. The amplitudes of both kernels are also quite similar, contrasting the differences found in the length scales. Not only are the differences significant between the two cases, but also the uncertanties in the kernel's hyperparameters are quite high in general, except for the sine kernel, when using a *shape* mask.

5.2.2 Data Set B

Now passing onto our second data set, in which we shall compare an ARCHI's curve, obtained with a background grid of 600, a *circle* mask and a *dynam-dynam* combination, against the DRP's *OPTIMAL* light curve. Remembering from Chapter 5.1.3, DRP's curve had much more noise than the ones extracted using ARCHI.

In Figure 5.36 we find a comparison between the two normalized light curves and the correspondent models that were fitted to them. At first, is evident that the DRP's curve has more outliers than the one from ARCHI. This bigger distribution of points in the light curve, lead to a model with a smaller planetary radius, evidenced by the smaller dip in flux, during the transiting event. Contrastingly, ARCHI's light curve result in a model that appears to be good, although it's not possible to quantify how good.

If we perform a closer analysis of the fitted values, with Table 5.9, we notice that the differences between them are somewhat significant, even when taking into account the uncertainties. When comparing the uncertainties in the planetary parameters, we notice that they are slightly higher than in Data Set A, but that is due to the lower signal to noise ratio in this Data Set.

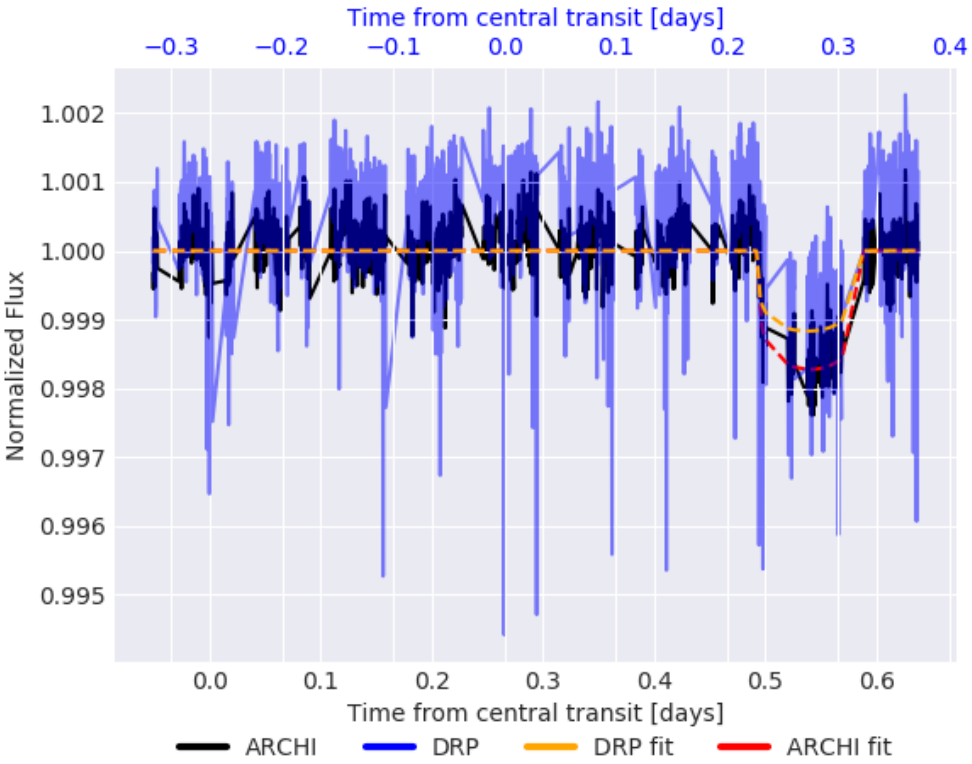


FIGURE 5.36: Comparison between two different light curves. In Blue, we have the DRP's *OPTIMAL* LC and, in Black, ARCHI's LC, obtained with a background grid of 600, a *circle* mask and a *dynam-dynam* combination. Furthermore, we can also see the fitted models: In orange the one from DRP's LC and, in red, the one from ARCHI.

TABLE 5.9: Comparison of the GP parameters for both cases under study, for Data Set B. Two different light curves were used: ARCHI's light curve while using a background grid of 600 and a *circle* mask with a *dynam-dynam* combinations and the *OPTIMAL* light curve from the DRP.

Parameter	Pipeline	
	ARCHI	DRP
rp	$0.0391^{+0.0010}_{-0.0011}$	$0.0318^{+0.0017}_{-0.0016}$
a	$3.8350^{+0.9382}_{-1.1493}$	$3.0849^{+1.1339}_{-0.4752}$
inc	$82.3237^{+1.0214}_{-1.9145}$	$82.4538^{+4.0672}_{-2.1141}$
λ	$-18.2861^{+0.5233}_{-0.7096}$	$-14.5948^{+0.0437}_{-0.0444}$
γ_{cos}	$0.7165^{+2.3097}_{-1.4259}$	$-14.2322^{+0.4201}_{-0.4299}$
γ_{sin}	$-2.3761^{+0.5660}_{-0.6741}$	$-12.8851^{+0.1584}_{-0.1643}$

The main difference that we find when comparing both light curves is the kernel's hyperparameters. In ARCHI's case we find similar values to those found for Data Set A but, in the DRP's case we, once again, find small values for the length scales. Although, in this case, the length scales of both kernels are approximately equal, instead of having

TABLE 5.10: Comparison between the CDPP from the DRP's LC and ARCHI's LC. This comparison is made before and after applying the GPs and calculated with the DRP's CDPP algorithm, for a time scale of 30 minutes.

LC	Original	Corrected
ARCHI	44.4	43.3
DRP	111.7	9.7

one kernel with a length scale much smaller than the other one. Thus, we expect to find, one more time, an over-correction of the light curve.

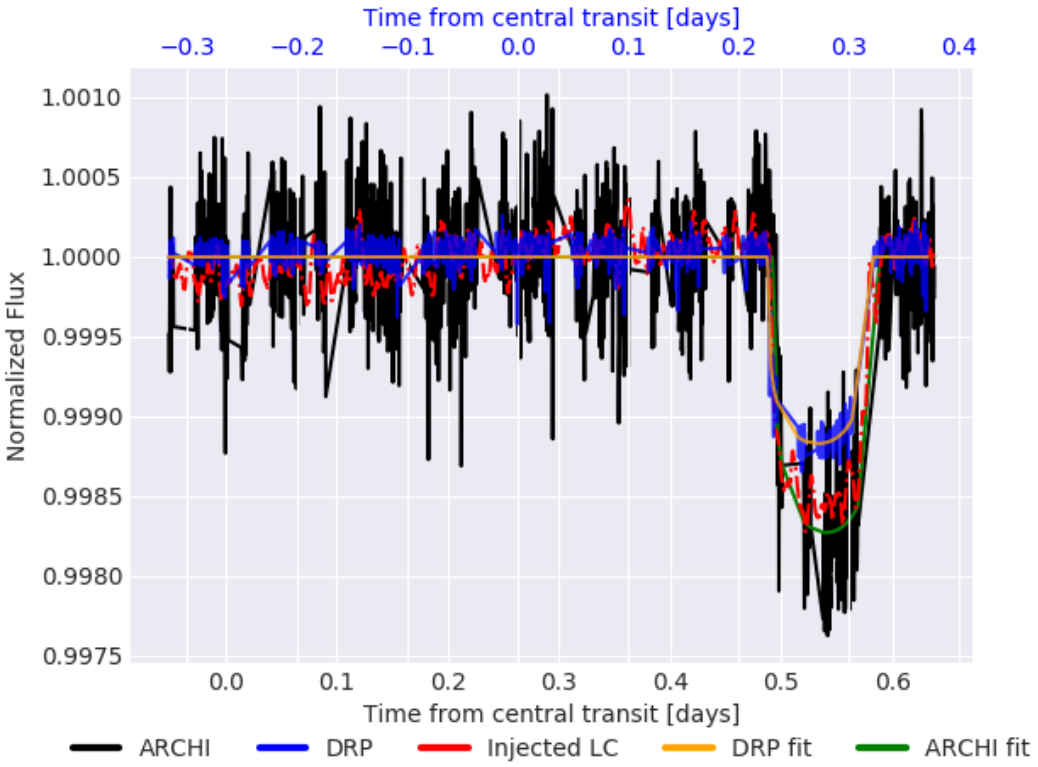


FIGURE 5.37: Correction of the LCs, using the samples drawn from the GP. In red we have the injected transit, in green the fit with ARCHI's LC and in orange the fit with DRP's LC. We find that the time for the central trasit, both cases are not near the transit. In the DRP's case, due to the high noise seen in the LC, the point with the minimum flux is near the middle of the LC. In ARCHI's case, there was a bug in which the t_0 calculation was made before normalizing the light curve.

In Figure 5.37 we see that the DRP's LC does not present as many outliers as before the correction, being much flatter outside the transit. However,we find that the fit made with ARCHI is much closer to the injected transit than the one from the DRP. Comparing the noise in each light curve, in Table 5.10 we see that the correction applied over ARCHI's LC barely had any effect, only reducing 1 ppm. Contrastingly, in the DRP's case, the noise reduces more than 10 times, from 111 ppm to 10 ppm.

As discussed, we do believe that this huge reduction in the noise is caused by an over fit of the kernel hyperparameters, and that the presented light curve is over-corrected.

