Radboud University

Master's Thesis

Making and Analysing Simulated Images of Globular Clusters for MICADO @ ELT

Author:
Luc IJspeert

Supervisor:
Dr. Søren Larsen

A thesis submitted in fulfilment of the requirements for the degree of Master of Science

in the

Department of Astrophysics
Faculty of Science
Radboud University

July 2019

Contents

List of abbreviations

AGB Asymptotic Giant Branch

AO Adaptive Optics

CDF Cumulative Distribution Function

CMD Colour-Magnitude Diagram

ELT Extremely Large Telescope

HB Horizontal Branch

HLR Half-Light Radius

HRD Hertzsprung-Russel Diagram

IMF Initial Mass Function

LTAO Laser Tomography Adaptive Optics

MCAO Multi-Conjugate Adaptive Optics

MS Main Sequence

NGS Natural Guide Star

PDF Probability Density Function

PSF Point Spread Function

SCAO Single Conjugate Adaptive Optics

1 Abstract

The Extremely Large Telescope (ELT) is in the process of being built at this moment, and is scheduled for first light in 2025. It will be the largest optical telescope in the world for years to come after that. At this moment, the largest telescopes that cover visible and infrared wavelengths are a little over ten meters in diameter: the ELT will almost quadruple that. This leads to high expectations, especially in the astronomical community. Some of the expectations of this impressive device are quantified, within the field of globular clusters. Models of globular clusters are built to obtain realistic positions and magnitudes that can subsequently be used to make synthetic images of these models. The images are simulated with `SimCADO`, taking into account the atmosphere, the telescope performance and the imaging instrument at first light (MICADO).

The generated clusters cover a range of distances of the order of nearby galaxies, and vary in mass and size. The age is set to a typical 10 Gyr for globulars, and the metallicity of about $0.1Z_{\odot}$ is also representative of this type of cluster. The images are made with exposure times of 30 minutes and a pixel scale of 4 milli-arcseconds. Photometry is performed with IRAFs implementation of the DAOPhot algorithms by Stetson (1987). The position measurements show good agreement with the model clusters, and outliers can succesfully be identified based on a large deviation of the measured and real positions. The magnitude accuracy is a strong function of both magnitude and radius. Both are not unexpected in crowded stellar fields due to the difficulty of measuring the light from one individual star where many others are present. In the centre of clusters, magnitudes can deviate by many magnitudes, while on the far outskirts, a deviation of up to half a magnitude is more typical.

Some strange features are identified in the photometry, in both the difference between measured and real magnitude and in the colours of the stellar populations. The cause remains unknown, although errors in either photometry or image simulations seem most likely at this point. Future projects could expand on the parameters used in this research, while saving on time by using the code written here to produce the globular clusters. Most gain can probably be made by investigating a range of total exposure times, making use of short individual exposures that are added together later.

2 Introduction

Telescopes have become bigger and more powerful over the course of history and as technological advancements allowed. The two main motivations for building larger main telescope mirrors[1] are on the one hand to get a larger light collecting area and thus the capability to see fainter objects, and on the other hand to get better resolving power. The smallest angular separation a telescope can resolve is theoretically related to its size with the simple formula $\theta \approx \lambda/D$, where λ is the observed wavelength, D is the diameter of the aperture and θ is the diffraction limit in radians. Unfortunately, if a telescope is placed on the surface of earth, the atmosphere interferes[2]. This will limit the angular resolution to the seeing at the site of observation, which is generally about 1 arcsecond (also denoted with $''$) or $0.4''$ in optimal conditions. The diameter at which a telescope reaches this resolution for red light is under half a meter; increasing the aperture size beyond that would not increase the angular resolution.

So if we want better observations, and we do, then we might try one of two things: lift our telescopes into space above the atmosphere or find another creative solution. In space there is no atmosphere to inhibit the propagation of our valuable photons. However, it is still very expensive to get large, heavy satellites into orbit or further out into space. Plus, while cost is a factor, it is overshadowed by the immense practical difficulty of getting something so delicate launched safely, after which it would have to be assembled or preferably assemble itself. On top of that, it is a non-trivial, if not impossible task to adapt, upgrade or even repair such a machine. The perfect example here is the largest space telescope in assembly today: the James Webb Space Telescope (JWST) (NASA, website 2019[a]) with its 6.5 meter primary mirror that folds up to fit inside a rocket. The JWST will not orbit the earth; instead it will be positioned in the second Lagrangian point (L2) of the Earth-Sun system. This will mean that it is unfeasible to reach it for servicing missions, something that *has* happened several times for the Hubble Space Telescope (NASA, website 2019[b]).

This brings me back to the point of technological advancements. Fairly recently in the history of telescopes, so called adaptive optics (AO) have become reality. These systems aim to negate the blurring effect of the atmosphere by analysing the wave fronts of the incoming light and changing the shape of one or more of the telescope mirrors to compensate for the deformation. This of course is easier said than done, which is why this is still a field of ongoing development at this time. The results, however, should bring us close to the diffraction limit of the telescope it is applied to. This is why ground based telescopes have grown in size far past the limit imposed on them based on their earthly tethers.

2.1 A new size class

There are already several established implementations of AO in observatories like the Keck II telescope and the Very Large Telescope (Wizinowich et al., 2000; Hippler, 2019). In these cases, the AO was added some time after they were taken into operation. The next generation of large ground based telescopes will be built with AO systems as an integral part of their design, enabling diffraction limited observations from first light. The now under construction (European) Extremely Large Telescope (ELT) (Gilmozzi and Spyromilio, 2007) will be the largest among the first extremely large telescopes, alongside the planned Thirty Meter Telescope (Sanders, 2013) and Giant Magellan Telescope (Johns et al., 2012). One of the two first light instruments of ELT is the Multi-adaptive optics Imaging Camera for Deep Observations (MICADO) (Davies et al., 2010) which will get an advanced AO module with several modes of operation. There are numerous interesting science cases to be explored in more depth or for the first time with this new size class. With a 39 meter primary mirror and the AO to fully make use of its resolving power, the prominent science cases are those

[1] Lenses quickly become impractical above a certain size.

[2] Pun intended.

of high angular resolution. To give an idea of what is to come, I list a few examples from Fiorentino et al. (2017): research on (proto-)planets and sub-stellar objects, stellar kinematics in globular clusters, looking for intermediate mass black holes, the chemical composition of globular clusters in the local universe, calibration of cosmological distances, the assembly of high-redshift galaxies and searching for the first galaxies. A more detailed description of the telescope and MICADO can be found in section 3.2.

2.2 Globular Clusters

In this thesis, I aim to quantify some of the expectations of ELT in one specific research field. The focus is on globular star clusters and the associated crowded stellar fields. Globular clusters (GCs) are large collections of stars that are gravitationally bound to each other and born around the same time. They are amongst the oldest objects in the universe, making them interesting objects by themselves, but also as a gateway to learn more about the evolution of our galaxy and the age of the universe (C. Peterson, 1987). GCs have been subject to study for many decades, their first explicit categorisation possibly dating back to Shapley (1916). Early on, they were used as our best observational estimates of the age of the universe, as they were the oldest known objects.

More recently, GCs still play a role in determination of timescales. But after the accurate measurements of the cosmic microwave background radiation and the accompanying age determination, this role is somewhat more nuanced and focuses more on the formation and early evolution of galaxies (VandenBerg et al., 2013). These massive collections of stars are so much brighter together than an individual star can be that they can be studied at far greater distances, and can be cautiously used to infer things about their host galaxies (Larsen, 2013). With the advent of ever more powerful telescopes, the possibility of resolving single stars in distant GCs comes within arms reach. I will go into more depth on GCs in section 3.1

2.3 Goals

There are several questions that I will be trying to answer, the first and most important of which is how accurately we can perform point spread function (PSF) fitting photometry towards the crowded centres of GCs. After the stellar magnitudes are calibrated, it will also be possible to estimate a distance modulus, and with that the distance to the cluster can be checked. Additionally, I will determine how well the age of these clusters can be reproduced from the main-sequence turnoff point in a colour-magnitude diagram (CMD).

The way this will be done is by first simulating a cluster with known age and distance, then making a simulated image of that cluster and analysing the image using photometry that would also be used on real images. The resulting positions of the stars can be checked against the exact positions from the simulation. The same goes for the derived age and distance, using the input parameters as the reference point. In the ideal case, the two will match, meaning that the photometry perfectly reproduced the simulated cluster and thus closing the 'loop'[3]. This is not expected. However, the results can give a quantitative estimate of what can be achieved with the real data.

In order to get to photometry and measured parameters, the astronomical objects (in this case GCs) as well as the observing instruments (in this case MICADO @ ELT) will be simulated. For the second part there is a Python package in development called `SimCADO` (Leschinski et al., 2016) that is already capable of reproducing the effects of the optical train[4] of the telescope up to and including the imaging sensor itself. The current version handles imaging; simulating MICADO's ability to take spectra will be added in a later version, in the form of `SpecCADO` (for which I refer the reader to the documentation of `SimCADO`).

[3] A loop because we go from parameters to simulation to measurement back to parameters.

[4] Jargon for all the elements of the telescope and instrument that have an effect on the incoming light.

To make images with this software package, there has to be some input representing the astronomical object that one wants images of. This can be in the form of an existing image of a galaxy or nebula for example, or a simple, built-in (young) stellar cluster with a Gaussian density profile. The other method would be to define individual stars, one at a time or a whole set at once. This is the part that I will be writing code for from scratch. For the purpose of creating a wide range of possible star clusters, I will be using the option to define a whole set of stars in one go. In section 4.1 I describe how the astronomical object, for now consisting of only stars[5], is assembled.

MICADO works in the near infra-red, so I will be using astronomical filters in this region of the spectrum. A filter allows a small part of the spectrum of light to efficiently pass through it, and blocks out the rest. Knowing exactly what 'colour' of light we are looking at and combining measurements from different filters enables us to learn much more about the sources that emitted that light compared to not knowing what frequency of light is collected. The magnitude (or brightness) measurement of a star is always tied to a specific filter[6] that has been characterised in terms of the light it lets through. I will simulate images for a set of three broadband filters: J, H and Ks that have central frequencies of 1220, 1630 and 2201 nanometers respectively. More details on how the images are made and analysed and the data reduction are in sections 4.2, 4.3 and 4.4.

[5] Gas and dust would be a necessary addition to be able to reproduce i.e. spiral galaxies

[6] Except the *bolometric* magnitude, which is specifically defined as the brightness of a star across *all* of the spectrum.

3 Theory

3.1 Globular clusters

In the classical view, GCs are comprised of a simple stellar population of stars born at the same epoch. Other (open) clusters were distinct by being less massive, much younger and more metal rich. This view has changed drastically over time, away from the idea of a clear cut category that defines what a GC is (Larsen, 2011). Young star clusters with similar masses have been found to exist, eliminating mass as a separating feature. Also the age distribution of open clusters shows overlap, making a simple age cut-off impossible. The abundance of heavy elements in GCs is generally lower than the young clusters that formed in the metal-enriched gas clouds of galaxies. However, the GC population itself can be divided into a higher and lower metallicity subgroup (Zinn, 1985; Minniti, 1996). The metal-rich GCs found in other galaxies even reach solar-like metal abundances (Peng et al., 2006), eliminating another - relatively simple to obtain - observational quantity.

Perhaps the most important development in the understanding of GCs is that their stellar population is in fact not so simple; multiple populations of stars can be identified in them (Gratton, Carretta, and Bragaglia, 2012). This means their actual formation history cannot be explained by a conventional single period of stellar formation that would result in some spread in the perceived age of the cluster. Seemingly several distinct episodes of star formation are needed to reproduce the observations. These populations of stars do overlap considerably in the observed quantities making it very hard to tell them apart, explaining their relatively late discovery. A different scenario that could possibly explain this separation into distinct populations is if a super massive star ($\gtrsim 10^3 M_\odot$) formed during the formation of the cluster (Gieles et al., 2018). Such a star could enrich the surrounding protostars[7] with elements produced in hot-hydrogen burning, making them appear to be from a slightly different population than the stars already formed before the super massive star.

One remaining feature to mention is the distribution of stars in GCs. The stars are very tightly packed in the cluster centres, while not showing a very long tail of dispersed stars towards the outer edges. This is presumably due to the tidal stripping of the outside regions of the GC over time; other heavy objects like clusters or galaxies pull stars that are loosely bound away. This leaves the radial distribution of stars with a sharp cut-off. Evidence of this process is found in the form of tidal 'tails' in some clusters (i.e. Palomar 5 (Odenkirchen et al., 2001)): these are streams of stars that extend from the cluster roughly following the orbit of the cluster around its parent galaxy. King (1962) has modelled the radial distribution of GCs and found a typical cut-off radius of about 30 times the 'core radius', the scaling parameter representing the size of the cluster's core. This distribution is also used in the code written for this thesis.

It is clear then that there are still some mayor unknowns in the field of GCs, or clusters in general, their formation process being the biggest one. Being able to study them at further distances and in more detail with new telescopes like the ELT is crucial in further developing our knowledge[8] about their formation and evolution, and everything they can tell us about their surroundings. The fact that these clusters are so centrally condensed, together with their massive amount of stars, makes them visible at very long distances. Unfortunately there is a point where our current telescopes cannot distinguish individual stars in the cluster anymore, and it all becomes a blur. There is still science to be done with this light, however. As Shown by Larsen, Brodie, and Strader (2017), the integrated light of GCs can be used to great effect in obtaining accurate heavy element abundances at intergalactic distances ($\sim 4Mpc$). It is mentioned that photometry of single stars at similar distances, enabled by future 30–40 m telescopes, is the key to constraining the assembly- and chemical enrichment

[7] Very young star that is still gathering material from the gas cloud around it.

[8] The classic: 'more data is needed'. Something that can be said of any active field of research, perhaps.

histories of galaxies in a large enough volume of space that it can be representative of the whole universe.