5.2.3 Data Set C

This Data Set has light curves with the transits near the borders of the image, thus invalidating the normal usage of the normalization process since, in this case, it would choose points from within the transit. To solve this, we had to change, manually, the points that were going to be used to estimate the normalization line. Adding to this, Data Set C is the one with the most images and, due to time constraints, it was not possible to analyze all three light curves. Instead, we decided to only study Star 2 of this Data Set.

The analysis was done without a background grid, and using a *shape* mask, with a *dynam-dynam* combination.

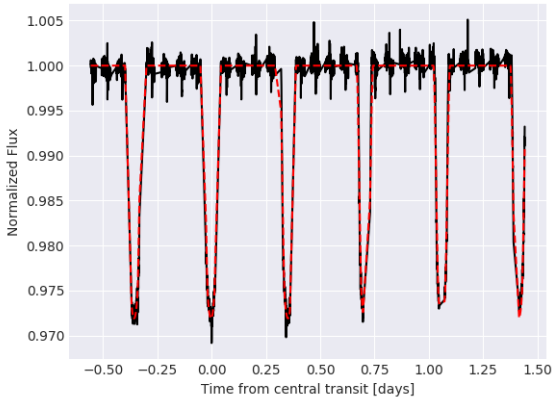


FIGURE 5.38: Model created with the fitted models, for a LC with multiple transits. In black we have ARCHI's LC, extracted during a *shape* mask, a *dynam-dynam* combination and without using a background grid. This LC is from Data Set C, for Star 2.

Looking at Figure 5.38 we see that our model is well adjusted to all transits that occur in the light curve, and that the model can work, even if multiple transits are present in the light curves, as long as the normalization process works as expected and does not use a mid-transit region for the creation of the line that normalizes the LC.

5.2.4 Overall remarks

In this section we looked at the GPs applied over ARCHI's light curves and the *OPTIMAL* light curve, from the DRP. In the first place it's important to note that the transit models, created with the estimated parameters, overlap the light curves, thus proving that the

parameters are close to their real values. However, further characterization of the results was not possible, due to a mismatch between the truth values and the injected transits.

Despite these good results, the removal of noise from the light curves is trickier to characterize. We have found that the kernel hyperparameters can suffer drastic changes when comparing light curves from the same data set. In most cases we find that the length scale of both kernels are close to one another. However, in some cases, we find that the length scales go to very small values, which leads to the sampled values following, almost perfectly, the input values and thus over-correct the light curves.

Even though we did not have the time to fully understand what is causing those abnormalities in the kernel behaviour, we postulate that this occurs due to our model consisting of only the transit model and the GP, that will attempt to model everything else that is not the transit. A more complete model, with a kernel for white noise and another for other variations that occurs over the observation, in a non-periodic fashion, would allow us to only sample the kernel that is associated with the rotational movement and only correct that kind of noise. Lastly, we have found that the parameter's determination is not dependent on the kernel parameters, i.e. even when we find over-corrections, the estimated parameters give models that are concordant with the light curves.

5.3 ARCHI's limitations and the next steps

Starting with the photometric module, the most obvious problem passes by the fact that CHEOPS, in a typical observation, is configured for the target star without taking into account the background ones. This means that we can find cases in which the exposure time is such that the star's magnitude leads to saturation of the CCD, which will lead to us not being able to extract meaningful information. Under those conditions we do not know how the shape based algorithms behave nor the effects on the surrounding stars and thus, more tests should be made to characterize ARCHI when working under those conditions.

Furthermore, as we have seen in Chapter 2.1.2, the shape detection method can fail for faint objects in the image, thus revealing that for short exposure times, we may not be able to reliably track some stars.

Following the good results presented by the DRP and the apparent plateau found with the increase of the background grids, we could improve the performance by implementing some of the DRP's methodologies. The first proposal is to improve the mask shifting

with DRP's anti aliased shifting algorithm. Due to the good results shown by the DRP's center detection, for the target star, we could also make a study of the feasibility of porting it to work with the background stars, or to be used as an improvement of the *dynam* star tracking method. As we have seen, contamination from nearby stars is a problem that we face and, similarly to the DRP, we could also try to estimate and correct their contributions.

Regarding the noise metric ported from the *Kepler* mission by my colleague Pedro, discussed in Chapter 2.6, it also needs to be improved to account for the temporal gaps that appear due to the SAA. Even though the SavGol filter showed that it can remove the transit from the light curves reasonably well, we could still improve this step, maybe through the application of a box-fitting algorithm (BLS) [82], if viable, or through the usage of Gaussian Processes. Although the later would raise the computational costs of the pipeline.

Another direction that could be also explored is the improvement of the DRP's data reduction algorithms. As an example, the background estimation algorithm could be improved by removing the background stars from the histogram used to estimate the background.

The Gaussian processes, albeit being functional, still have room for improvements. In the first place, the normalization routine is a basic one and it easily fails, as seen with Data Set C. Once again, a transit detection routine would allow us to avoid those regions. Not only that, but our model is still incomplete and is not able to fully characterize the noise and, sometimes, it leads to an over-fit. In this work we have assumed that the noise was periodic, and treated it in such way. It's true that a big portion of the noise is correlated with the rotation angle, although we also have star variability, among other effects that are correlated with the time and not expected to be periodic.

Furthermore, the kernels in use, albeit being able to fit the noise in the Light curves, are not focused for circular domains. We expect that a better kernel could also improve the GPs performance. In Chapter 3.1.3.1 we have also seen ways to validate the convergence of the chains which, unfortunately, there was not enough time to properly implement.

Lastly, the analysis of the GPs is still incomplete since, as we have seen, the injected parameters did not match with the injected light curve, limiting our ability to properly characterize the fitted parameters.

To finalize, it's important to keep in mind that the real data from the CHEOPS mission is yet to come and that the simulations, albeit useful, are not completely accurate. Thus, with CHEOPS's first light, ARCHI will, more than likely, need to be tweaked to extract the most of the available data.

Chapter 6

Conclusion

The main goals of the project have been completed, we have managed to extract light curves from the background stars, as well as reduce the noise in them with Gaussian Processes. Alongside those goals, we also fitted planetary parameters with good results.

In this work we studied three different data sets, each with a different purpose. The first one, Data Set A, allowed us to see that the target star produced the best results with either the DRP's star tracking algorithm or the one implemented on ARCHI, depending on the size of the background grid. For the background stars, the *dynam* method, i.e. the one based on the analysis of the star's shape, was clearly the superior choice, revealing lower noise levels than the other ones. Furthermore, we found that applying background grids with more than 1800 points in each side does not translate into lower noise but, instead, it stabilized. The same is found for the uncertainties. When comparing against the DRP, we found that ARCHI's best curve translates into a 5 ppm reduction in the noise. Furthermore, we found that the target star's PSF irregular shape lead to a contamination in the signal of the closest light curve with period equal to two times the one from the satellite. The contamination was also found for the second closest star, although it only entered once the PSF region. On the star that is further away, we could also see a similar signal, although no reasonable explanation could be found.

In the second Data Set, Data Set B, we obtained a reduction of 60 ppm, when comparing against the DRP's *OPTIMAL* light curve. Furthermore, we found that ARCHI's light curves, without using a background grid and with a background grid of 1800, managed lower relative uncertainties than the DRP's light curve. Lastly, with Data Set C we proved that ARCHI can detect transits on the background stars. However, since we did not simulated a Data Set with a planet orbiting a star close to the image's edge, we have yet to

validate transit extraction for stars under those conditions.

The application of the Gaussian Processes has revealed itself to be successful, with the application of a bi-dimensional Exponential Squared kernel, over the sine and cosine of the satellite's rotation angle. Using it, we found that the models are close to the injected light curves, albeit it was not possible to quantify the noise in each of the fitted parameters, due to a mismatch between the configuration values and the injected transit.

Regarding the removal of noise by drawing samples from the GPs, we have managed to do it, albeit finding a behavior in the kernel for which we cannot produce a complete explanation, but suspect that it may be related to a short coming in the creation of our model for the noise. Depending on the methods, used to obtain a light curve, from within the same Data Set we can have an over-fit of the kernel and thus, the samples follow, very closely, the input values. Using the light curves that did not result in an over fit of the kernel, we found an improvement of 7 ppm for the first Data Set and of 1 ppm for the second one. The last Data set, allowed us to validate the model used to fit the planet's parameters, when multiple transits occurred in the same light curve.

Appendix A

Great-circle distance derivation

From [83] we can express the coordinates as seen in Equation A.1. We can now calculate the distance between two points, with right ascensions(α_i) and declinations (δ_i). It's important that we do not forget that we only use a small part of the CHEOPS's original image and thus we have, in practice, values for the RA and DEC not very far apart from each other. Using Pythagoras theorem, it's easy to calculate the distance between two points if we know the size of the displacements in each direction.

$$\begin{aligned} X &= cos(\delta)cos(\alpha) \\ Y &= cos(\delta)sin(\alpha) \\ Z &= sin(\delta) \end{aligned} \tag{A.1}$$

In this case, we are only interessed in the changes in RA or, in other words, we can say that the z coordinate has zero change. Thus, we have Equation A.2.

$$dist = \sqrt[2]{\Delta X^2 + \Delta Y^2} \tag{A.2}$$

Furthermore, since we know that we are working with small angles, we can thus assume that the declinations are approximately equal, and can be considered to be equal to their average.

$$\delta_1 \approx \delta_2 \approx \delta_{avg} \tag{A.3}$$

By joining A.3 and A.2:

$$\begin{aligned}
\frac{dist}{cos(\delta_{avg})} &= \sqrt[2]{(cos(\alpha_2)^2 - cos(\alpha_1)^2)^2 + (sin(\alpha_2)^2 - sin(\alpha_1)^2)^2)} \\
\left(\frac{dist}{cos(\delta_{avg})}\right)^2 &= 2 - 2 * (cos(\alpha_2)cos(\alpha_1) + sin(\alpha_2)sin(\alpha_1)) \\
\left(\frac{dist}{cos(\delta_{avg})}\right)^2 &= 2 - 2 * cos(\alpha_2 - \alpha_1)
\end{aligned} \tag{A.4}$$

Recalling, from the beginning of this chapter, we said that not only the DEC was small, but also the RA changes were small, which means that $\alpha_2 - \alpha_1$ will be a near zero value and we can apply the small angle approximation in A.4, arriving at Equation A.5.

$$dist = cos(\delta) * (\alpha_2 - \alpha_1) \tag{A.5}$$

Appendix B

Software diagrams

B.1 Data storage classes

B.1.1 DATA class

In Figure B.1 we see the schematic of the *DATA* class, which is where all inputs and outputs of the pipeline are stored inside. The full lines represent the public methods, available to be accessed outside of the class. The dashed lines represent internal function calls to private methods made by the public interface. The dotted lines represent class properties, used to retrieve information from the stars or from the internal status of the class, i.e. if errors were found or if the photometry routine was not used for this object.

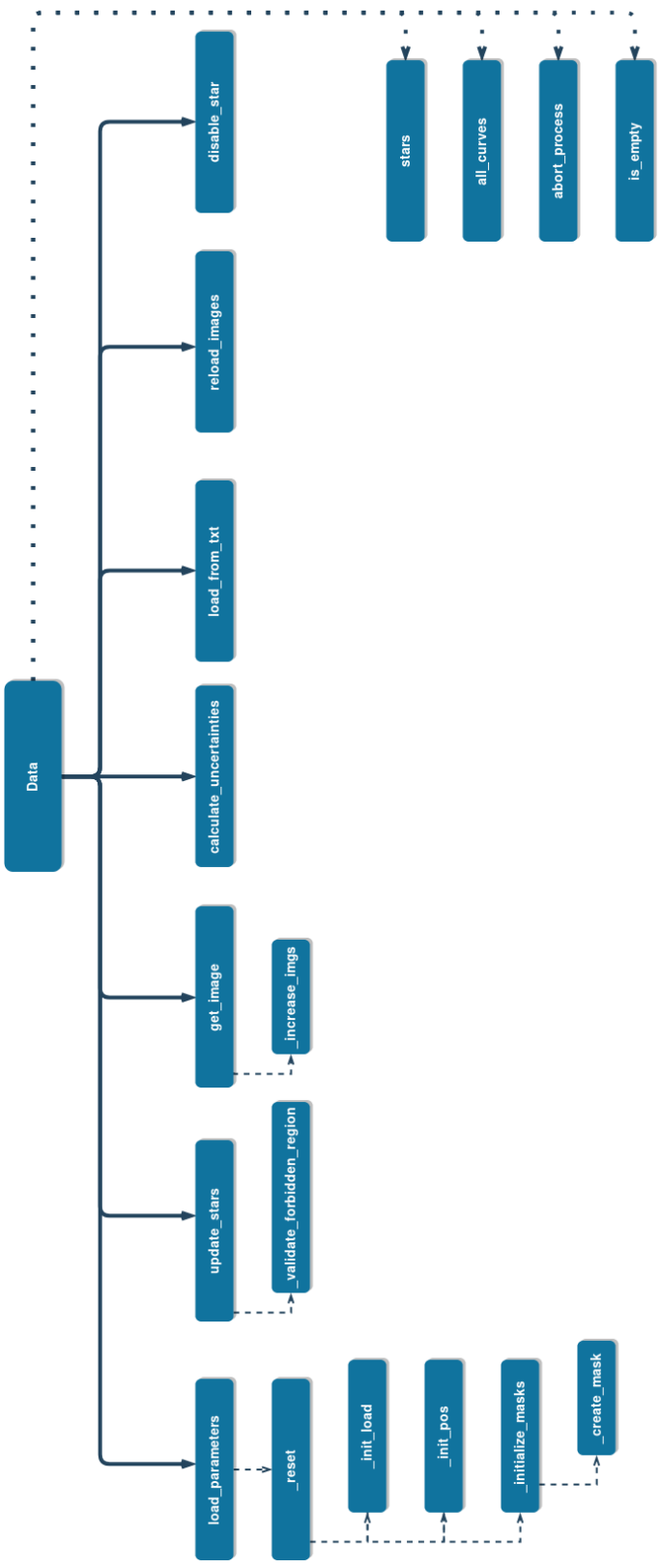


FIGURE B.1: Schematic of the **DATA** class. Full lines represent public methods, dashed lines calls to private methods and dotted lines class properties.

B.2 Photo_controller

In Figure B.2 we have a schematic of the user facing interface of the *Photo_Controller* object. The internal optimize function is a private method to the class but, from the outside, we can still manually trigger the optimization with the "optimize" function.

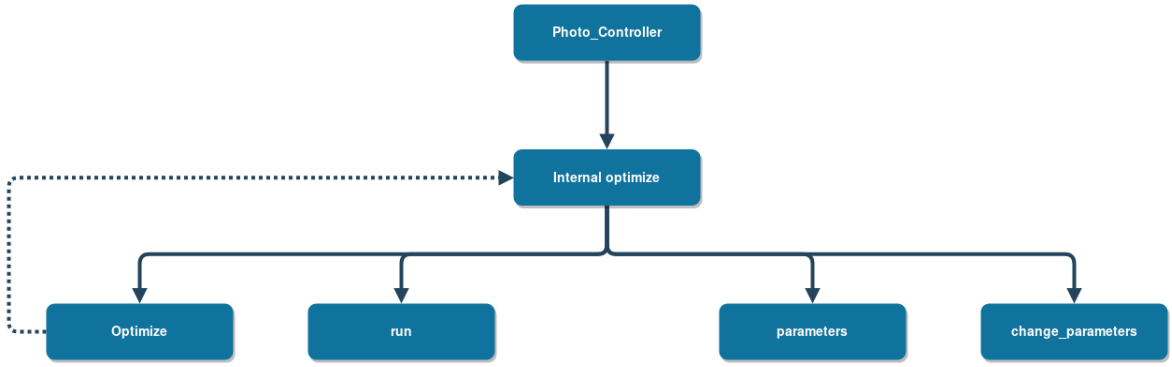


FIGURE B.2: Schematic of the Photo_controller user interface.

B.3 Gaussian Processes

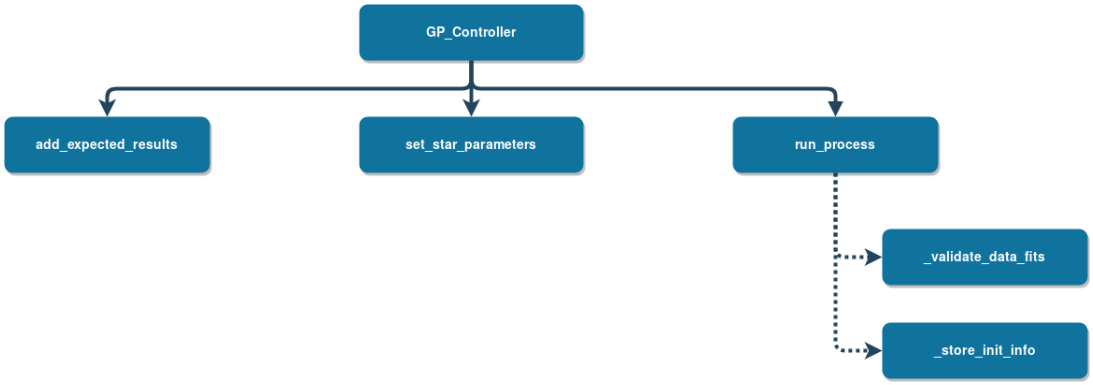


FIGURE B.3: Schematic of the GP_controller user interface.

Appendix C

A brief look into the simulated data sets

In order to fully characterize ARCHI, we used two different data sets, mimicking different situations. We shall use the following notation:

- R_N: Neptune's radius;
- R_J: Jupiter's radius;
- R_S: Star's radius;
- $R_\odot$: Sun's radius;
- R_P: Planet's radius.

The three data sets, named A, B and C, all use the same configuration of stars, as seen in Figure C.1, alongside their name convention. Since we are dealing with simulated Data sets, the star's radius, mass and limb darkening coefficients are given in Table 7 to 12 from [17], depending on the temperature of the star itself. If the value itself is not specified, we performed a linear interpolation with the data immediatly before and after.

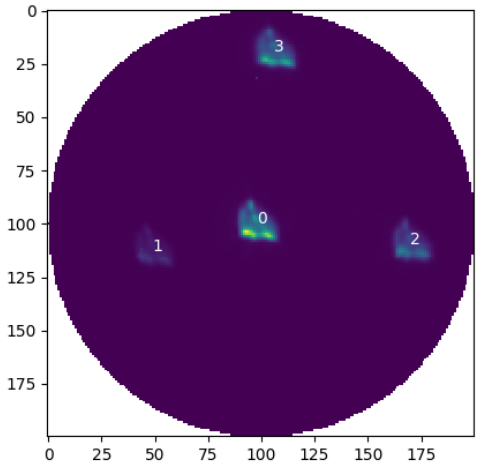


FIGURE C.1: Naming convention for the simulated stars in Data Set A, B and C.

TABLE C.1: Magnitude of each simulated star.