3.2 The telescope

Not only in the context of GCs, but in general, photometry of individual stars at the furthest distance possible is a desired feat. Crowded stellar environments are an extreme case where we want to be able to perform photometry on single stars. On the one hand we might want to look as far away as we possibly can, meaning we look at very faint objects. On the other hand these objects are packed with stars, so it is very hard to tell them apart. This is where large telescopes with advanced AO systems really excel, due to their ability to both collect an enormous amount of light and to keep point sources separated down to very small angular separations. However, this literally comes at a cost: expensive telescopes make for expensive observing time. This is why simulating the performance of the instrument beforehand is such a good idea; we want to quantify what might be expected from observations in order to support the science cases that will make good use of the instrument's capabilities.

When completed, the ELT is an impressive feat of engineering: many aspects of it push beyond the capability of existing technology. The dome (the structure that houses the telescope) is an imposing building as well: with 86 meters in diameter (ESO, 2011) it is more than twice as large as the 39.3 meter primary mirror. Making large telescope mirrors to great precision is a time consuming process, taking on the order of several years for telescopes this size (Lewin, 2017; ESO, 2018). The ELT primary mirror is not made in one piece, but consists of 798 segments that are 1.4 meters wide each (ESO, website 2019). To support itself, a single mirror would have to be very thick, making it much too heavy[9] to be used in the moving structure of a telescope. Each of the individual segments only has to be 5 centimetres thick in order to be strong enough, making them weigh 165 kilograms on their own.

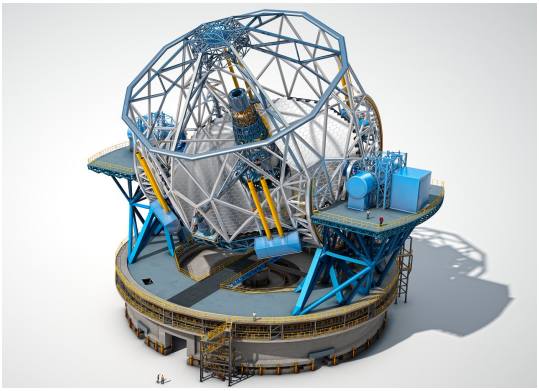

Figure 1: Preliminary design rendering of the ELT. Credit: ESO.

The secondary mirror in itself is larger than a good portion of optical telescopes built to date. With a diameter of about 4 meters it is comparable to the William Herschel Telescope and a bit less than double the diameter of main mirror of the Hubble Space Telescope. The big difference is that the secondary mirror of ELT is convex, not concave like the primary mirrors. The star light is reflected by a total of five mirrors, designated M1 to M5, before entering the instrument focus.

One of the instruments at first light will be MICADO, primarily made for high resolution imaging. The pixel scale can be set to 4 or 1.5 milli-arcseconds per pixel in the 'wide' field or the 'zoom' mode[10], making optimal use of the telescope's resolution potential. The diffraction limit of the telescope varies from a couple of milli-arcseconds in the optical to about 14 milli-arcseconds at the largest wavelengths of the near-infrared that can be imaged. The slightly smaller pixel scale ensures that the PSF is sampled by more than one pixel. This is necessary for the correct analysis of the images later on. The detector consists of 9 chips arranged in a

[9] And incredibly unwieldy!

[10] The names 'wide' and 'zoom' are used in `SimCADO` for these modes, at least.

square with small gaps in between them; the middle chip on one of the edges is slightly offset outward. The chips each have a resolution of 4k by 4k pixels adding up to a total of nearly 151 million pixels. The full field of view will be roughly 1 arcminute in the wide imaging mode: about thirty times smaller than the diameter of the moon as seen from earth.

Achieving diffraction limited imaging will be made possible by the advanced AO that MICADO can make use of. These systems are designed to mitigate the blurring effect of the atmosphere, which is time-dependent. MICADO includes an AO mode called single conjugate AO (SCAO). The MAORY (Multi-conjugate Adaptive Optics RelaY) module will provide two more modes of AO, namely a multi-conjugate (MCAO) and a laser tomography (LTAO) mode (Davies et al., 2010). The first two modes work by observing one or more natural guide stars (NGS) that have to be bright enough and in the proximity of the science target. The LTAO mode on the other hand is not dependent on the presence of a real star but instead creates its own stars in the upper atmosphere by shooting lasers upwards. The sodium lasers produce light at a wavelength of 589 nanometers that excites sodium atoms in the mesosphere that then starts radiating (Bonaccini Calia et al., 2010). The disadvantage is that the correction might not be as good as with a number of real stars.

As the name suggests, AO systems work by adapting some of the optical components in the telescope and/or the instrument. Depending on the implementation, there are a number of deformable mirrors that are controlled by a wave-front sensor that is looking at the incoming light. The SCAO mode can only make use of a single NGS, which in practice means the correction of the light is best only for a small region around the NGS. The shape of the PSF degrades further away from the position of the NGS, so that the PSF is not only time dependent but also spatially dependent. In MCAO (and LTAO alike), more guide stars are tracked at more positions on the field of view. This allows for the correction of the wave-front across a much larger portion of the image, making it less dependent on where the star is in the image.

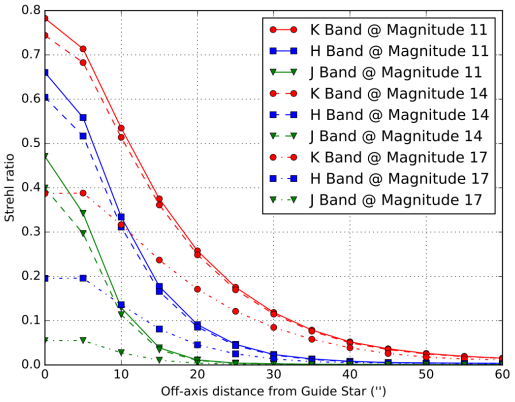

Figure 2: simulations of the Strehl ratio for ELT's SCAO mode as a function of angular distance from the guide star. (Vidal et al., 2018)

A common metric for determining the quality of corrections to the PSF is the Strehl ratio. Figure 2 by Vidal et al. (2018) is a graph of simulations of the Strehl ratio for ELT's SCAO mode as a function of angular distance from the guide star, in several filters and for different NGS brightnesses.

Performance of an instrument can be measured with the maximum magnitude that can be identified (at a certain signal to noise ratio) in a particular imaging filter. Figure 3 shows the expected performance of MICADO in the J, H and K filters as function of the exposure time (Davies et al., 2010).

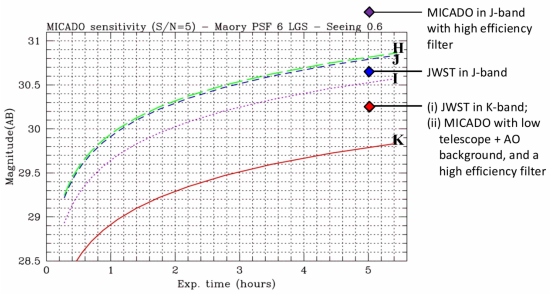

Figure 3: Expected performance of MICADO in terms of sensitivity. Some reference points are shown for the James Webb Space Telescope. (Davies et al., 2010)

4 Methods

4.1 StarPopSim: basis and functionality

As mentioned, I will be using the Python package `SimCADO` to simulate the optical train of ELT and its instrument MICADO, more on that can be found in section 4.2. To be able to make a `SimCADO` source object, the following parameters have to be passed on to it: x and y positions in arcseconds, apparent magnitudes, corresponding astronomical filter and spectral types. Generating these different inputs to represent real stellar clusters is discussed in the subsections below. The Python program that I have written for this purpose produces these quantities from a number of basic inputs like the total stellar mass in the desired cluster, its age and its metallicity. For a description of some of the basic functions see 4.1.5. It is not limited to one stellar population: one can specify multiple ages and/or metallicities to generate more populations of stars. In principle, elliptical galaxies can be produced by ramping up the number of stars and stretching one or more of the axes of the spatial distribution, with an additional rotation (inclination angle) if desired. Other galaxy types would be a fitting addition, but unfortunately that is outside of the scope of this project.

The result is a computer code equivalent of an astronomical object: the class `AstObject`, which combines all the relevant stellar properties into a coherent structure. This class is part of the publicly available code `StarPopSim` that can be downloaded from GitHub.

4.1.1 Generating masses

Arguably the most important characteristic that defines a star is its initial mass. The stellar masses, together with their spatial positions, will be the only parameters that are generated at random using Monte Carlo techniques. To generate stellar masses I use a simple version of the Initial Mass Function (IMF) by Kroupa (2001), cutting off the part below 0.08 solar masses.

The employed method is inverse sampling, which uses the probability density function (PDF) of the desired distribution and transforms it in such a way that the sampled points are themselves distributed according to the PDF. The recipe to do this is as follows. One starts by integrating the PDF over the relevant domain, in this case from 0.08 $M_\odot$ up to a variable M in solar masses, to get the cumulative distribution function (CDF). To normalise this function to one, it is divided by the definite integral (interval 0.08 $M_\odot$ up to an arbitrarily chosen 150 $M_\odot$). This CDF is then inverted to get the desired parameter (mass) as a function of the cumulative parameter (defined between zero and one by normalisation of the CDF).

The computer samples numbers from a uniform distribution between zero and one using a pseudo-random number generator. The package `NumPy` has some fast implementations, for example. These numbers are put into the inverted CDF, which results in a set of numbers for the mass of the stars. This set now has the wanted distribution, following the PDF that was started off with.

The lower mass cutoff was chosen for various reasons. One of which is that including masses below 0.08 $M_\odot$ will increase the complexity of the Kroupa IMF, since it has various different slopes depending on the mass. On the other hand, extremely low mass stars do not produce a lot of light, so they will hardly ever be picked up on an image, or contribute much to the total integrated luminosity. This partly justifies the hard cut. To add to that, by the nature of the IMF, there are a lot more low mass stars than high mass stars. For a simulation on a computer with finite resources, it is a good idea to leave out the bulk of stars that would not add anything substantial to the results. It is however good to be aware of this limitation as user of the software.

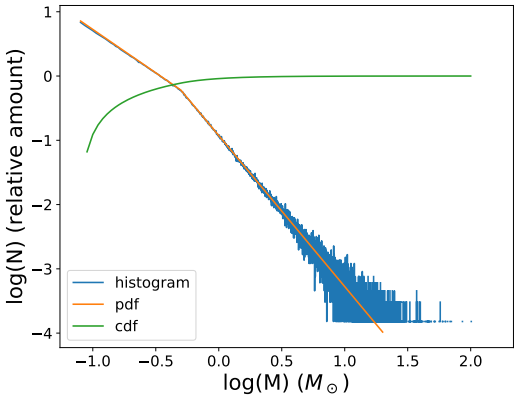

Figure 4: Plot of the histogram of masses for one million stars overlaid with the theoretical Kroupa IMF, as well as the CDF of that function.

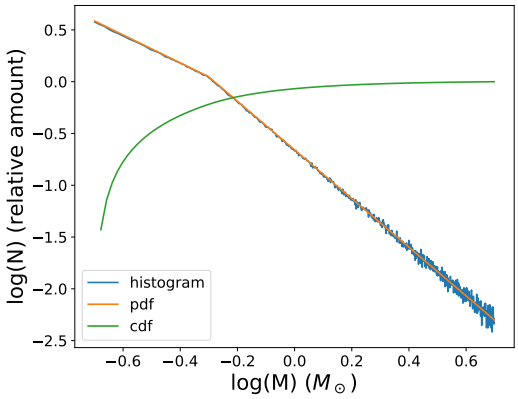

Figure 5: The same lines as plotted in 4, now for a smaller mass range of 0.2 to 5 $M_\odot$. The bend in the IMF occurs at a mass of 0.5 $M_\odot$.

In the end, the real limiting factor for the lower mass of the stars is in the isochrone files that will be discussed in 4.1.3. The program will take the lowest mass it finds in these data files as lower limit, as long as its above 0.08 $M_\odot$. The isochrone files that are used by default do not go below a mass of 0.1 $M_\odot$.

The upper mass limit is more arbitrary. There are natural limits to the size of a star, but the mathematical equation that is the IMF does not impose any limits by itself. The IMF reflects the fact that there are very few extremely high mass stars, but in principle there is no upper limit to what the code can generate. Since there is no hard theoretical limit known, the implemented default limit of 150 $M_\odot$ is a rough estimate of what is reasonable for a star. Both the upper and lower limit can be changed by the user, although the slope below 0.08 $M_\odot$ does not change as it does in the Kroupa IMF.

4.1.2 Radial distributions

Stellar clusters are of course three dimensional objects in space, so for simplicity they are assumed to be spherically symmetric[11]. What differentiates clusters spatially is their radial distribution. These have to be handled with care, since the three dimensional radial distribution is different from the corresponding 2D (projected) version. When we are looking at the stars, we see a projection from three to two dimensions on the sky. So observationally obtained radial distribution profiles of clusters are of the projected kind. Since some objects might require 3D rotation, as well as for general realism, I chose to go for the 3D distributions as opposed to flat stellar clusters.[12]

This means doing some inverse Abel Transforms (Abel, 1826) to go from observed profiles to the needed 3D probability density functions. The Abel transform and respectively the inverse of it are given by:

$$F(\rho) = 2\int_{\rho}^{\infty} \frac{f(r)\cdot r}{\sqrt{r^2-\rho^2}} dr \qquad (1) \qquad\qquad f(r) = \frac{-1}{\pi}\int_{r}^{\infty} \frac{dF}{d\rho}\frac{1}{\sqrt{\rho^2-r^2}} d\rho \qquad (2)$$

where ρ is the polar distance from the centre (so in 2D) and r is the spherical distance from the centre. The formula resulting from the inverse Abel transform is the PDF of the

[11] Except for the earlier mentioned possible stretching of certain axes, but that is a later step.

[12] Plus, it's only a small relative increase in the amount of data generated.

radial component of the spatial stellar distribution. However, this cannot be used to generate the radii of stars just yet.

Just as for the stellar masses, the method of inverse sampling is employed for the positional parameters. The integration is now not over mass, but over spatial coordinates. This means that for the radial distribution, the PDF times the radius squared has to be integrated from zero to r. The extra r^2 there comes from the Jacobian determinant for spherical coordinates; this would be a single r for the polar coordinate version. Unfortunately, the computed equations are not always invertible, or very ugly when inverted. Luckily there is an easy numerical solution that might even be faster in some cases. To invert an equation numerically, I calculate the outcome Y at say, a thousand points X. The uniformly generated numbers between zero and one are then given to the `numpy.interp()` function as the 'x' values, while our results (Y) are given as the second argument (called 'xp'), and the points (X) are given as the third argument of the function ('yp'). This interpolates the CDF between pre-calculated values and returns the corresponding value for the radius, thus effectively inverting the CDF. Note that this will only work correctly for single valued functions, in other words, the CDF has to be monotonically increasing or decreasing. This is safeguarded by the definition of a cumulative function.

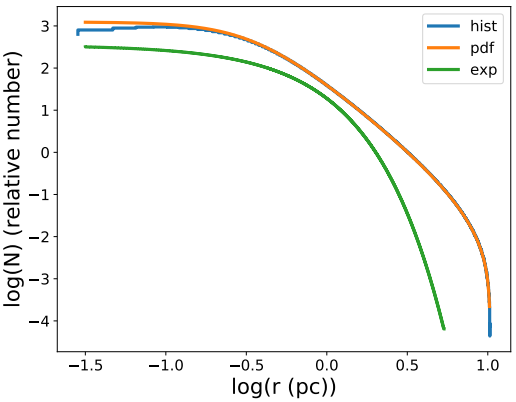

Figure 6: Histogram of the implemented King profile (blue) and the theoretical curve (orange), compared to an exponential profile (green). Here s=0.34 and R=10.3.

Figure 6 shows the general shape of the King profile (King, 1962), which has been modelled specifically after GCs. It has two distinctive bends in it: one around the core radius (s) and one at the edge that signifies the cut-off radius (R). Every radial distribution profile will have a scaling factor, analogous to s. For each profile, the ratio between the scaling parameter and the half-light radius (HLR) of the cluster will be different. For this particular case, the HLR is about 2.9 times larger than the core radius. The number density curve in the plot is for a half light radius of 1 *pc* and uses the typical value of R for GCs as found by King. This value is 30 times the core radius s; the ratio between R and s (so here: $\frac{R}{s} = 30$) is also called the concentration parameter. An exponential profile with the same scale factor is also plotted for comparison.

As an example I will demonstrate the mathematical steps with the King profile, which is the most important one for this research. The King profile in its original state (King, 1962) looks like this:

$$F(\rho) = C\left\{\left[1+\left(\frac{\rho}{s}\right)^2\right]^{-\frac{1}{2}} - \left[1+\left(\frac{R}{s}\right)^2\right]^{-\frac{1}{2}}\right\} \tag{3}$$

where C is a constant (to be determined by normalisation), s is a scale factor that can be interpreted as the core radius and R is the cut off radius where the density of stars drops to zero. After applying the inverse Abel transform, this profile has the form:

$$
\begin{aligned}
f(r) &= C\left\{\frac{1}{2s}\left[1+\left(\frac{r}{s}\right)^2\right]^{-\frac{3}{2}} - \frac{2}{\pi s}C_2\left[1+\left(\frac{r}{s}\right)^2\right]^{-1} + \frac{4-\pi}{2\pi s}C_2^3\right\} \qquad (4)\\
C_2 &= \left[1+\left(\frac{R}{s}\right)^2\right]^{-\frac{1}{2}}
\end{aligned}
$$

where C_2 is introduced to shorten the equation and the last term is an added constant to ensure that the profile is still zero at the outer radius R. The next step is to integrate the function over spherical radius to get the CDF. The result of which is:

$$
\begin{aligned}
N_{CDF}(r) &= C\left\{\frac{1}{2}\left(arcsinh\left(\frac{r}{s}\right) - \frac{r}{s}\left[1+\left(\frac{\rho}{s}\right)^2\right]^{-\frac{1}{2}}\right)\right.\\
&\left. -\frac{2C_2}{\pi}\left(\frac{r}{s} - arctan\left(\frac{r}{s}\right)\right) + \frac{4-\pi}{6\pi}C_2^3\left(\frac{r}{s}\right)^3\right\} \qquad (5)
\end{aligned}
$$

The normalisation constant is now easily obtained by setting $N_{CDF}(R)$ equal to one and solving for C. At this point the CDF must be inverted; you might be able to see that algebraically this is impossible. So instead this is done with numerical method as explained earlier. Since the radial profiles fall off steeply, less reference points are needed at larger radii, so the radii for the interpolation grid are made with logarithmic spacing.

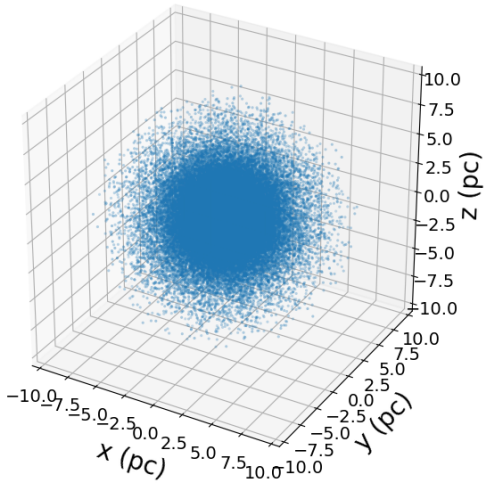

Figure 7: The 3D scatter plot of cluster 1

The distribution for the angular coordinate ϕ is uniform between zero and 2π. Theta has to be distributed as a sine between zero and π to get a uniform distribution on the surface of a sphere. That means the inverted CDF is $arccos(2N_{CDF}-1)$ with N_{CDF} sampled uniformly between zero and one. All spherical coordinates combined, the stars can finally be given unique positions in space to form the cluster they are part of. Figure 7 is the 3D scatter plot of cluster 1: this looks a lot more crowded than it will end up being on the image, where most faint stars will not be visible.

4.1.3 Stellar isochrones

For most of the other stellar properties, I use the MESA Isochrones & Stellar Tracks (MIST) (Dotter, 2016; Choi et al., 2016; Paxton et al., 2011; Paxton et al., 2013; Paxton et al., 2015) and interpolate the data for each star using their randomly sampled initial mass. Two other parameters are needed to be able to do this: the age and the metallicity of the population of stars in question. These parameters are to be specified by the user when starting a simulation. There is a wide range of available metallicities from Z = 0.045 down to 1.4e-6. The age of the generated stars can range from 0.1 Myr all the way up to 20 Gyr.

The isochrone data consists of a number of stars (or data points) at a large range of initial masses for each available time step. Each one of those data points has certain physical

properties associated with it, like the current mass of the star and its luminosity. Making a plot of one such property at one fixed time step would then result in a line where stars of all starting masses, with that specific age, theoretically lie upon. This is what is called an isochrone[13], which is used to define a single population of stars (meaning they are born around the same time). Stars that are grouped in a cluster are often member of one or just a couple of populations (Li, Grijs, and Deng, 2016).

Using the initial masses generated as explained above, the physical properties in the isochrone files can be extracted by interpolation and then assigned to the stars. Several of these properties will be needed to make the wanted mock images (done by `SimCADO`). Most notably we need to know how bright the stars are in different parts of the electromagnetic spectrum. Fortunately, the MIST files also provide magnitudes in various filter bands. The selected isochrone files include some of the broadband filters that MICADO will most likely have: the I, J, H Ks band.

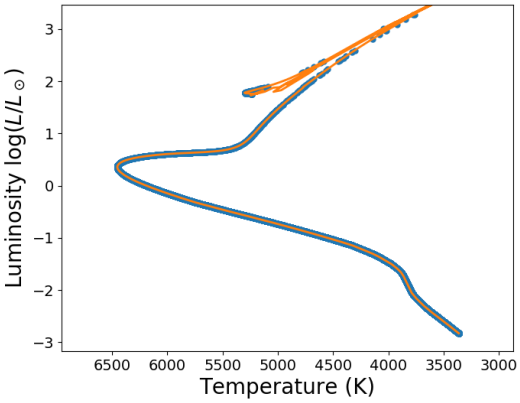

Figure 8: Herzprung-Russel diagram of cluster 1 (blue) with the corresponding isochrone in orange.

4.1.4 Spectra and remnants

One of the parameters in `SimCADO` is the spectral classification of each star. This is an optional parameter, but it gives the possibility of doing spectroscopy as well as using a photometric filter that is not available in the MIST files. Plus, it would be unrealistic to have all stars be of the default spectral type A0V. Depending on the way images are made from the source object, this could either drastically affect the end results or have no influence at all. Say for example we make one `SimCADO` source object using the stellar magnitudes from the J filter and we do not specify individual spectral types. If images are made in the J, H and Ks filters using this one source object, `SimCADO` will in theory calculate what the magnitudes of the stars should be in the other two filters (H and Ks) using the model spectrum for the default spectral type. The result is that for all stars, the magnitudes go through the same transformation to go from the J band to the H and Ks bands. This means the colours of the stars, defined as the difference between two of their magnitudes, is the same for all of them!

The other way of doing this actually makes specifying spectral types unnecessary, in the case that enough information is available. If for each image in a different filter, the source object is updated to one that uses the same new filter for the magnitudes, the problem does not occur and we can have stars of different colours. There is a possibility, however, that the wanted imaging filter is not an available magnitude for the simulated stars. In that case it is crucial to have the right spectral type specified for each star.

In `SimCADO`, there exists a table listing (almost) all spectral classifications accompanied with some of the defining physical properties. I have used this table in reverse to determine the spectral type of the generated stars[14]. The more exotic spectral types are disregarded for this process (such as Wolf-Rayet stars and sub-dwarfs), since they have properties overlapping with those of other types. It *does* still contain white dwarfs; more on these and other remnants below.

[13] Derived from the Greek *isos* - 'equal' and *khronos* - 'time'.

[14] A better integration between `StarPopSim` and `SimCADO` would skip this step and feed the physical properties directly to the imaging routines.

A grid in the form of a k-dimensional 'lookup' tree (or k-d tree for short) is made of three physical quantities: effective temperature, luminosity and current mass. These three are sufficient to make sure that there are no degeneracies between the remaining spectral types, and that they are far enough apart that a good estimate can be made from a continuous set of variables. The k-d tree is a highly efficient computational method for obtaining the closest point in a grid (the spectral types) and a distance to it, for each entry in a large set of data points (the stars). When queried it will spit out indices that correspond to the array of spectral types that has gone into the tree. Since it is much more efficient to store one small number for each star than it is to store three or more characters of at least a byte each for every star, the indices will be kept and used alongside the list of spectral type names[15].

The MIST isochrones include fairly far evolved stars. White dwarfs are represented, for instance. However, they are the only stellar remnants from the list: white dwarfs, neutron stars and black holes. For this reason, `StarPopSim` has some functions that give estimates of the properties of neutron stars and black holes using formulas from Cummings et al. (2018), Fryer et al. (2012), and Althaus et al. (2010). The inclusion of these estimates might, if not much, take some additional computation time. Therefore this is optional and can be controlled in the relevant functions with the parameter 'realistic_remnants', to be set to True or False. Keep in mind that when these are not used, remnants (excluding white dwarfs) will have wildly wrong properties assigned to them, like a mass of zero.

For photometry and spectrometry, black holes and neutron stars are only of visible relevance due to the direct surroundings of the remnant, like ejecta or accretion disks. These are unfortunately not modelled here (yet), so disregarding them at this stage is not a big problem. They are also inherently not very numerous, since their progenitors are present in much lower numbers than lower mass stars that would form white dwarfs instead. The white dwarfs, as discussed above, are properly modelled due to their presence in the MIST files.

White dwarfs have their own spectral classification, starting with a capital 'D' followed by another letter indicating the presence of some strong spectral features. Since the classification based on the three mentioned parameters does not relieve the degeneracy between them, only one class of white dwarfs is used. All the white dwarfs are put in the class denoted with a 'C' (so DC+number gives the full name) which denotes no strong spectral lines.

4.1.5 Basic functions and input

`StarPopSim` in its current form is focused heavily on simulating GCs, although some functionality to create other types of objects like open clusters or elliptical galaxies does exist. To make a cluster of stars, the following functions can be used. Also listed are some functions that facilitate needs like making visual representations of the astronomical object that is created. All of the arguments and keywords are specified in appendix A, for the functions below plus the rest of the functionality for the `AstObject` and some more utility functions.

`objectgenerator.AstObject()`

This is the class that holds all of the information on the created stars; it returns an `AstObject` object that will be stored under the name 'astobj' for demonstrational purposes. This class is initiated with all of the parameters that define what the assembly of stars will look like. A total mass (in $M_\odot$) or total number of stars must be specified, as well as a list of ages (in years) and metallicities. If the age is a number below 12, it is taken to be a base ten logarithm. If either multiple ages or metallicities (or both, in which case they need to have the same length) are specified, a set of stellar populations with the given parameters are created. An optional keyword argument specifies the ratio of stars between the multiple populations, for instance `[1, 2]` would put twice as many stars in the second stellar popula-

[15] This might not seem like much, but when dealing with tens of millions of stars, things add up quickly.

tion. The final required argument is the distance to the astronomical object in parsecs. Some optional keywords are the type of radial distribution, the radial distribution scale parameter and a compact mode that is useful when creating more than ten million stars. The compact mode will limit the amount of stars generated by generating less low mass stars that would not contribute much to the overall light output, while being very numerous.

`astobj.coords`
The three dimensional positions of all stars. The coordinates are stored in a 'per star' format, so getting all of the x-coordinates is done through `astobj.coords[:, 0]`.

`astobj.ApparentMagnitudes()` The apparent magnitudes are what goes into the simulation of the optical train of the telescope. An astronomical filter can be specified; default behaviour is to return magnitudes for all of the available filters.

`astobj.SpectralTypes()`
This function somewhat roughly classifies all the stars into the best fitting spectral classes. The returned arrays consist of one array of reference numbers pointing at positions in the second array containing the names for the spectral types.