Star	Magnitude [mag]	Teff [K]	Radius [$R_\odot$]	Mass M_{sun}	Limb darkening	
					coeff1	coeff 2
0	9.23	6400	1.39529	1.27823	0.33645	0.30224
1	11.1	5660	0.981	0.98	0.4607	0.4038
2	10.0	5808	1.048	1.037	0.4169	0.2572
3	9.75	4076	1.094	1.343	0.5433	0.7238

Even though the star's configuration is the same, both the number of planets and their parameters are different, as we will now see.

C.1 Data set A

TABLE C.2: Parameter of the planets injected on Data set A.

Star	Planet Radius	Period	Impact parameter [$\frac{1}{R_S}$]	R_P [$\frac{1}{R_S}$]	a [$\frac{1}{R_S}$]	inc [deg]
0	1.106 R_J	0.941 days	0.25	0.07958	3.14027	85.4338

C.2 Data set B

TABLE C.3: Parameter of the planets injected on Data set B.

Star	Planet Radius	Period	Impact parameter [$\frac{1}{R_S}$]	R_P [$\frac{1}{R_S}$]	a [$\frac{1}{R_S}$]	inc [deg]
0	0.5 R_J	0.85 days	0.25	0.03597	2.93441	85.11271
2	1 R_N	2 days	0	0.0253	5.1911	90

C.3 Data set C

TABLE C.4: Parameter of the planets injected on Data set C.

Star	Planet Radius	Period	Impact parameter [$\frac{1}{R_S}$]	R_P [$\frac{1}{R_S}$]	a [$\frac{1}{R_S}$]	inc [deg]
0	1.165 R_J	0.941 days	0.3	0.08383	3.14027	84.51800
1	1 R_J	5 hours	0	0.1535	2.1309	87.3103
2	1.5 R_J	8.5 hours	0.1	0.0957	1.4270	90

Appendix D

Photometric results

Taking into account the number of graphs and tables yielded by the algorithm, this Appendix contains the full information obtained with the different combinations of mask - initial detection - star track that we have.

In the tables, the combination that yields the lowest noise levels is highlighted in green, alongside the mask size that gives those results. Contrastingly, in red we have the combination with the highest noise levels and corresponding mask size.

For Data Set B, due to clear superiority of the *dynam* star tracking method for the background stars, and due to the closeness of the *static* and *offsets* methods for those stars, we decided to only use the *dynam* and *offsets* star tracking methods.

D.1 Data Set A

D.1.1 *Shape* mask

Background grid of 600

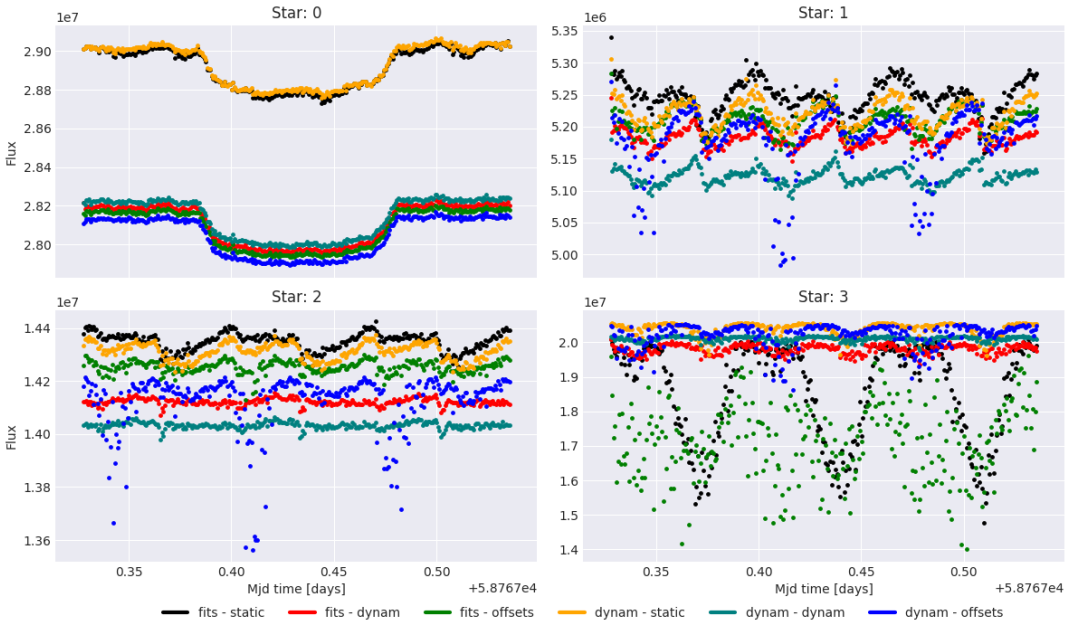


FIGURE D.1: Light curves obtained with all the combinations of methods, using a *shape* mask and a background grid of 600. The name of each curve is used to identify the initial detection method and, afterwards, the star tracking method.

TABLE D.1: Table with the noise, in ppm, for all the Light curves (seen in Figure D.1) using a background grid of 600 and a shape mask.

Methods		Noise (ppm)				Mask's size			
Initial	Track	0	1	2	3	0	1	2	3
fits	static	77.7	308.4	163.0	2741.1	42.0	28.0	26.0	4.0
fits	dynam	53.5	268.7	117.0	728.2	17.0	20.0	16.0	4.0
fits	offsets	50.1	309.9	185.0	9453.0	16.0	23.0	21.0	1.0
dynam	static	69.2	282.5	165.0	543.0	42.0	25.0	24.0	11.0
dynam	dynam	50.2	255.3	102.7	396.7	17.0	15.0	13.0	5.0
dynam	offsets	48.8	414.1	292.5	841.7	15.0	22.0	18.0	10.0

Background grid of 1000

TABLE D.2: Table with the noise, in ppm, for all the Light curves, while using a background grid of 1000 and a shape mask.

Methods		Noise (ppm)				Mask's size			
Initial	Track	0	1	2	3	0	1	2	3
fits	dynam	50.9	266.5	107.9	782.6	37.0	37.0	28.0	6.0
fits	offsets	49.8	285.7	169.8	1846.1	23.0	45.0	42.0	15.0
fits	static	77.1	306.4	162.4	2677.7	62.0	46.0	45.0	7.0
dynam	dynam	48.2	248.2	108.2	339.2	23.0	21.0	25.0	9.0
dynam	offsets	47.4	338.4	197.9	373.9	26.0	56.0	47.0	21.0
dynam	static	68.8	296.3	160.9	474.7	70.0	36.0	37.0	18.0

Background grid of 1400

TABLE D.3: Table with the noise, in ppm, for all the Light curves (while using a background grid of 1400 and a shape mask.

Methods		Noise (ppm)				Mask's size			
Initial	Track	0	1	2	3	0	1	2	3
fits	dynam	49.8	263.3	107.4	721.2	37.0	48.0	39.0	9.0
fits	offsets	50.0	283.6	171.7	1869.3	33.0	61.0	60.0	21.0
fits	static	76.2	305.6	161.6	2628.7	87.0	63.0	62.0	10.0
dynam	dynam	48.8	246.2	101.8	322.7	39.0	30.0	36.0	13.0
dynam	offsets	49.1	338.0	197.5	363.4	32.0	81.0	66.0	29.0
dynam	static	70.1	292.1	160.7	577.1	94.0	50.0	57.0	24.0

Background grid of 1800

TABLE D.4: Table with the noise, in ppm, for all the Light curves (while using a background grid of 1800 and a shape mask.

Methods		Noise (ppm)				Mask's size			
Initial	Track	0	1	2	3	0	1	2	3
fits	static	76.5	304.0	162.6	2557.3	124.0	81.0	79.0	13.0
fits	dynam	48.5	263.9	110.1	732.5	44.0	63.0	52.0	11.0
fits	offsets	49.7	284.7	171.2	2006.8	43.0	84.0	78.0	26.0
dynam	static	69.2	292.9	162.8	552.5	127.0	64.0	72.0	32.0
dynam	dynam	47.0	249.3	106.7	348.1	41.0	38.0	46.0	16.0
dynam	offsets	47.8	335.2	199.2	362.2	47.0	104.0	91.0	37.0

Background grid of 2200

TABLE D.5: Table with the noise, in ppm, for all the Light curves (while using a background grid of 2200 and a shape mask.

Methods		Noise (ppm)				Mask's size			
Initial	Track	0	1	2	3	0	1	2	3
fits	static	218.6	304.5	452.3	2721.5	47.0	100.0	44.0	15.0
fits	dynam	47.3	264.9	109.4	714.5	62.0	82.0	62.0	14.0
fits	offsets	49.2	283.7	171.5	2012.7	51.0	96.0	96.0	32.0
dynam	static	69.2	293.0	161.8	543.8	155.0	78.0	90.0	39.0
dynam	dynam	47.3	246.1	104.5	335.6	50.0	45.0	56.0	20.0
dynam	offsets	48.3	413.0	582.7	371.3	50.0	82.0	47.0	45.0