`visualizer.Scatter2D()`
This function makes a plot of the stars in a two dimensional plane that can be set to the x-y, y-z or z-x plane. The only required input is the array of coordinates directly form 'astobj'. However, the effective temperatures and/or magnitudes in a chosen filter can be given as input as well, enabling the option to colour the stars according to how hot they are and scaling the size of the markers proportionally to their brightness.

The other functions in the visualiser module are a three dimensional plotter, an HR diagram and a CMD and a histogram of any distributions of interest (for instance the radial distribution).

`imagegenerator.MakeSource()`
This function makes it easy to make a `SimCADO` source object from an existing `StarPopSim` astronomical object. The only two inputs are the 'astobj' and the filter to observe it with. This step, plus the image making process, could in principle be exchanged for another telescope's optical train simulator.

`imagegenerator.MakeImage()`
Acts as a simple wrapper for the `SimCADO` function `simcado.run()`. The exposure time, telescope field of view, CCD chip layout, AO mode, again a filter and a file name are taken as input. Much of the finer details of `SimCADO` are lost, so if this is desired I encourage the reader to investigate the options. The filter specified here will be the actual imaging filter, but where possible it is best to keep this the same as the filter specified in `MakeSource`.

Additionally, there is a way to make astronomical objects and images of them via the command line. To do this, call the files 'constructor.py' or 'imager.py', respectively, with the appropriate keyword arguments specified. These may differ from the keywords used in the functions above. For some help in exploring the different options there is an interactive mode to both of these modules, called with the keyword '-inter'. This feature is a good starting point to get familiar with `StarPopSim`.

4.1.6 The simulated data

When I was satisfied with the state of `StarPopSim` and confident that it worked well for the use case of this thesis[16], I started simulations of the astronomical objects. To convince myself that the code was working properly I plotted histograms of the generated stellar properties overlaid with their theoretical distribution curves, and repeated this for many combinations of input parameters. Some of these histograms are shown in figures 4, 5 and 6. Other tests include looking directly at the three dimensional plot of the positions (like in figure 7) and checking that the other stellar properties followed the expected theoretical models (for instance the Hertzsprung-Russel diagram (HRD) in figure 8).

The main question to be answered has to do with how compact the GCs are, or better, how crowded they are perceived to be. There are three parameters that have a major influence on this: the distance to the cluster, the size of the cluster and the amount of stars in the cluster. The first two parameters determine the angular size on the sky, and the last one (quantified by the cluster's mass) determines how many stars are within that angular size.

My supervisor had a fairly good idea of the parameter space we needed to explore: distances between 800 kpc and 15 Mpc, cluster masses between 10^5 and 10^7 $M_\odot$ and half-light radii between 1 and 20 pc. I divided this up into a manageable total of 150 points, resulting in a three dimensional grid of points. The age of the clusters was chosen to be 10 Gyr, typical for a globular, and the metallicity was set at Z=0.0014, roughly one tenth of the solar value. For their radial distribution of stars they all share the King profile, as described in 4.1.2. The input scaling parameter is the core radius, which differs from the HLR. For a constant concentration parameter, there is a constant conversion factor between this core radius and the HLR. The concentration parameter is set to 30 for all clusters, resulting in a core radius about 2.9 times smaller than the HLR. So dividing the wanted half-light radii by this number gave me the input scaling parameter for the radial profile. The cluster masses are converted internally in `StarPopSim` to a number of stars to generate. This is an estimate based on the IMF, so the true generated mass of the cluster will differ slightly from the input. The input properties of all the individual clusters can be found in appendix B.

The clusters are then imaged using `SimCADO` with three photometric filters each: the J, H and Ks bands. These span a wide range of wavelengths in the near-infrared part of the spectrum. More on the settings used for the imaging as well as about `SimCADO` in section 4.2.

4.2 SimCADO

`SimCADO` is the simulation package being developed for ELT and MICADO, capable of producing images and later also spectra that look as close to the real science data as possible. To make the images realistic, many components and effects have to be taken into account.

4.2.1 Overview of the simulation

Figure 9 shows a schematic representation of the data flow in `SimCADO`. It starts with a `simcado.Source()` object (A) containing two positional coordinates (in arcsec) per source, magnitudes (represented as weights) and a reference to the spectrum of each star (an integer indicating which of the model spectra that star is linked to). Only a list of the unique spectra in the source are actually saved this way, reducing the amount of data. The step at the first arrow is to apply all effects that only affect the wavelength domain: these include the photometric filters and the telescope mirrors. At (B) the light from the source is already more representative of the number of photons that will be counted in the detector, except of course that the rest of the optical train still has to be applied. The next arrow denotes the

[16] I have ambitions for further functionality, but that will have to wait.

application of effects that simultaneously need to take into account the spatial and spectral information. These include atmospheric dispersion and the convolution with the PSF kernel (in simplified terms this means the PSF is applied to each star). From (C) to (D) the images at all the different wavelengths are combined into one monochrome image. At the arrow from (D) the spatial effects are applied, like the telescope jitter (small movements side to side) and field rotation. Rotation of the image during exposure is counteracted by the derotator, but in the case this does not work properly it can be simulated.

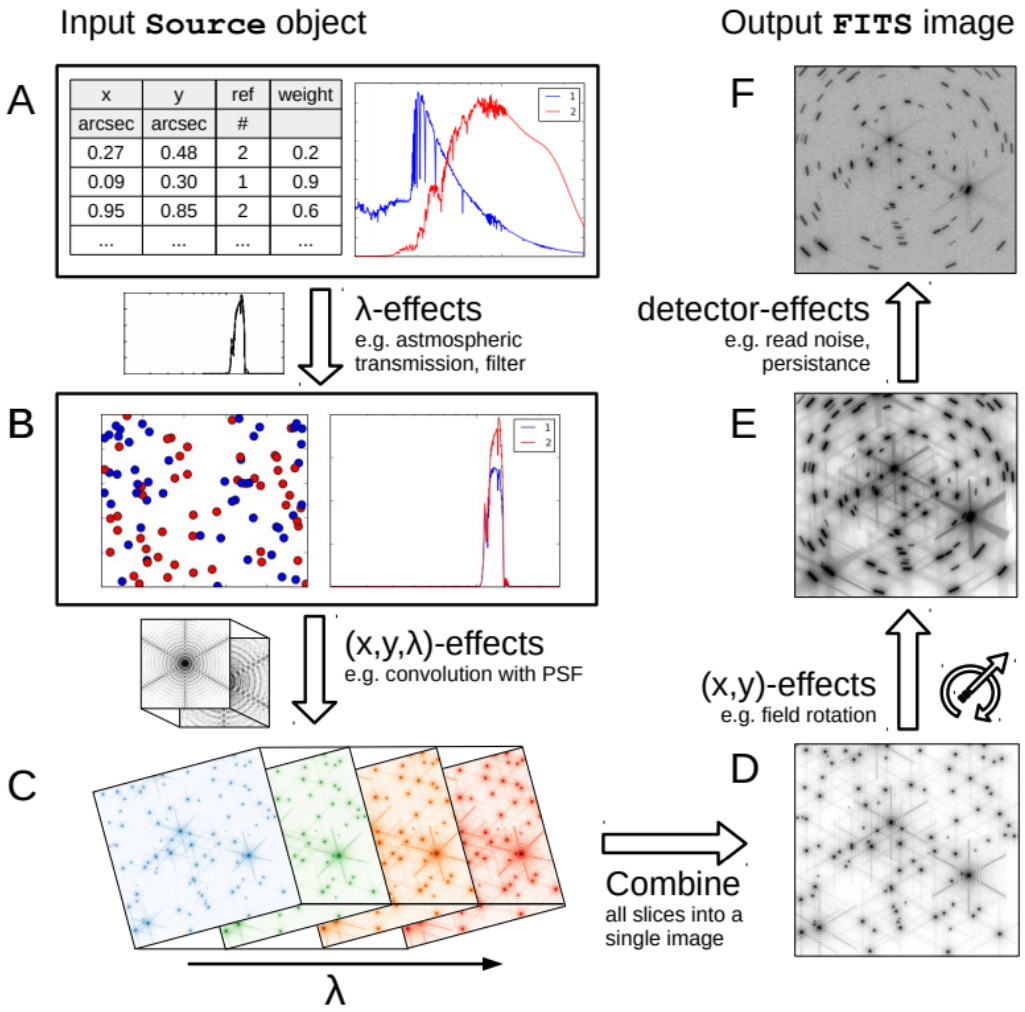

Figure 9: Simplified representation of what happens to the source when put through the optical train simulation of `SimCADO`. Made by Leschinski et al. (2016) and explained further in the text.

At the next point (E), the background flux is added. This consists of the atmosphere emission (certain atomic/molecular transitions produce photons in the optical or infrared) and thermal emission of the mirrors (close to negligible for all but the K band). Finally, at the arrow going to (F), the light virtually falls onto the detector. The detector itself has additional effects, including noise: correlated as well as uncorrelated white and pink[17] noise

[17] The name pink noise originates from the fact that the frequency distribution of this type of noise includes more low frequencies (pink contains more red which is of a low frequency in the visible spectrum).

are added here. There is also Poisson noise associated with the random nature of recording discrete numbers of photons, which is taken into account for both the source photons and the atmospheric emission. Several more sources of noise arise from the detailed workings of the detector, for instance some electrons might leak from one pixel to the next.

For a more detailed description of `SimCADO` and its functions, I refer to the mostly up-to-date online documentation[18] as well as the accompanying paper by Leschinski et al. (2016).

4.2.2 Adaptive optics

The atmosphere has the largest effect on the final image and influences both the spatial and the spectral domain. The spatial distortion of the atmosphere is largely countered by the AO systems; what remains of the distortion is combined with the PSF. These PSFs for the different AO modes are simulated separately by the teams working on these instruments (Vidal et al., 2018; Arcidiacono et al., 2014). `SimCADO` then applies these PSFs and only needs to add the wavelength dependent effects of the atmosphere.

As eluded to in the introduction, the PSF is different for each wavelength band we look at due to the way light is diffracted. The fact that we need to use AO to get close to the diffraction limit, means that a spatial variability will be introduced. Certainly in the case of SCAO, where only one reference star is used (see 3.2), the appearance of the PSF depends strongly on position. This effect is small for the central intensity peak, so for faint stars it is not very noticeable. In bright stars, where the fainter features of the PSF are visible, one can see an elongation in the radial direction centred on the reference star[19]. In `SimCADO`, the field variations are modeled discretely for nine different regions in the image. This means there is no continuous change in the shape of the PSF, but it goes in steps.

This field-varying PSF is a later addition to `SimCADO` (with version 0.6). At that point, the images for this project had already been simulated. Two factors played a role in the decision not to redo all the images with the field-varying models. These new models include nine different PSFs, which all need to be convolved with the image separately, effectively increasing the simulation time nine-fold. Secondly, analysing the images with a varying PSF might come with problems that make it a much more involved process. Both of these factors unfortunately meant that this was not a feasible option within this project.

4.2.3 Using SimCADO

`SimCADO` has many user commands, of which the two most important functions here are `simcado.source.stars()` and `simcado.run()`. The former will create and return a `SimCADO` source object for the given array of stars and the latter puts the source object through the simulation to then return a 'fits' format image.

Version 0.6 of `SimCADO` is used with the following parameters for simulating the optical train. The exposure time was set to half an hour, making one exposure per filter per cluster. The wide telescope field of view with 4 milli-arcseconds per pixel was used and only the centre chip of the detector is read out. The pixel scale is chosen to favour nearby clusters that cover a large area on the sky and kept consistent throughout the process; the far away, compact clusters might benefit from the smaller pixel scale. The detector read out takes a substantial time of the overall simulation, so only the central chip is used to save time there, but also in processing of the data. The nine detector chips each produce separate images that have to be analysed separately as well, increasing that process in duration proportionally. The AO mode is set to SCAO: this will probably be the only mode available directly at first light. The total of 450 images then have to be analysed to squeeze all of the information back out of them; this is discussed in section 4.3.

[18] Click on 'documentation' or manually go to the page: simcado.readthedocs.io

[19] See anisocado.readthedocs.io for the field-varying PSF models.

Additionally, three images are made that will serve to produce PSF models in each filter. In real observations, this is usually done with the science image itself; a few stars that are isolated enough from the rest are selected in each image to make the model. For consistency across the analysis, I opted not to do this, and instead made a separate image per filter with 36 isolated stars. These images are made with all the same settings as the cluster images, and in these simulations there is conveniently no variation in observing conditions between the images. This makes it possible to use the stars in these 'PSF images' to make the PSF model for each filter, that is subsequently used for all cluster images. This also takes away the need for the very involved process of selecting PSF stars in all of the 450 frames.

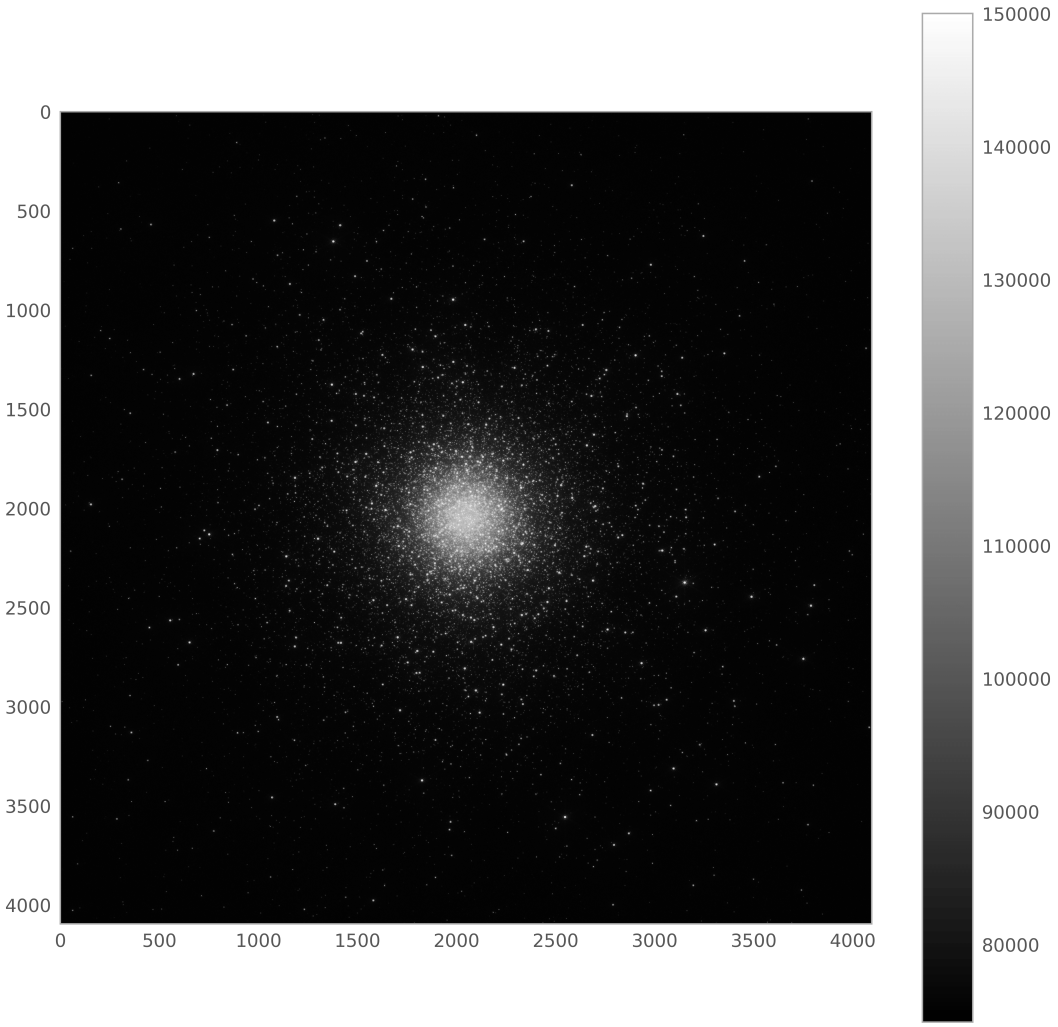

Figure 10: Example output of `SimCADO` for cluster 127 in the Ks filter.

4.3 Photometry

For the analysis of the produced images, the technique used is point spread function fitting photometry, or PSF photometry for short. In simple terms, a PSF is the image that is perceived when looking at a point source: a light source that cannot be resolved. Contrary to aperture photometry, the flux of stars is not measured by putting a (usually) circular aperture over the stars and counting the number of received photons inside. Aperture photometry does not work when stars are very close together or even partially overlap in what is appropriately

called a crowded field. GCs are typical examples of very crowded fields, up to the point where no individual stars can be recognised within their centres.

In PSF photometry, a model of the stars in the image is built by picking several isolated stars that have no contamination in their PSF of other nearby stars. This model is fitted to all the individual stars that can be distinguished. In principle, counting all the photons that contribute to the best fit of the PSF model belong to that single star only. The fitted PSF models are subsequently subtracted from the image to produce the residuals. In this residual image more stars can often still be found. The advantage is that the star-finding algorithm can now actually distinguish these leftover stars. The same process is repeated until there are no clear star candidates left.

To perform the described procedure, the intention was to use a Python package called `photutils` (Bradley et al., 2019), which is affiliated with `astropy` (Astropy Collaboration, Robitaille, et al., 2013; Astropy Collaboration, Price-Whelan, et al., 2018). This package has a lot of functionality and can in principle perform PSF photometry in crowded fields. There is even an implementation of the same algorithms that `DAOPhot` by Stetson (1987) uses for this. `DAOPhot` is a piece of code written in Fortran that has seen a lot of use in scientific publications, and for good reason: it is good at what it does. See for example Becker et al. (2007) for a comparison between various photometry codes.

The reason for not using `photutils` after all is that during the image reduction process, the system memory usage became too high. This could be prevented by setting a certain parameter to a lower value. However, the image reduction took much longer than the alternative that I ended up using. It is not clear to me how this performance will change in the future; hopefully it will improve significantly, as these packages are still under ongoing development[20]. Unfortunately, any updates to these Python packages come too late for the purpose of this thesis.

`DAOPhot` is integrated into `IRAF` (Tody, 1986), a broader software package for analysis and reduction of astronomical images[21]. Since an installation is available to me and I have at least basic knowledge of how to operate it, `IRAF` became the software of choice for analysing the simulated images. Furthermore, my supervisor is knowledgeable about both `IRAF` and `DAOPhot`, and pointed at their status as tried and trusted.

4.3.1 Use of IRAF

`IRAF` has many tasks, most of which I did not need, so they are not mentioned here. The installing procedure is well documented across the web, so I will also not repeat those steps here. Described below are the tasks I used in the reduction of the simulated data, as well as their parameter settings. Some very helpful Linux commands are mentioned as well, for completeness as well as convenience were this to be used as a guide to go through the same process.

First of all we want to create a model of the PSF that the `allstar` task can use to fit to the images. As discussed in 4.1.6, three separate images are used here. As turns out from trial and error, the PSF stars have to be quite faint to get a good model and subsequently a good subtraction of stars. This is likely due to the PSF of bright stars flattening off when getting near the saturation value of the CCD[22]. The magnitudes that I ended up using where between 22.6 and 23.2. Since fairly many (36) PSF stars were generated, the faint outer structure of the PSF could still be modelled reasonably well. The faintest features were disregarded, however, as they never rise very far above noise level, and a smaller PSF model

[20] With the latest version of `photutils` being numbered v0.6, it is still very much a WIP, warning the user that "The PSF photometry API is currently considered experimental".

[21] It very creatively stands for: 'Image Reduction and Analysis Facility'.

[22] Charge-Coupled Device, or the light collecting camera chip in more everyday terms.

(encompassing less of the outer structure) worked better.

The parameter values that worked well for me in this particular case are found in appendix C. If different data is used, many of the values will have to be tweaked for optimal performance. Here I will go over the steps taken to perform the photometry. The command `epar <taskname>` is used to change parameters for tasks or in parameter files. When editing is done, `:wq` is typed to save and quit from the editor. One can also jump to another parameter file by typing `:e` in the desired line. For a detailed description of a task and its parameters, the command `help <taskname>` is used.

The PSF models are made by running the tasks `daofind`, `phot`, `pstselect` and finally `psf`. The output model file is used as input for the `allstar` task later. First, one very useful Linux command for the following part is to make a file containing all the image names, or better, part of the image names to be analysed. This is done with: `ls *.fits | sed s,A,B, > grid.txt` where the middle bit replaces string A with string B. String A can also contain wildcards, but instead of '?' it uses '.' and instead of '*' it needs '.*'. If this is done in a convenient way, only three of these lists are needed (instead of three per filter). `IRAF` can add strings to the end of each entry in these files, but the text will be added *in front* of the file extension. This meant it worked best for me to make one file containing a list of all my images with a `.fits` extension and another with `.dat` for non-images. The third file is just a list that repeats the filename of the PSF model as many times as there are images for a filter. Things will not work when mixing in-/output file lengths. The lists are specified in the parameter files with an 'at': `@grid.txt//-H` where the double forward slash tells the program what to append to the items in the grid file. The H can denote the filter, so that one list can be used for all filters. More text can be added for the various output files; short and descriptive works best here.

The image reduction requires repeated use of the tasks `daofind`, `phot` and `allstar`. Each iteration takes the output from the previous one as input and will remove more stars from the images until no more are found or the iterations are stopped. Another reason for doing at least a second iteration is to obtain a better PSF model. If the stars in the science image that are used for the PSF model still have some neighbouring stars, that will contaminate the PSF model. A first iteration, subtracting many stars from the image, could potentially remove these neighbours. A new model can then be constructed with less influence from nearby stars, improving the fit for the next iterations.

Unfortunately there is a small catch: when no stars are found in an iteration, `daofind` will produce an empty list with which the `phot` task will not produce any output file. This then leads `allstar` to produce a fatal error, interrupting the routine. The only way to circumvent this is to manually find which photometry files are missing and deleting those from the list of images. This requires multiple lists to be made, as it differs per filter whether stars are still found.

Testing the photometry on one of the simulated images reveals that all stars that could be subtracted, where subtracted after 3 iterations. This may differ per image, but generally two or three runs should extract most or all of the stars. Therefore the image reduction is terminated after three repetitions of the three mentioned tasks above, plus one run of the star finding algorithm. That last step is to identify stars that would still have been subtracted out, stars that are not found anymore and artefacts that are mistaken for stars.

4.3.2 The obtained data

After the image reduction is done, a large collection of files has appeared alongside the original data. The most interesting one to us is produced by `allstar`: the data file containing the magnitude estimations of our stars. The subtracted images are also of interest, but mainly for checking how well the subtraction has worked. They also make for a satisfying representation

of the results. The magnitudes found in this data are instrumental magnitudes; this means that they don't represent physical brightness (yet), but rather the brightness relative to an arbitrary zero-point. They are derived from the measured stellar fluxes internally, with this formula:

$$m_{inst} = -2.5 \cdot log_{10}\left(F_*\right) + m_0 \tag{6}$$

where m_{inst} is the instrument magnitude and F_* is the measured flux for each star. A zero-point (m_0) can be added; in theory this is to immediately convert to apparent magnitudes, but this will be done in a calibration step discussed in the next section. For now, this zero-point is a free choice that does not matter for the later calibration of the instrumental magnitudes.

Figure 11: Central cut out of cluster 134, before any stars are subtracted.

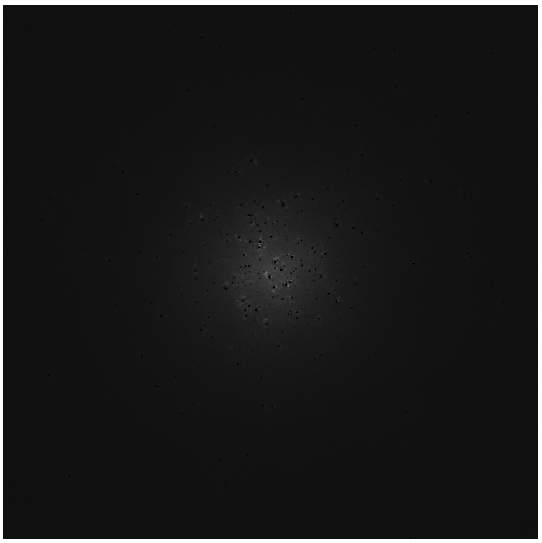

Figure 12: Central part of cluster 134 after it has gone through 3 subtractions.

Other notable variables listed in the `allstar` data files are the x and y positions for the best fit of the PSF model, an uncertainty measure for the magnitudes, the measured sharpness of the source and a χ-squared statistic. It is desirable to get χ^2 values close to one, indicating that the fit is a good match. In the test image, this statistic was closer to a half than to one. This turned out to be caused by a *too high* default value for the flat-field error. Since I am not making use of flat-field images, it seemed appropriate to set this value to zero. Indeed the χ^2 values became close to one after this change.

Figures 11 and 12 contain the same cluster before and after subtraction. In this case the subtraction was very good, showing no leftover bright peaks. Only a fuzzy glow can be seen in the centre of the cluster, where the many bright stars might have left some residuals. Also the high number of faint stars in the centre might contribute to this dim glow. A cluster where the subtraction was less good, especially for the bright stars, is displayed in figure 13. These bright stars might not be recognised by the finding algorithm, due to the very high amplitude secondary peak in their PSF (ring around central peak). This ring blends in with the rest of the star, making it look too wide to be identified as star. It might also have to do with the near-saturation values of the central peak of the star. Another source of residual light is the leftover glow from not-optimally subtracted bright stars.

Figure 13: Cluster 129 after the subtraction of the fitted stars. Some prominent stars are still left over, partially subtracted or not subtracted at all.

4.4 Further analysis

The photometry will result in a number of quantities for the stars as outlined in the previous section. As mentioned above, the most interesting of these are the positions of the stars and their instrument magnitudes; these are the direct measurements from the images that we want to examine further. In this section I explain how these measurements are calibrated and used to estimate the distance to the cluster as well as the age of the population of stars.

4.4.1 Coordinate systems

The position of the stars on the detector is measured in pixels, starting in the top left at (0, 0)[23]. We want to compare these positions to the 'real' ones that were given to `SimCADO` for simulation of the images. The original coordinates of the stars in `StarPopSim` were measured in parsecs, which gets converted to arcseconds before they are passed on to the telescope

[23]This is a remnant of old TV screen technology that started drawing the image at the top left. Although some programs will automatically place the origin in the bottom left corner.

simulation. Contrary to the final image, the origin of the coordinate system is in the centre of the to-be-imaged area.

In the chosen imaging mode, every pixel on the detector collects light from 4 milli-arcseconds on the sky. This means multiplying by $4 \cdot 10^{-3}$ transforms the pixel coordinates back to arcseconds. The single chip from the detector that is simulated has a '4k' resolution in both directions.[24] Translation of the origin is achieved by subtracting half of the detector width in pixels from the pixel coordinates of the stars, setting the new origin at the location of (2048, 2048) with respect to the old origin. However, based on comparison of positions with respect to the imaged stars, this is a bit too simple.