D.1.2 *Circle* mask

Background grid of 0

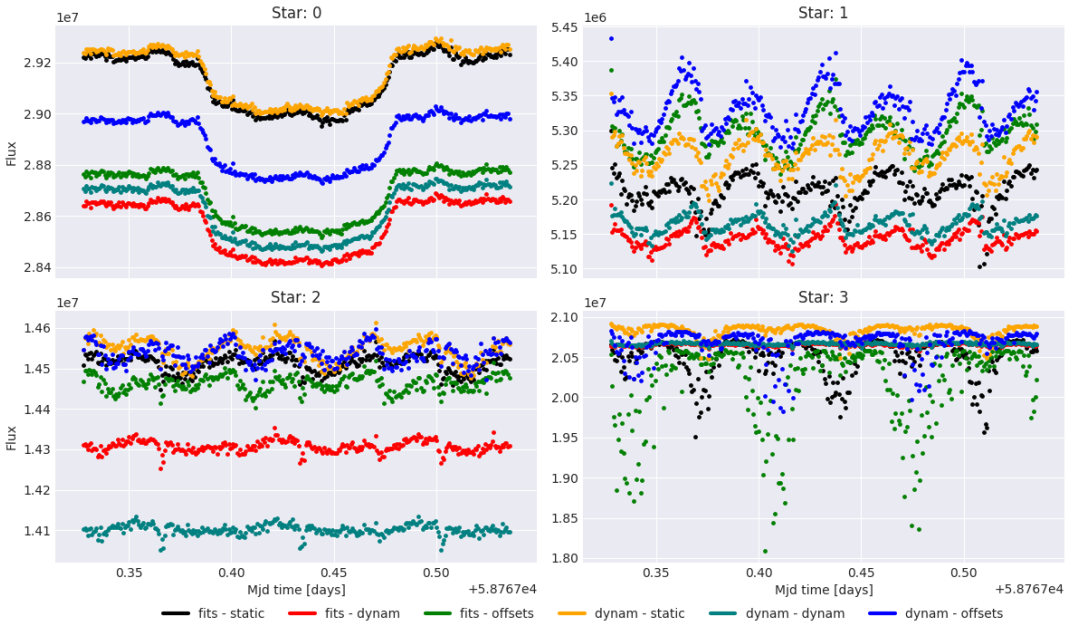


FIGURE D.2: Light curves obtained with all the combinations of methods, using a *circle* mask and a background grid of zero. The color code is used to identify the initial detection method and, afterwards, the star tracking method.

TABLE D.6: Table with the noise, in ppm, for all the Light curves (seen in Figure D.2) using a background grid of 0 and a circle mask.

Methods		Noise (ppm)				Mask's size			
Initial	Track	0	1	2	3	0	1	2	3
fits	static	59.6	303.9	129.4	757.8	30.6	18.0	23.0	16.0
fits	dynam	55.7	250.3	110.7	102.0	20.8	15.0	19.0	15.8
fits	offsets	54.7	322.5	164.5	1305.4	22.2	21.0	22.0	15.0
dynam	static	55.4	310.6	139.1	227.5	31.1	20.0	24.0	18.0
dynam	dynam	52.4	250.0	107.8	112.7	21.5	15.7	16.0	15.9
dynam	offsets	54.9	342.6	184.1	360.3	25.4	23.0	24.0	17.0

Background grid of 600

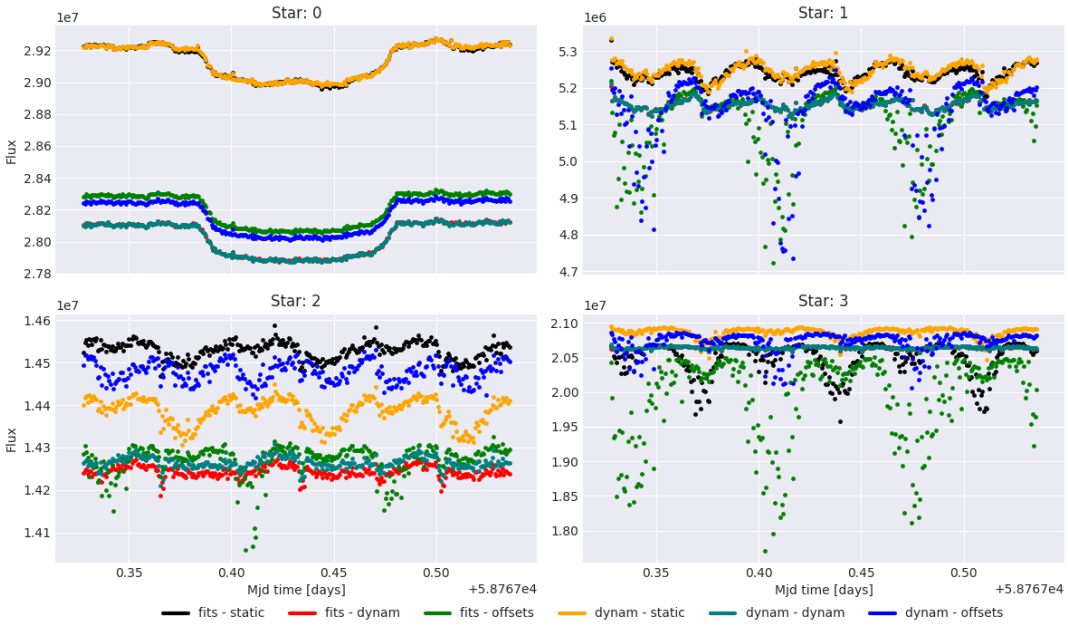


FIGURE D.3: Light curves obtained with all the combinations of methods, using a *circle* mask and a background grid of zero. The color code is used to identify the initial detection method and, afterwards, the star tracking method.

TABLE D.7: Table with the noise, in ppm, for all the Light curves (seen in Figure D.3) using a background grid of 600 and a circle mask.

Methods		Noise (ppm)				Mask's size			
Initial	Track	0	1	2	3	0	1	2	3
fits	static	59.0	276.7	116.0	642.2	92.0	57.0	70.0	48.0
fits	dynam	51.6	247.7	105.3	106.2	49.0	46.0	54.0	47.0
fits	offsets	50.7	769.3	251.6	2230.6	53.0	47.0	56.0	43.0
dynam	static	55.7	276.6	126.4	194.4	92.0	58.0	62.0	55.0
dynam	dynam	52.2	252.3	104.1	112.5	49.0	46.0	55.0	47.0
dynam	offsets	50.4	554.5	182.4	283.7	52.0	50.0	68.0	52.0

Background grid of 1000

TABLE D.8: Table with the noise, in ppm, for all the Light curves, while using a background grid of 1000 and a circle mask.

Methods		Noise (ppm)				Mask's size			
Initial	Track	0	1	2	3	0	1	2	3
fits	dynam	50.0	247.6	104.5	108.4	80.0	75.0	90.0	78.0
fits	offsets	51.0	328.3	162.0	8312.0	78.0	108.1	109.0	31.0
fits	static	58.3	273.9	115.7	573.6	154.0	94.0	116.0	82.0
dynam	dynam	50.7	247.1	105.9	113.6	81.0	77.0	91.0	78.0
dynam	offsets	49.6	345.1	182.6	321.5	78.0	115.0	116.0	85.0
dynam	static	56.4	288.5	126.9	202.0	153.0	96.0	103.0	91.0

Background grid of 1400

TABLE D.9: Table with the noise, in ppm, for all the Light curves, while using a background grid of 1400 and a circle mask.

Methods		Noise (ppm)				Mask's size			
Initial	Track	0	1	2	3	0	1	2	3
fits	dynam	51.2	246.9	104.3	108.2	114.0	106.0	129.0	109.0
fits	offsets	50.2	302.4	161.4	1451.9	118.0	144.0	153.0	102.0
fits	static	58.3	274.6	118.9	545.4	215.0	130.0	144.0	115.0
dynam	dynam	51.0	246.5	104.6	110.9	114.0	109.0	127.0	109.0
dynam	offsets	50.7	342.8	183.6	340.0	110.0	153.0	162.0	119.0
dynam	static	56.4	287.0	127.4	212.0	214.0	135.0	145.0	127.0

Background grid of 1800

TABLE D.10: Table with the noise, in ppm, for all the Light curves, while using a background grid of 1800 and a circle mask.

Methods		Noise (ppm)				Mask's size			
Initial	Track	0	1	2	3	0	1	2	3
fits	static	185.0	273.2	114.7	629.2	156.9	167.0	208.0	148.0
fits	dynam	50.4	246.3	103.6	107.9	147.3	136.0	162.0	140.0
fits	offsets	50.6	301.8	161.9	8297.9	151.6	186.0	196.0	56.0
dynam	static	56.2	285.8	127.2	200.3	274.9	173.0	185.0	164.0
dynam	dynam	49.8	248.3	104.5	110.1	146.2	137.0	162.0	140.0
dynam	offsets	50.0	338.6	184.4	340.6	141.6	197.0	207.0	153.0

Background grid of 2200

TABLE D.11: Table with the noise, in ppm, for all the Light curves, while using a background grid of 2200 and a circle mask.

Methods		Noise (ppm)				Mask's size			
Initial	Track	0	1	2	3	0	1	2	3
fits	static	58.2	272.4	115.1	566.6	337.0	205.0	255.0	181.0
fits	dynam	49.4	247.1	104.8	108.2	177.0	165.0	200.0	171.3
fits	offsets	50.3	8599.8	568.3	8386.2	172.0	55.0	172.9	68.0
dynam	static	56.2	286.0	127.1	199.1	335.7	212.2	227.0	201.0
dynam	dynam	49.8	245.0	103.5	110.0	179.0	170.0	197.0	171.5
dynam	offsets	50.6	408.3	290.8	321.9	173.0	198.0	203.0	186.5

D.2 Data Set B

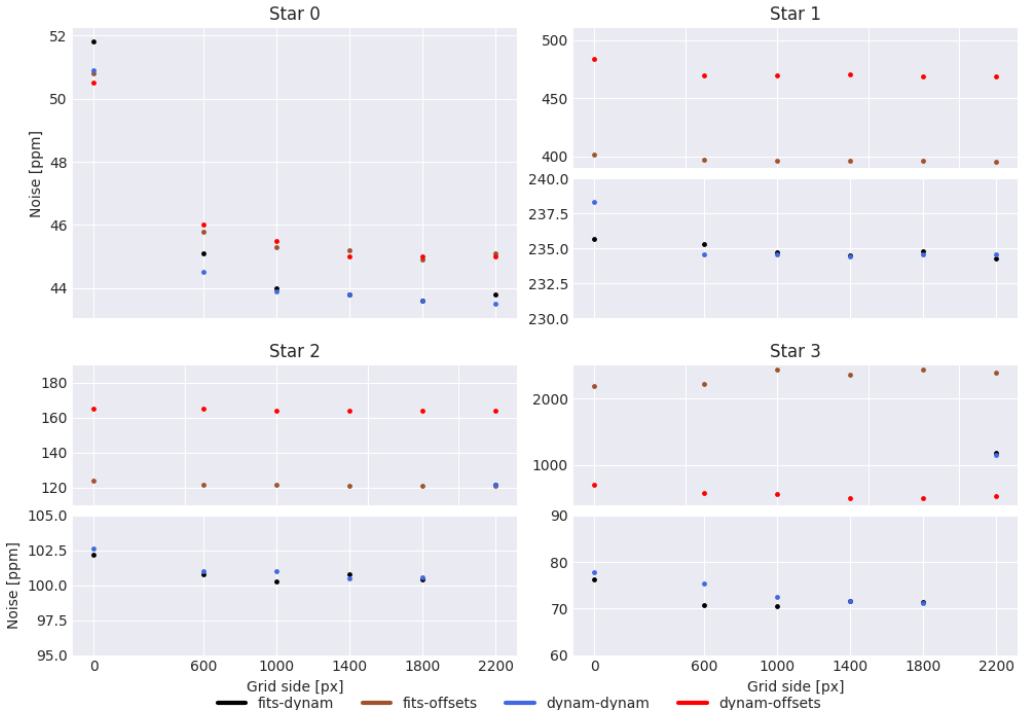


FIGURE D.4: Evolution of the noise with the increase of the background grid, for each possible combination and a *circle* mask, in Data Set B, with the DRP's CDPP algorithm

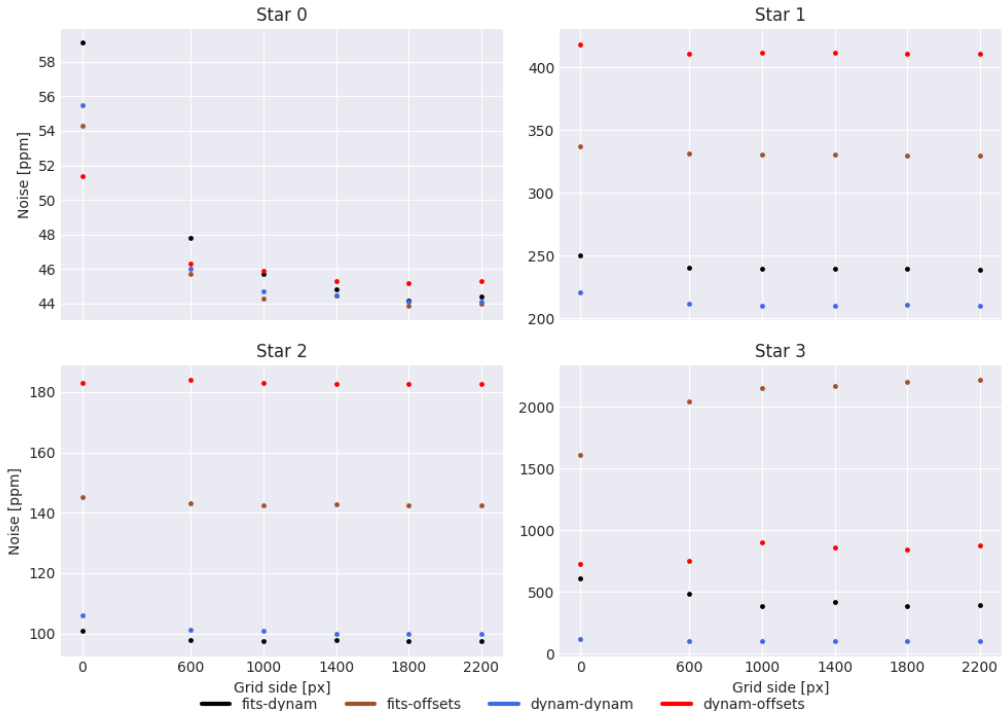


FIGURE D.5: Evolution of the noise with the increase of the background grid, for each possible combination and a *shape* mask, in Data Set B, with the DRP's CDPP algorithm.

Appendix E

How to use ARCHI

E.1 Configuration values

```
##########################################
#               Paths                    #
##########################################

# folder in which the fits files are stored
base_folder: ""
# path to save and search for the optimized radius file
optimized_factors: ""
# folder to store the graphs and related information
results_folder: ""
#name of the curve to be used as the comparison basis. Options pass by: DEFAULT;
    OPTIMAL; RINF; RSUP
official_curve: "OPTIMAL"

##########################################
#      General configurations            #
##########################################

# method used on the mask creation. shape/circle
method: "shape"
#How the initial centers are determined. possible values:  fits/dynam
initial_detect: "fits"
# uses opencv to calculate the center of each star for each frame dynam/offsets/
    static
```

```
detect_mode: "dynam"
# Calculate uncertainties in each point
uncertainties: 0
# size of the background grid. Has to be multiple of 200
grid_bg: 0


##########################################
#          Optimization confs            #
##########################################

# enable optimization of the increase factors
optimize: 0
# number of times that the optimization process is expanded
optimization_extensions: 6
# Number of cores to use for the optimization process. if it's running on a
    laptop, it's recommended to use 2 or 3
optim_processes: 3
# fine search for the mask best size. The step keyword has no effect over this
    process
fine_tune_circle: 1
# values that the factor can take -> used in the optimization process
val_range: [1,10]
# Step used to increase the masks, during the optimization process
step: 1

##########################################
#          Run time processes            #
##########################################

# 0 if the code is being run on computer with GUI
headless: 0
# Use the low memory mode
low_memory: 0
# K2 or DRP
CDPP_type: "K2"
# compares the method against the official pipeline
debug: 1
# Show photometric curve for each star
```

```
show_results: 1
# shows the images and masks in real time
plot_realtime: 0

##########################################
#      Data storage/ file handling       #
##########################################

# create various pictures to the reports
report_pictures: 0
# export the photometric data to a txt as well as the one from the official
    pipeline
export_text: 1
# export the photometric data to a fit file, as well as relevant information
export_fit: 0
# export images and information to a pdf file
export_pdf: 0
```

E.2 Simple Photometry run

In order to use ARCHI, one has to start by setting the desired configurations in the configuration file, presented in Appendix E.1. After that step, it's only needed to call the desired functions, to achieve the desired functionality. In the script given bellow, it's assumed that the user wants to pass some information via the command line. However, if one does not wish to do so, the user simply has to manually change the parameters to the desired values.

```
import ARCHI
import sys


def start_process():
    """
    Uses both the Photo_controller and the GP_controller to run a full routine of
     analysis under a data set. One can pass the configuration values from
    the command line, when calling the script.

    """
```

```
# Command line arguments passed to the script by supernova/user
job_number = sys.argv[1]
first_burn = int(sys.argv[2])
second_brun = int(sys.argv[3])
prod = int(sys.argv[4])
procs = int(sys.argv[5])


data_fits = ARCHI.Data()

# Photometry part
controller = ARCHI.Photo_controller(job_number, config_path="/home/amiguel/
work/configs/config_cluster.yaml")

data_fits = controller.run(data_fits)

# Gaussian Processes

gp_cont = ARCHI.GP_controller(burns=[first_burn, second_brun, prod],
job_number=job_number,
                        results_folder=controller.parameters["results_folder"
], nwalkers=128)

# Set the parameters of the star that hosts the planet under analysis
gp_cont.set_star_parameters({
                                0: {
                                    'RS': 1.3952941176470586,
                                    "MS": 1.2782352941176471,
                                    'limb_type': "quadratic",
                                    "limb_coefs": [0.33645483523200004,
0.3022439091652]
                                }
                            })

# set the truth values
expected_results = [0.07958153054340054, 3.140278199264141,
85.11270640107678]
```

```
44
    gp_cont.add_expected_results(0, values=expected_results)
46
    # Initial guess of the parameters that will be fitted
48
    init_guess = [{'rp': 0.07, 'a': 3.0, 'inc': 80.0, 't0': 0}]
50  data_fits = gp_cont.run_gps(data_fits, nthreads=controller.parameters['
    optim_processes'],
                                initial_guess=init_guess,
52                              stars=[2])

54  # Store all of the available information
    ARCHI.store_data(data_fits, job_number, **controller.parameters)
56
if __name__ == '__main__':
58  start_process()
```

Bibliography

[1] M. Mayor and D. Queloz, "A jupiter-mass companion to a solar-type star," *Nature*, vol. 378, no. 6555, pp. 355–359, nov 1995.

[2] D. Cenadelli and A. Bernagozzi, "The discovery of the first exoplanets," in *Handbook of Exoplanets*. Springer International Publishing, 2018, pp. 3–20.

[3] H. J. Deeg and J. A. Belmonte, "Impact of exoplanet science in the early twenty-first century," in *Handbook of Exoplanets*. Springer International Publishing, 2018, pp. 95–113.

[4] D. Briot and J. Schneider, "Prehistory of transit searches," in *Handbook of Exoplanets*. Springer International Publishing, 2018, pp. 35–49.