There are two more subtle effects that need to be taken into account. The first concerns the difference between list indexing in the Fortran coding language and Python: the former starts counting at 1 and the latter starts counting at 0. This means the results from the IRAF photometry have pixel counts that are 1 more than the pixel counts from the stars in the images when viewed in python; this is easily fixed by subtracting one from these IRAF coordinates. The second effect is within `SimCADO`: an object positioned at (0, 0) will show up peaking in brightness at pixel (2049, 2049) when imaged (figure 14). Why exactly this is the case I do not know; it would be more logical if the position would be at either pixel 2048 or 2047, since either of these could be defined as the 'middle' of 4096[25]. It turns out that this also slightly differs per filter. I measured the offset for the filters J, H and Ks to be 0.6, 1.4 and 1.0 pixels, respectively. The resulting formula for converting from coordinates in pixels (corrected for the 1-indexed IRAF values) to coordinates in arcseconds is:

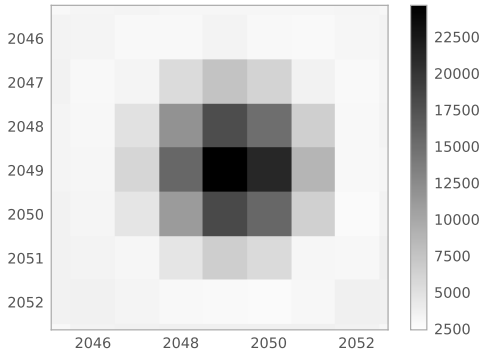

Figure 14: A single star in the centre of the image in the Ks filter (zoomed to pixel level).

$$x_{as} = 4 \cdot 10^{-3} \cdot (x_{pix} - 2048 - C_{offset}) \tag{7}$$

Calibrating the stellar coordinates will allow for the direct comparison of the stars from the cluster in code form and those from the photometry results. Of course these coordinates will not match exactly. Furthermore, there are many more stars in the clusters than are detected in the observations. This makes matching the found stars to their counterparts in the objects a difficult task; stars from the photometry files might coincide with multiple stars in the theoretical cluster based on just two of their spatial coordinates alone. The solution to this potential problem is to factor in the magnitude of the stars as well. This probably decreases the degeneracy considerably, if not fully lifting it. Before they can be used, however, the magnitudes have to be calibrated. More details on how the stars where matched across data sets are in section 4.4.3.

4.4.2 Calibration of the magnitudes

The instrument magnitudes can be converted to apparent magnitudes by using known reference stars. I use separate test images in the various filter bands for this purpose. Since I know the apparent magnitude of the stars in these test images, I then have a reference for the amount of flux that those magnitudes will produce in the image. Using the apparent

[24] Meaning 4096 by 4096 pixels and not the 3840 of the Ultra-HD standard often mistaken for 4k.

[25] Python is zero-indexed, so the pixels actually run from number 0 to number 4095, making 2047 and 2048 candidates for being in the 'middle'.

magnitudes of the reference stars and subtracting their respective instrument magnitudes, an average correction term is determined. This term is added to the measured instrument magnitudes of the stars in the clusters to get their calibrated apparent magnitudes:

$$m_{app} = m_{inst} + m_{cal} = m_{inst} + \frac{1}{N}\sum_{i=1}^{N}(m_{app,ref,i} - m_{inst,ref,i}) \tag{8}$$

with m_{cal} the averaged calibration term and the sum is over the reference stars, as indicated in the subscripts.

The now calibrated magnitudes can be used in making a CMD. This is the observational equivalent of an HRD, where the luminosity of a set of stars is plotted against their effective temperature. The CMD approximates this by taking the magnitude in one photometric filter as a measure for luminosity and the colour of the stars as a measure for their temperature. The latter might require some explanation: stars are relatively good approximations of black-body radiators, meaning most of the light they radiate is due to their (effective![26]) temperature. The nature of this thermal radiation is that more blue light is emitted if the object gets hotter. this means that for a set of two given filters, the ratio of light in the bluer filter compared to the redder filter will increase for a hotter star. The colour is defined as the magnitude in the bluer filter minus the magnitude in the redder filter. Since brightness increases inversely with magnitude, this means a higher value for the 'colour' means a redder and thus a cooler star. A disadvantage is that this measure of the colour becomes decreasingly sensitive to the temperature at higher temperatures.

4.4.3 Identification and outliers

There are many possible algorithms thinkable that would be able to link a star from the photometry data to one in the original cluster data. Not all implementations would perform equally: something that has become clear from trial and error.

One way of quickly and computationally efficiently matching two sets of data is with the use of a so-called k-dimensional lookup tree, or k-d tree for short. There is an implementation of this algorithm in `scipy.spatial` that I will use for this purpose. It can be used to compute any given number of nearest-neighbours of the photometry stars in the grid of actual stars in the cluster. The first order approach is to take the closest point in the plane of the sky as a match, by taking just one neighbour per photometry star. This might be the best approach to match stars between photometry filters, since the photometry inherently will not have too many stars overlapping or exactly on top of eachother. However, being limited to positions is not very effective in comparing to the whole cluster, because of the sheer amount of stars that appear close together on the plane of the sky.

The k-d tree is easily expanded to include magnitudes for a more-dimensional 'distance' measurement. Including one of the magnitude filters already improves the match-up a lot: the correlation between photometric and real magnitudes substantially tightens. This is still not an ideal way to match up the stars: the x and y positions are now less well matched and we want to give the position of the star a higher weight in determining a match. The idea is that first a set of stars in the real data is selected based on x and y coordinates alone. Then, using that subset of stars, the closest match is determined using the x and y coordinates as well as the magnitude of the corresponding filter[27]. Through testing, I found that a subset of stars of around 8 gives the best results in terms of both magnitude and coordinate correlations. These two steps still form a relatively simple process that takes full advantage of the computational efficiency of the k-d tree and does not increase the computation time by much.

[26] In the centre of the star it is much, much hotter.

[27] Each filter will have its own set of coordinates determined from the image of that filter.

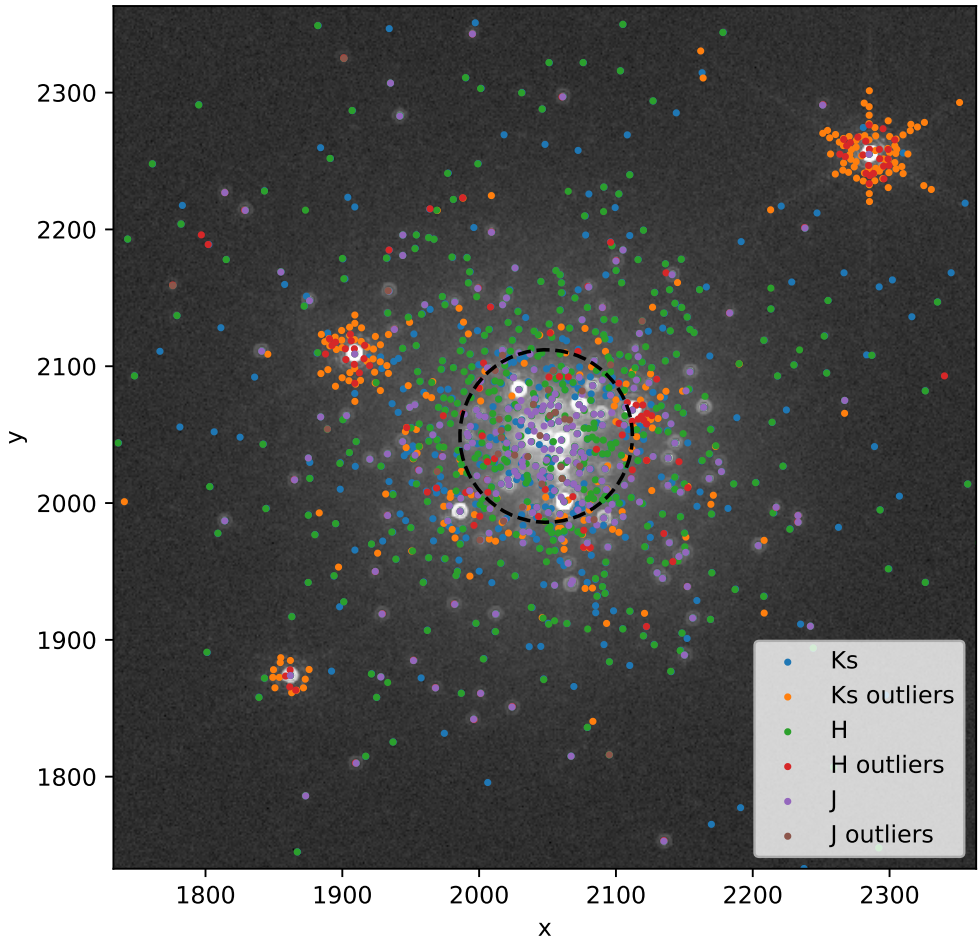

Figure 15: All the detected stars from the photometry of cluster 1 (M=$10^5 M_\odot$, D=$0.8 Mpc$, hlr=$1 pc$). The dashed black line indicates the HLR.

Another effective improvement to any matching method is to pre-select stars from the cluster that have a magnitude above the maximum magnitude of the photometry data for that filter. This removes a lot of faint stars that will only cause mismatches. To be entirely safe, I do not actually use the maximum magnitude as a selection criterion, but the maximum magnitude plus two. This still removes most of the stars from the cluster's data array.

Finally, outliers can be determined in various ways, depending on the needs of the user as well as the employed matching algorithm. One possibility is to use the distance between the real position of the star and the photometric position, designating everything above a certain distance as an outlier or mismatch. I have set this distance to 1.5 times the pixel scale of 4 milli-arcseconds. It can be visually verified that this outlier criterion is effective, as figure 15 shows. Most outliers (orange, red and brown points) are clustered around a few bright stars: these are the faint features of the PSF that are mistaken for stars. Note that not all found stars might be visible in the image due to overlap of the coloured points with those from other filters. The Ks filter was plotted first, so many of the blue points are hidden below points from the H or J filters.

A direct match between the different filters can naturally not depend on the magnitudes

for identification of the right star, since the stars do not have the same brightness in the different parts of the spectrum. A match can only be made based on the x and y coordinates found in the image, since this should be the same for each filter[28]. This is also done with the help of a k-d tree, although calculating the closest distance in this case is fast enough as well. The two methods give equivalent results, as shown in figure 16: all the orange dots from the brute force method overlap with blue points from the k-d tree method.

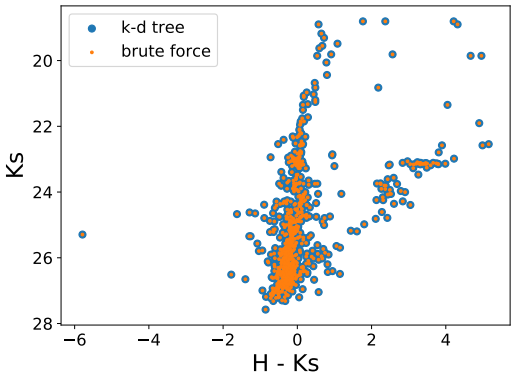

Figure 16: CMD resulting from the direct match-up of stars between filters, using two different methods that yield equivalent results.

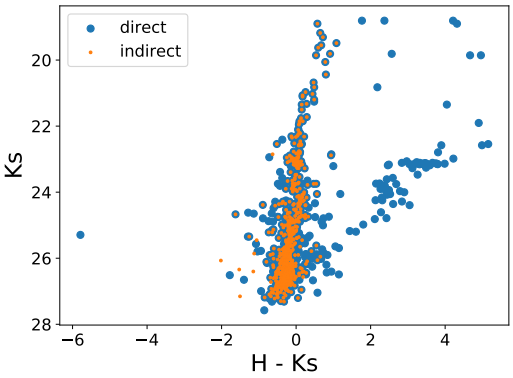

Figure 17: CMD resulting from two different methods of matching up the stars in different filters: a direct and an indirect way (via the real cluster).

A match between the photometry data for different filters can also be made indirectly. This is a nice way to check how well the algorithm described above, that takes into account the stellar magnitudes, performs. This indirect method first links the stars in each filter to the stars of the cluster, resulting in arrays of indices that reference to the real cluster. To connect the stars from different filters, we can look for matching indices in these arrays by means of an intersection. The intersection of the index arrays of two different filters tells us which stars have been found in both filters.

Figure 17 shows the comparison between the direct and the indirect method, both performed with the k-d tree. Again, the two methods share many of the exact same points, as would be expected if there is no flaw in either methods. The direct match, however, yields many more points that stray away from the general vertical trend. The advantage of the indirect method is that many mismatches are already naturally filtered out. If some feature is wrongly picked up as a star, it is unlikely to be matched to the same star in the cluster as a feature or star in another filter. Whereas the direct match will happily connect any misidentified star to another one in a different filter, as long as it is closest. Note that for both the direct and indirect method no outliers were taken out manually. Even after taking those into account, there are still erratic points left in the direct match.

If no reference data was available, like for real observations, one way to improve upon the direct match would be to look at every star individually to see if it can be confidently identified as a real star or not. This would be unfeasible if the number of stars found is very high, like often the case in GCs. The only other option is to find a fitting parameter (or combination thereof), like the sharpness of the source, that enables the correct detection of misidentified stars.

In the end, one can never be sure that each star has been linked to its true counterpart. For example when two stars of similar magnitude overlap in the image to form a single

[28] After calibration, at least. See 4.4.1.

unresolved point source, that 'star' can at best be linked to one of the two contributors. This means there is always a certain amount of error in comparing the photometry of crowded fields with the theoretical cluster model. The inclusion of binary stars would only accentuate this problem, causing bad magnitude matches even if the binaries are isolated.

4.4.4 Distance and age estimates

A CMD can be used to estimate several important quantities; a distance to the cluster, its age and its metallicity. All of these estimates rely on the identification of a certain part of the lifecycle of the stars, like the main sequence (MS), where stars spend most of their lifetime. At this stage, they are burning hydrogen into Helium in their core, and they do so on a relatively well defined and very stable location in the luminosity-temperature parameter space. This location is almost exclusively dependent on the initial (birth) mass of the star. Connecting all the stars that are in this stage of their life results in an only slightly meandering line going roughly from the top-left to the bottom-right in the HRD and CMD.

The distance can be determined from several stages in the evolution of the stars: the MS is an option, but also the asymptotic giant branch (AGB - later stage of evolution) and the horizontal branch (HB), where not Hydrogen but Helium is being fused in the core. This stage is more long-lived than others (except the MS), which means stars pile up in the same general area of the CMD. The advantage is that these stars are more luminous than those on the MS, so they can be seen from further away. The distance is measured by comparing this clump of HB stars to another (possibly theoretical) HB defined with absolute magnitudes. Absolute magnitudes, as opposed to apparent magnitudes, are defined at a fixed distance to the observer of 10 parsec (about 32.6 light years). The vertical distance between the positions of the two HBs in the CMD, measured in magnitudes, is called the distance modulus of the cluster. In different words, this is how many magnitudes the stars are fainter than their absolute magnitude due to their distance. The distance modulus is simply related[29] to the distance in parsec by the following formula.

$$\mu = m - M = 5 \cdot log_{10}(d) - 5 \tag{9}$$

In this formula, μ is the distance modulus, m is the apparent magnitude, M is the absolute magnitude (usually denoted by small and capital letters respectively) and d is the distance measured in parsecs. To get the distance, this is rewritten to the form:

$$d = 10^{\frac{\mu}{5}+1} \tag{10}$$

The step in this calculation that makes it an estimate is the determination of the difference in magnitude between the observed cluster and the theoretical HB (or MS or AGB) with absolute magnitudes.