[5] W. J. Borucki, "KEPLERMission: development and overview," *Reports on Progress in Physics*, vol. 79, no. 3, p. 036901, feb 2016.

[6] G. R. Ricker, J. N. Winn, R. Vanderspek, D. W. Latham, G. Á. Bakos, J. L. Bean, Z. K. Berta-Thompson, T. M. Brown, L. Buchhave, N. R. Butler, R. P. Butler, W. J. Chaplin, D. Charbonneau, J. Christensen-Dalsgaard, M. Clampin, D. Deming, J. Doty, N. D. Lee, C. Dressing, E. W. Dunham, M. Endl, F. Fressin, J. Ge, T. Henning, M. J. Holman, A. W. Howard, S. Ida, J. M. Jenkins, G. Jernigan, J. A. Johnson, L. Kaltenegger, N. Kawai, H. Kjeldsen, G. Laughlin, A. M. Levine, D. Lin, J. J. Lissauer, P. MacQueen, G. Marcy, P. R. McCullough, T. D. Morton, N. Narita, M. Paegert, E. Palle, F. Pepe, J. Pepper, A. Quirrenbach, S. A. Rinehart, D. Sasselov, B. Sato, S. Seager, A. Sozzetti, K. G. Stassun, P. Sullivan, A. Szentgyorgyi, G. Torres, S. Udry, and J. Villasenor, "Transiting exoplanet survey satellite," *Journal of Astronomical Telescopes, Instruments, and Systems*, vol. 1, no. 1, p. 014003, oct 2014.

[7] E. Jehin, M. Gillon, D. Queloz, P. Magain, J. Manfroid, V. Chantry, M. Lendl, D. Hutsemékers, and S. Udry, "TRAPPIST: TRAnsiting Planets and PlanetesImals Small Telescope," *The Messenger*, vol. 145, pp. 2–6, Sep. 2011.

[8] H. Rauer and A. M. Heras, "Space missions for exoplanet science: PLATO," in *Handbook of Exoplanets*. Springer International Publishing, 2018, pp. 1309–1330.

[9] V. S. Meadows and R. K. Barnes, "Factors affecting exoplanet habitability," in *Handbook of Exoplanets*. Springer International Publishing, 2018, pp. 2771–2794.

[10] *CHEOPS - Definition Study Report*.

[11] L. P. da Silva, G. Rolland, V. Lapeyrere, and M. Auvergne, "Radiation effects on space-based stellar photometry: theoretical models and empirical results for CoRoT space telescope," *Monthly Notices of the Royal Astronomical Society*, vol. 384, no. 4, pp. 1337–1343, mar 2008.

[12] L. Pinheiro da Silva, V. Lapeyrere, and P. Bernardi, "Photometric Calibration," in *The CoRoT Mission Pre-Launch Status - Stellar Seismology and Planet Finding*, ser. ESA Special Publication, M. Fridlund, A. Baglin, J. Lochard, and L. Conroy, Eds., vol. 1306, Nov 2006, p. 309.

[13] O. S. M. S.Hoyer, P.Guterman and JC.Meunier, "Data reduction pipeline of cheops mission."

[14] *CHEOPS Observers Manual*.

[15] *Data reduction procedures for the on-ground payload calibration*.

[16] *CHEOPS: Data Products Definition Document*.

[17] *CHEOPSim User Guide*.

[18] "Schematic of the right ascension and declination," Online in http://blog.leapmotion.com/introducing-planetarium-design-science-behind-vr-widgets-showcase/. Accessed in: 01/07/19.

[19] M. Kennedy and S. Kopp, "Understanding map projections," in *Understanding Map Projections*. ESRI Press, 2001.

[20] H. Karttunen, P. Kröger, H. Oja, M. Poutanen, and K. J. Donner, Eds., *Fundamental Astronomy*. Springer Berlin Heidelberg, 2007.

[21] H. J. Deeg and R. Alonso, "Transit photometry as an exoplanet discovery method," in *Handbook of Exoplanets*. Springer International Publishing, 2018, pp. 633–657.

[22] Nasa, "The transit method," https://heasarc.gsfc.nasa.gov/docs/tess/primary-science.html.

[23] J. N. Winn, "Transits and occultations."

[24] J. M. Almenara, H. J. Deeg, S. Aigrain, R. Alonso, M. Auvergne, A. Baglin, M. Barbieri, P. Barge, P. Bordé, F. Bouchy, H. Bruntt, J. Cabrera, L. Carone, S. Carpano, C. Catala, S. Csizmadia, R. D. la Reza, M. Deleuil, R. Dvorak, A. Erikson, M. Fridlund, D. Gandolfi, M. Gillon, P. Gondoin, E. Guenther, T. Guillot, A. Hatzes, G. Hébrard, L. Jorda, H. Lammer, A. Léger, A. Llebaria, B. Loeillet, P. Magain, M. Mayor, T. Mazeh, C. Moutou, M. Ollivier, M. Pätzold, F. Pont, D. Queloz, H. Rauer, C. Régulo, S. Renner, D. Rouan, B. Samuel, J. Schneider, A. Shporer, G. Wuchterl, and S. Zucker, "Rate and nature of false positives in the CoRoT exoplanet search," *Astronomy & Astrophysics*, vol. 506, no. 1, pp. 337–341, aug 2009.

[25] D. Charbonneau, "Astrophysical false positives encountered in wide-field transit searches," in *AIP Conference Proceedings*. AIP, 2004.

[26] A. Santerne, R. F. Díaz, J. M. Almenara, A. Lethuillier, M. Deleuil, and C. Moutou, "Astrophysical false positives in exoplanet transit surveys: why do we need bright stars ?"

[27] F. Fressin, G. Torres, D. Charbonneau, S. T. Bryson, J. Christiansen, C. D. Dressing, J. M. Jenkins, L. M. Walkowicz, and N. M. Batalha, "THE FALSE POSITIVE RATE OFKEPLERAND THE OCCURRENCE OF PLANETS," *The Astrophysical Journal*, vol. 766, no. 2, p. 81, mar 2013.

[28] N. C. Santos and J. P. Faria, "Exoplanetary science: An overview," in *Astrophysics and Space Science Proceedings*. Springer International Publishing, jul 2017, pp. 165–180.

[29] I. Boisse, F. Bouchy, G. Hébrard, X. Bonfils, N. Santos, and S. Vauclair, "Disentangling between stellar activity and planetary signals," *Astronomy & Astrophysics*, vol. 528, p. A4, feb 2011.

[30] A. C. Cameron, "The impact of stellar activity on the detection and characterization of exoplanets," in *Handbook of Exoplanets*. Springer International Publishing, 2018, pp. 1791–1799.

[31] P. Ballerini, G. Micela, A. F. Lanza, and I. Pagano, "Multiwavelength flux variations induced by stellar magnetic activity: effects on planetary transits," *Astronomy & Astrophysics*, vol. 539, p. A140, mar 2012.

[32] K. Masuda, "Exploring the architecture of transiting exoplanetary systems with high-precision photometry," Ph.D. dissertation, University of Tokyo, Tokyo, Japan, 2016.

[33] M. Oshagh, N. C. Santos, I. Boisse, G. Boué, M. Montalto, X. Dumusque, and N. Haghighipour, "Effect of stellar spots on high-precision transit light-curve," *Astronomy & Astrophysics*, vol. 556, p. A19, jul 2013.

[34] S. Czesla, K. F. Huber, U. Wolter, S. Schröter, and J. H. M. M. Schmitt, "How stellar activity affects the size estimates of extrasolar planets," *Astronomy & Astrophysics*, vol. 505, no. 3, pp. 1277–1282, aug 2009.

[35] H. Parviainen and S. Aigrain, "ldtk: Limb darkening toolkit," *Monthly Notices of the Royal Astronomical Society*, vol. 453, no. 4, pp. 3822–3827, sep 2015.

[36] N. Espinoza and A. Jordán, "Limb darkening and exoplanets: testing stellar model atmospheres and identifying biases in transit parameters," *Monthly Notices of the Royal Astronomical Society*, vol. 450, no. 2, pp. 1879–1899, apr 2015.

[37] H. Çuha and A. Erdem, "Limb darkening effect on transit light curves of HAT-p-32b." Author(s), 2018.

[38] A. Claret, "Limb and gravity-darkening coefficients for the TESS satellite at several metallicities, surface gravities, and microturbulent velocities," *Astronomy & Astrophysics*, vol. 600, p. A30, mar 2017.

[39] H. M. Müller, "Limb-darkening measurements on exoplanet host stars and the sun," Ph.D. dissertation, Fachbereichs Physikder Universität Hamburg, 2015.

[40] G. S. Da Costa, "Basic Photometry Techniques," in *Astronomical CCD Observing and Reduction Techniques*, ser. Astronomical Society of the Pacific Conference Series, S. B. Howell, Ed., vol. 23, Jan 1992, p. 90.

[41] R. R. Laher, V. Gorjian, L. M. Rebull, F. J. Masci, J. W. Fowler, G. Helou, S. R. Kulkarni, and N. M. Law, "Aperture photometry tool," *Publications of the Astronomical Society of the Pacific*, vol. 124, no. 917, pp. 737–763, jul 2012.

[42] A. Popowicz and B. Smolka, "A method of complex background estimation in astronomical images," *Monthly Notices of the Royal Astronomical Society*, vol. 452, no. 1, pp. 809–823, jul 2015.

[43] P. Teeninga, U. Moschini, S. C. Trager, and M. H. Wilkinson, "Improving background estimation for faint astronomical object detection," in *2015 IEEE International Conference on Image Processing (ICIP)*. IEEE, sep 2015.

[44] C. Sterken and J. Manfroid, *Astronomical Photometry, a Guide*. Springer Netherlands, 1992.

[45] P. C. Austin, L. J. Brunner, and J. E. H. M. SM, "Bayeswatch: an overview of bayesian statistics," *Journal of Evaluation in Clinical Practice*, vol. 8, no. 2, pp. 277–286, may 2002.

[46] G. "Andrew, C. John, S. Hal, D. David, V. Aki, and R. Donald, *Bayesian Data Analysis, third edition*. CHAPMAN & HALL/CRC, 2004.

[47] H. Parviainen, "Bayesian methods for exoplanet science," in *Handbook of Exoplanets*. Springer International Publishing, 2018, pp. 1567–1590.

[48] N. D. Verhelst, "An efficient MCMC algorithm to sample binary matrices with fixed marginals," *Psychometrika*, vol. 73, no. 4, pp. 705–728, apr 2008.

[49] S. S. Qian, C. A. Stow, and M. E. Borsuk, "On monte carlo methods for bayesian inference," *Ecological Modelling*, vol. 159, no. 2-3, pp. 269–277, jan 2003.

[50] L. Rocha, L. Velho, and P. Carvalho, "Image moments-based structuring and tracking of objects," in *Proceedings. XV Brazilian Symposium on Computer Graphics and Image Processing*. IEEE Comput. Soc.

[51] OpenCv. (2019, Jan.) Opencv python official python tutorial - contour features. Online. [Online]. Available: https://docs.opencv.org/3.1.0/dd/d49/tutorial_py_contour_features.html

[52] P. M. M. da Silva, "Pixel level decorrelation for cheops," Master's thesis, Faculdade de Ciências da UNiversidade do Porto, 2019.

[53] J. L. Christiansen, J. M. Jenkins, D. A. Caldwell, C. J. Burke, P. Tenenbaum, S. Seader, S. E. Thompson, T. S. Barclay, B. D. Clarke, J. Li, J. C. Smith, M. C. Stumpe, J. D. Twicken, and J. V. Cleve, "The derivation, properties, and value of kepler's combined differential photometric precision," *Publications of the Astronomical Society of the Pacific*, vol. 124, no. 922, pp. 1279–1287, dec 2012.

[54] D. G. Koch, W. J. Borucki, G. Basri, N. M. Batalha, T. M. Brown, D. Caldwell, J. Christensen-Dalsgaard, W. D. Cochran, E. DeVore, E. W. Dunham, T. N. Gautier, J. C. Geary, R. L. Gilliland, A. Gould, J. Jenkins, Y. Kondo, D. W. Latham, J. J. Lissauer, G. Marcy, D. Monet, D. Sasselov, A. Boss, D. Brownlee, J. Caldwell, A. K. Dupree, S. B. Howell, H. Kjeldsen, S. Meibom, D. Morrison, T. Owen, H. Reitsema, J. Tarter, S. T. Bryson, J. L. Dotson, P. Gazis, M. R. Haas, J. Kolodziejczak, J. F. Rowe, J. E. V. Cleve, C. Allen, H. Chandrasekaran, B. D. Clarke, J. Li, E. V. Quintana, P. Tenenbaum, J. D. Twicken, and H. Wu, "KEPLER MISSIONDESIGN, REALIZED PHOTOMETRIC PERFORMANCE, AND EARLY SCIENCE," *The Astrophysical Journal*, vol. 713, no. 2, pp. L79–L86, mar 2010.

[55] J. M. Jenkins, H. Chandrasekaran, S. D. McCauliff, D. A. Caldwell, P. Tenenbaum, J. Li, T. C. Klaus, M. T. Cote, and C. Middour, "Transiting planet search in the kepler pipeline," in *Software and Cyberinfrastructure for Astronomy*, N. M. Radziwill and A. Bridger, Eds. SPIE, jul 2010.

[56] R. Luger, E. Agol, E. Kruse, R. Barnes, A. Becker, D. Foreman-Mackey, and D. Deming, "EVEREST: PIXEL LEVEL DECORRELATION OFK2 LIGHT CURVES," *The Astronomical Journal*, vol. 152, no. 4, p. 100, oct 2016.

[57] R. L. Gilliland, W. J. Chaplin, E. W. Dunham, V. S. Argabright, W. J. Borucki, G. Basri, S. T. Bryson, D. L. Buzasi, D. A. Caldwell, Y. P. Elsworth, J. M. Jenkins, D. G. Koch, J. Kolodziejczak, A. Miglio, J. van Cleve, L. M. Walkowicz, and W. F. Welsh, "KEPLERMISSION STELLAR AND INSTRUMENT NOISE PROPERTIES," *The Astrophysical Journal Supplement Series*, vol. 197, no. 1, p. 6, oct 2011.

[58] C. E. Rasmussen and C. K. I. Williams", *Gaussian Processes for Machine Learning*. The MIT press, 2006, available Online: http://www.gaussianprocess.org/gpml/.

[59] S. Aigrain, S. T. Hodgkin, M. J. Irwin, J. R. Lewis, and S. J. Roberts, "Precise time series photometry for the kepler-2.0 mission," *Monthly Notices of the Royal Astronomical Society*, vol. 447, no. 3, pp. 2880–2893, jan 2015.

[60] N. P. Gibson, S. Aigrain, S. Roberts, T. M. Evans, M. Osborne, and F. Pont, "A gaussian process framework for modelling instrumental systematics: application to transmission spectroscopy," *Monthly Notices of the Royal Astronomical Society*, vol. 419, no. 3, pp. 2683–2694, nov 2011.

[61] J. D. R. Camacho, "The use of gaussian processes in the analysis of stellar noise in exoplanet search," Master's thesis, Faculdade de Ciências da Universidade do Porto, 2017.

[62] B. A. Swastanto, "Gaussian process regression for long-term time series forecasting," Master's thesis, FAculty of Electrical Engineering, Mathematics, and Computer Science, Delft University of Technology,, 2016.

[63] N. P. Gibson, "Reliable inference of exoplanet light-curve parameters using deterministic and stochastic systematics models," *Monthly Notices of the Royal Astronomical Society*, vol. 445, no. 4, pp. 3401–3414, oct 2014.

[64] D. Duvenaud, "The kernel cookbook: Advice on covariance functions," Online: https://www.cs.toronto.edu/~duvenaud/cookbook/. Accessed on 19/8/2019.

[65] D. K. Duvenaud, "Automatic model construction with gaussian processes," Ph.D. dissertation, Apollo - University of Cambridge Repository, 2014.

[66] S. Roberts, M. Osborne, M. Ebden, S. Reece, N. Gibson, and S. Aigrain, "Gaussian processes for time-series modelling," *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 371, no. 1984, pp. 20 110 550–20 110 550, dec 2012.

[67] L. Kreidberg, "batman: BAsic transit model cAlculatioN in python," *Publications of the Astronomical Society of the Pacific*, vol. 127, no. 957, pp. 1161–1165, nov 2015.

[68] D. Foreman-Mackey, D. W. Hogg, D. Lang, and J. Goodman, "emcee: The MCMC hammer," *Publications of the Astronomical Society of the Pacific*, vol. 125, no. 925, pp. 306–312, mar 2013.

[69] george. Python package: george. [Online]. Available: https://george.readthedocs.io/en/latest/tutorials/scaling/

[70] M. SEEGER, "GAUSSIAN PROCESSES FOR MACHINE LEARNING," *International Journal of Neural Systems*, vol. 14, no. 02, pp. 69–106, apr 2004.

[71] J. Gill and D. Hangartner, "Circular data in political science and how to handle it," *Political Analysis*, vol. 18, no. 3, pp. 316–336, 2010.

[72] P. Guerrero and J. R. del Solar, "Circular regression based on gaussian processes," in *2014 22nd International Conference on Pattern Recognition*. IEEE, aug 2014.

[73] G. Nuñez-Antonio, E. Gutiérrez-Peña, and G. Escarela, "A bayesian regression model for circular data based on the projected normal distribution," *Statistical Modelling: An International Journal*, vol. 11, no. 3, pp. 185–201, may 2011.

[74] "Statistics and data analysis in geology," *Earth-Science Reviews*, vol. 9, no. 4, p. 388, dec 1973.

[75] R. J. Protheroe, "A new statistic for the analysis of circular data in gamma-ray astronomy." in *19th International Cosmic Ray Conference (ICRC19), Volume 3*, ser. International Cosmic Ray Conference, vol. 3, Aug 1985, pp. 485–488.

[76] E. Padonou and O. Roustant, "Polar gaussian processes for predicting on circular domains," 04 2015.

[77] S. R. Jammalamadaka and A. SenGupta, *Topics in Circular Statistics*. WORLD SCIENTIFIC, apr 2001.

[78] "Emcee documentation," Online. http://dfm.io/emcee/current/user/line/.

[79] D. F.-M. . contributors, "corner.py," Online. https://corner.readthedocs.io/en/latest/.

[80] D. Foreman-Mackey, "corner.py: Scatterplot matrices in python," *The Journal of Open Source Software*, vol. 24, 2016. [Online]. Available: http://dx.doi.org/10.5281/zenodo.45906

[81] "Slurm documentation," Online. https://slurm.schedmd.com/documentation.html.

[82] G. Kovács, S. Zucker, and T. Mazeh, "A box-fitting algorithm in the search for periodic transits," *Astronomy & Astrophysics*, vol. 391, no. 1, pp. 369–377, jul 2002.

[83] E. Weisstein, "Great circle," From MathWorld–A Wolfram Web Resource. http://mathworld.wolfram.com/GreatCircle.html. Acessed in 01/07/19.