The age of the cluster is determined assuming the cluster consists of one stellar population, which is often the case, but not always, as outlined in Li, Grijs, and Deng (2016). The idea behind estimating the age is that the heavier a star initially is, the quicker it burns through its nuclear fuel. After burning through its hydrogen on the MS, the star starts contracting to find a new equilibrium point where nuclear burning can take place. This causes it to move away from the position on the MS in the HRD and CMD. Stars generally move up and to the right during their evolution in such a diagram, causing a curve on the left end of the MS. If enough stars of the same population are observed, the point where the stars start evolving off the MS can be identified. This is called the MS turn-off point.

From theory and models, we know how long stars of certain masses spend on the MS. The turn-off point defines the transition from stars that have not gone through the hydrogen

[29] It is assumed here that there is no extinction of light between the observer and the source. The lower magnitudes in the presence of extinction would result in a distance estimate that is too high.

burning stage yet, and the ones that have just emptied their hydrogen fuel reserves. Therefore this point is directly related to how old the whole stellar population is. In practice, the stellar isochrone tracks mentioned earlier are fitted to the observations to obtain an estimate of the age of the GC. This is done by eye; least squares optimisation would likely not improve the accuracy.

The evolution of stars depends on the amount of heavy elements as well as the initial mass. The effect is smaller, but noticeable. Even different helium abundance can affect the stars' life cycle. Higher than normal[30] helium content makes stars hotter and brighter, shifting their position in the CMD. The metallicity, or fraction of elements heavier than helium, influences both of the estimates outlined here. To keep the amount of models to fit to the data manageable, I will take the metallicity of the clusters as a given constant. Measurements of the amount of metals can observationally be obtained by recording spectra. Certain atomic electron transitions found in the spectra would indicate the presence and amount of these metals. Estimating the metallicity is also possible photometrically by looking at the slope of the red giant branch, which becomes redder (so cooler) for a higher metal content.

[30]'Normal' being the primordial amount of helium produced in the Big Bang.

5 Results

The photometry data from IRAF/DAOPhot comes in many files, but the files containing the final PSF fitting results are designated 'allstar' in IRAF. These contain the fitted (x,y) coordinates and the instrument magnitudes. The measurements of the magnitudes will be refered to as the photometric magnitudes (or M_{phot}). After calibration these magnitudes can be compared to the original apparent magnitudes from the cluster (the 'real' magnitudes or M_{real}). The three iterations of the `ALLSTAR` task that were performed during the image analysis produce three separate files that are combined to form one array of all the found stars. In some cases it occurred that no additional stars were found in the second or third run, so some files are empty or missing. There are five cases where no stars were identified at all in at least one of the filters, not even on the first pass. This happened in systems with the parameters listed in table 1.

It is fairly clear from the parameters why no stars were found in these cases. These clusters all have the lowest mass, so the least amount of stars in them. Plus the distance is the furthest away, except for one of them. This cluster with a slightly closer distance of 10.4 Mpc is likely an unlucky deviation from the norm, since the other clusters at this distance and mass do still show several stars in the J filter. It has also become clear that the J filter always has the least number of detections. The other four systems are all at a distance of 13.6 Mpc, and only exclude the smallest of clusters in terms of HLR. The smallest cluster of the sequence of five has a total of one verified star in the J band, which must be a coincidental bright star at the edge of the cluster. The second and only other detection is the combined integrated light in the central region, which has been effectively recognised and removed as an outlier.

Table 1: The five clusters for which no stars were found in at least one of the filters.

Mass ($M_\odot$)	Distance (Mpc)	hlr (pc)
10^5	10.4	16
10^5	13.6	6
10^5	13.6	11
10^5	13.6	16
10^5	13.6	21

5.1 Error measures

DAOPhot provides a magnitude error that gives an indication of how good the PSF is fitted. Figure 18 shows these magnitude errors as function of both M_{phot} (on the y axis) and the distance from the centre (on the x axis). Most of these are well below half a magnitude, although some can reach well above one magnitude uncertainty. The closer a star is to the centre of the cluster and the fainter it is, the harder it becomes to get a good fit because of the very high background level and the close proximity of other bright stars. The error bars are smallest for far out, bright stars and increase both when moving down in brightness and when moving down in radius.

To get a better view of the accuracy, these magnitude errors are replaced by another quantity in the further results. All the stars are coupled to a star in the simulated cluster, which have known magnitudes. The quantity used for the accuracy of the magnitudes is the photometric magnitude minus the real magnitude (M_{phot} - M_{real}). Positive values are for stars that are fainter than they should be and negative values are for stars that are too bright. To facilitate in this definition, the vertical axis is inverted, so that stars *above* the zero line are increasingly too bright in the measurements.

While a strong and expected correlation is found between magnitude and magnitude error, little or no correlation exists between M_{phot} - M_{real} and the magnitude error. A more complex algorithm for linking up the photometry stars to the cluster stars could be imagined,

which would consider the magnitude errors in determining the best fitting star. There would in that case certainly be a correlation between these quantities, be it by design. However, the relatively simple algorithm as explained in 4.4.3 is aimed primarily at efficiency, while still being able to identify most of the stars correctly. It was compared against a slower algorithm that calculates all the individual distances between the points and selects the closest one. The resulting magnitude and position matches by the fast algorithm are equally as good or even better in some cases.

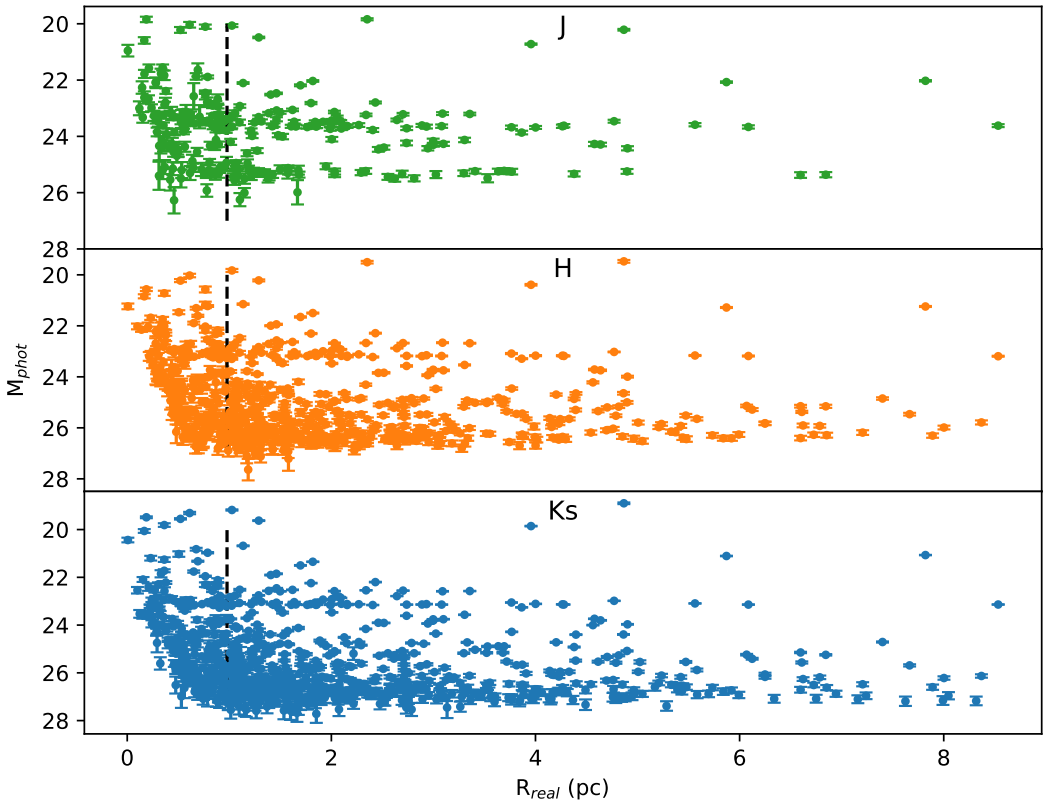

Figure 18: Photometric magnitudes in the three different filters versus the real radius in parsec, with errorbars indicating the uncertainty on the magnitude coming from the DAOPhot fitting algorithm.

5.2 Measured positions

Just like for the magnitudes, the error in the positions from the PSF fitting can be quantified with respect to the real cluster data. The algorithm that matches the stars in the photometry files to the cluster relies heavily on the positions of the stars, so any differences found are expected to be small. If a large difference is found, this is most likely because the detection was not an actual star. On the other hand, finding a large difference in magnitude is not necesarily indicative of an outlier, but can also mean the photometry is bad for that star. That is why the outlier selection criterion is based entirely on the positions. Figure 19 shows $x_{phot} - x_{real}$ versus x_{real} for one cluster: the outliers clearly follow a more vertical distribution. Most points tightly follow the zero line, with a wider spread around the origin of the axis than towards the extremes. This is due to more interference from other stars in the centre of the cluster, something that is expected to be seen also in the magnitude measurements.

The error on the measurement of the position as given by IRAF is well into sub-pixel level, being of the order 10^{-2}. This is in line with part of the $x_{phot} - x_{real}$ values, but many of those are of the order 10^{-1}, up to around 1 pixel off. Of course this is much higher for the outliers, that can be many pixels off. Especially when they occur in clusters around bright stars, where the faint features of the PSF get well above the noise level. This can be seen in figure 19 as the vertically stretched structures that stay close around one x value.

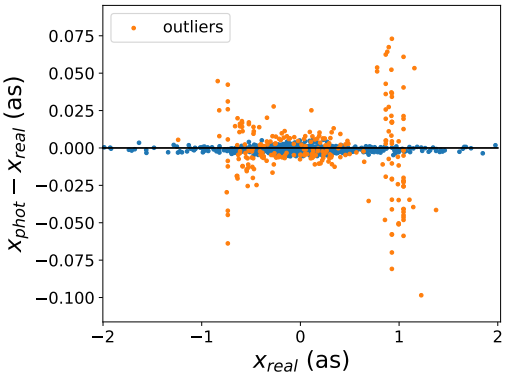

Figure 19: Accuracy of the x coordinate in the Ks filter for cluster 1. Outliers are selected based on their distance from the matched real star.

The criterion for outliers is a distance of more than 1.5 pixels away from the star it has been matched to. So naturally, the maximum distance seen for non-outliers is 1.5 pixels. This only happens in the cluster centre: beyond the HLR the spread decreases to roughly ± 0.25 pixels. This does, however, depend on the cluster a bit.

5.3 Magnitude accuracy

More important, in the context of this thesis at least, is accuracy of the magnitudes. Errors in the magnitude will be enlarged in the colours of the stars, to be used in the CMD, simply because we subtract two similar numbers.

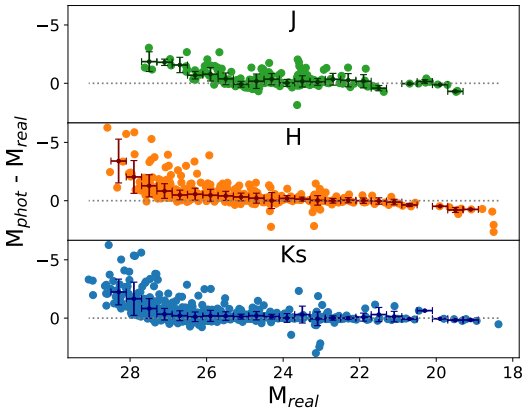

Figure 20: Plot of the magnitude accuracy versus the real magnitude for cluster 1 with binned averages and standard deviations.

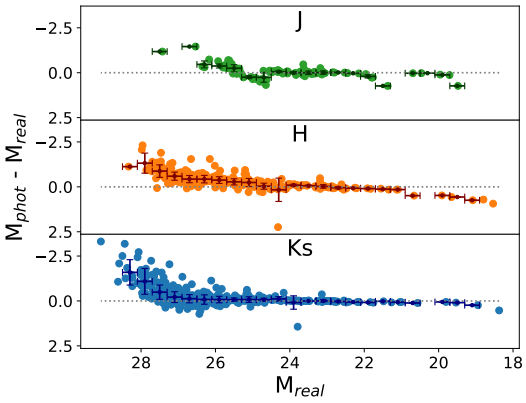

Figure 21: Same as in 20, but now only stars outside the HLR are included. The y scale is only half as big.

We can look at the magnitude accuracy (here: M_{phot} - M_{real}) in several ways, one of which is as a function of the (real) magnitude. This reveals how the accuracy deteriorates for fainter objects. More random error is expected in faint objects, because they are closer to the noise level. This will express itself as larger scatter. Another effect that is clearly visible in the data is an overestimation of the brightness for fainter stars, resulting in a negative M_{phot} - M_{real}. This is due to the fact that these faint stars would not have been detected was it not for a 'boost' to their brightness. This boost can be due to noise that coincides with the star exactly in the right way, or due to several stars working together to form one

brighter object in the image. The latter is a more likely scenario in the context of crowded stellar fields, where many stars can be projected on the same position on the sky[31].

As can be seen in figure 20, the magnitude accuracy over the whole cluster is actually very poor at the low magnitude end. However, if only the stars outside of the HLR are taken into account, the spread improves a lot. The same conclusion can be made when looking at the magnitude accuracy as function of the radial distance of the stars (see figure 22). This confirms the expectation that the photometry would be worse in the cluster centre. The overall shape of the distribution of points in figure 21 is the same as that of the previous plot, but the scale on the y-axis is much smaller. The highest deviation of the real magnitude has halved to minus two and a half from minus five. Also the spread at all other magnitudes has decreased visibly. What can also be seen in the J filter are a few points in odd positions; these will be ignored for now, but are further discussed in section 6.

The fairly wide spread seen in these figures (20 to 22) is also decreased when looking at different clusters. These figures were all for cluster 1, which is a very compact cluster with a HLR of 1 *pc*. The clusters that are less compact also show less inaccurate magnitudes towards their centres, which is not unexpected. If the stars are spread out over a larger area, then less interference is expected in the photometry measurements. In figures 23 and 24, displaying cluster 39, the maximum deviation from the real magnitude is about four times lower than that of cluster 1. It almost halves again when selecting for stars outside the HLR (fig. 24). There is a strong correlation between the mean deviation from M_{phot} - M_{real} = 0 at a certain radius and the HLR of the clusters. This can be summed up with a plot of the mean M_{phot} - M_{real} in a radius bin positioned just outside the HLR, versus the radius of the midpoint of this bin for each cluster, as shown in figure 26.

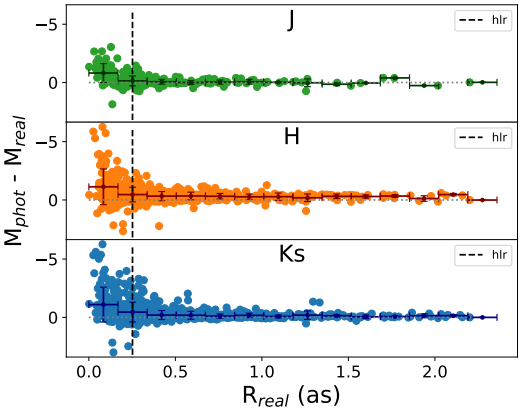

Figure 22: Accuracy of the magnitude as function of radius in cluster 1. The dashed vertical line depicts the HLR.

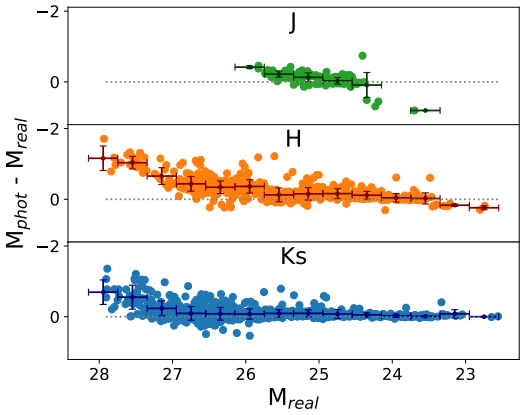

Figure 23: The magnitude accuracy versus the real magnitude for cluster 39 with running average.

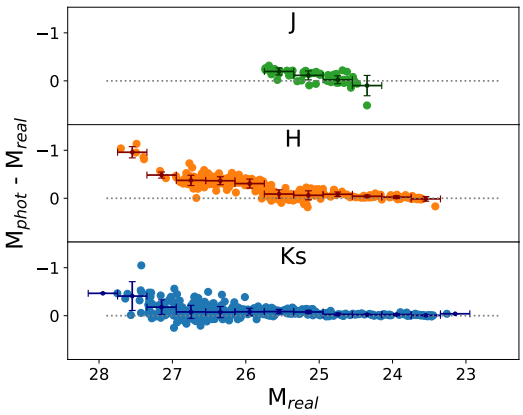

Figure 24: Same as in 23, but now only stars outside the HLR are included.

[31] The same in terms of the resolving power. Two very close stars will result in just one point source.

The clusters with the smallest HLR vary widely in accuracy of the magnitudes at the HLR, while this scatter quickly decreases for the larger clusters. It is interesting to note that while the maximum spread is smallest for the H filter, the improvement is best for the Ks filter. The spread in the latter filter quickly decreases below -0.3 in mean (M_{phot} - M_{real}), at a HLR of 11 *pc*. The standard deviation in these bins is of similar size, at around 0.3 and below.

For comparison, the same quantity is plotted for bins at the centre of each cluster in figure 25. While the maximum value of the scattered mean values does not increase significantly (except for in the J filter, where it roughly doubles), many more of the clusters are at higher values. None of the clusters with a HLR of 1 *pc* quite reach the zero mark, which does happen for the mean values measured *at* the HLR. Another clear difference is that the spread of means decreases a lot less strongly with cluster size: only at a HLR of 16 *pc* does the highest (negative) mean value in the Ks and H band reach below -1. Interestingly the standard deviation in these bins is comparable, which is due to the systematic over-estimation of magnitudes in the centre causing the running mean to increase while the running standard deviation stays of similar size.

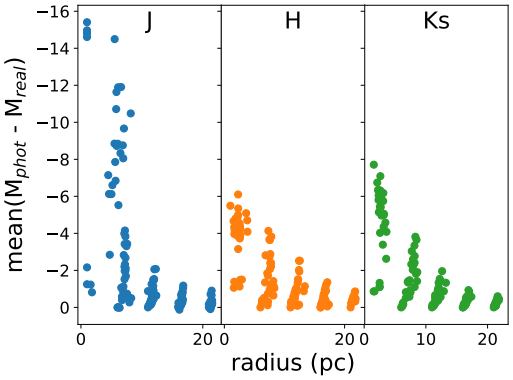

Figure 25: Mean accuracy of the magnitudes in the radius bin at the centre of each cluster as function of the centre radius of the bins.

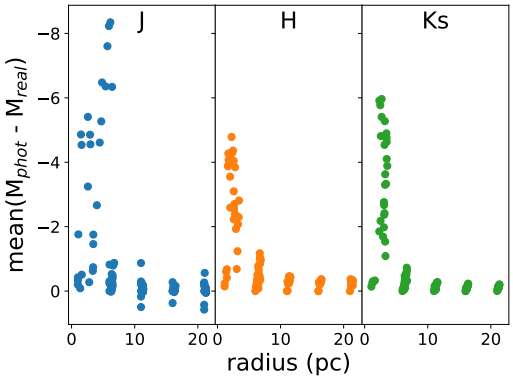

Figure 26: Mean accuracy of the magnitudes in the radius bin just outside the HLR of each cluster as function of the midpoint of the bins.

Another useful quantity to measure is the 50% completeness limit, which gives the highest magnitude for which 50% of the stars in a magnitude bin are still being recovered (fig. 27). Magnitudes are divided into 20 bins for each cluster. This measure is not a function of the distance to the cluster, since the distance does not determine how well faint stars can be seen. It does also not matter for this measurement if we select only stars outside of the HLR: the faintest stars are most likely already being identified on the outskirts of the cluster rather than in the centre. There *is* a correlation, again, with the size of the clusters. Many small clusters are too crowded for very faint stars to be resolved in the combined light of the

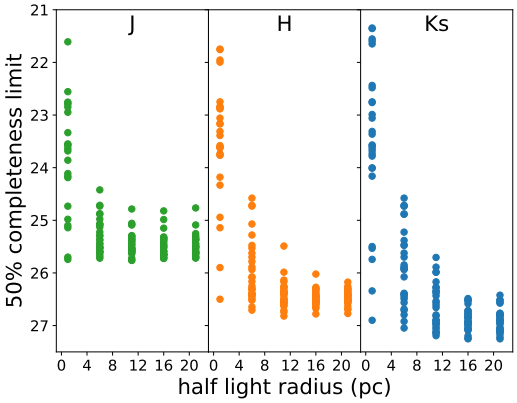

Figure 27: The highest magnitude for which 50% of the stars is recovered from the image of each cluster.

cluster, resulting in a much lower completeness limit. Another variable that has a large effect on the completeness limit is the photometric filter. For the images that have been made for this project, the Ks filter performs best in this regard, with the H filter following closely after it. The J filter performs worst, by around a magnitude. Not an unexpected result after seeing that the amount of successfully detected stars for this filter was consistently much lower than for the other two.

5.4 Distance, age and metallicity

Distances to the clusters were estimated using the HB of giant stars, that shows up as a clump of stars close together on the CMD. Because of the lower completeness limit in the J filter, the preferred CMD with J-Ks on the x-axis could only be used to get distances for the clusters at 800 *kpc*. J-Ks is preferred over H-Ks because there is more sensitivity to the evolutionary stage of the star in this colour. In practice this means that the evolutionary tracks are less vertical and specific features/stages can more easily be identified. The HB is a feature that can still be identified in the H-Ks CMDs, leading to the successful distance determination for both the closest and the next distance, which is 4 *Mpc*. However, it was still not possible to estimate the distance for the smallest clusters in terms of HLR at this distance, and it was hard for the least massive clusters due to the lower number of stars.

Table 2: Distance modulus estimates based on the HB. Masses (in $M_\odot$) are listed horizontally and the HLR (in *pc*) vertically. The first column states the colour and distance for the set of clusters indicated.

Mass >	HLR ∨	10^5	$6.1 \cdot 10^6$	$2.1 \cdot 10^6$	$8.1 \cdot 10^6$	$4.1 \cdot 10^6$	$1.01 \cdot 10^7$
J-Ks	1	24.55	24.52	24.52	24.52	24.52	24.53
D=0.8*Mpc*	6	24.55	24.53	24.52	24.52	24.52	24.52
	11	24.55	24.53	24.52	24.52	24.52	24.52
	16	24.55	24.52	24.52	24.52	24.51	24.52
	21	24.55	24.52	24.52	24.52	24.51	24.52
H-Ks	1	24.54	24.52	24.52	24.53	24.53	24.52
D=0.8*Mpc*	6	24.53	24.52	24.52	24.51	24.51	24.51
	11	24.52	24.52	24.52	24.51	24.52	24.51
	16	24.53	24.52	24.51	24.51	24.52	24.52
	21	24.52	24.51	24.52	24.52	24.51	24.51
D=4.0*Mpc*	1	-	-	-	-	-	-
	6	27.95	27.95	27.95	27.95	27.95	27.95
	11	27.95	27.95	27.95	27.95	27.95	27.95
	16	28.00	27.96	27.95	27.95	27.94	27.96
	21	27.94	27.94	27.95	27.95	27.94	27.95

The table above lists the estimates of the distance modulus. The real distance moduli for the two distances here are 25.515 and 28.010. The error in the estimates is based on the amount of spread in the HB: in the J-K colour it is 0.10 and for H-Ks it is 0.20, resulting in an approximate relative error of 5% and 10% in the distances. For the clusters at 4 *Mpc*, the distances where all underestimated by about 0.06 in distance modulus. Distances can in principle also be determined from the AGB, but this was only attempted for the most massive clusters. Due to the spread in the positions of the stars in the CMD these estimates where inherently less accurate, with a spread of ~1.8 in distance modulus (so ±0.9). This method did work all the way up to 10.4 *Mpc*, although the uncertainty there increased to

±1.0. Figure 28 shows an example of how a cluster is matched to the theoretical isochrones placed at different distance moduli.

Distances for GCs are often known, due to their association to galaxies. The distances to many nearby galaxies have been measured, and while the GCs that orbit those galaxies might not be at exactly the same distance, this is a good approximation due to the large relative distances to these galaxies. These distances can be used to compare the GC star population with isochrone models. In the context of galaxy formation and evolution, two more interesting metrics to estimate are the age and metallicity of the stars in a GC.

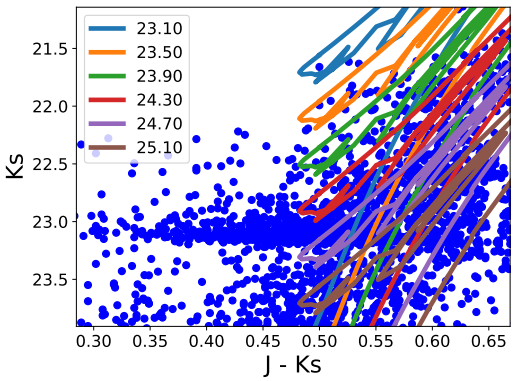

Figure 28: Several isochrones are plotted for various distances to find the best match for the cluster.

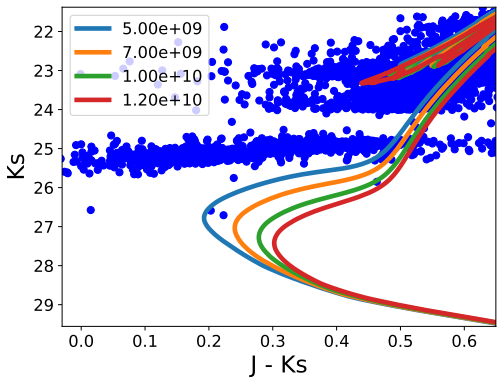

Figure 29: Several isochrones are plotted for various ages to find the best match for the cluster. In this case, no fit can be made for the age.

Age determinations were attempted under the assumption of a known distance. Similar to how distance is estimated, the cluster is compared to isochrones of different population ages. Unfortunately, the part of the isochrone most sensitive to age, the MS turn-off point, is not covered by stars from the clusters. The turn-off point is at a magnitude that was just not reached with the exposures made here. Another issue that can be foreseen is inaccuracy in colour at the higher magnitudes. The big spread in colour would mean that even when the turn-off point is reached, the age of the cluster would be ambiguous. Only when much higher magnitudes are obtained the colour would be recovered sufficiently precisely to distinguish between ages. However, more short exposures covering a longer total exposure time could perhaps improve on the overall accuracy.

Metallicity on the other hand is obtained from the brighter regions from the CMD. The position and more importantly, the slope of the red-giant branch changes for different metallicities. Again, the distance is assumed known and the age is also fixed. Different age would not influence this estimate much, because the position of the red-giant branch is not affected much. From the example in figure 30, it can be seen that the isochrone lines change most between metallicity values at the top of the red-giant branch, where the AGB is. This is also the part of the CMD where the magnitude and colour accuracy is best for our photometry.

For systems with fewer stars, like the one in figure 30, the range of possible values of the metallicity is estimated to be 0.0014 to 0.00045. This includes the actual value (0.0014 or $0.1Z_{\odot}$) and spans less than an order of magnitude. For the systems with many more stars, like the one in figure 31, the spread becomes a bit wider. I estimate that a metallicity between 0.0045 and 0.00014 is a reasonable range for these higher mass/nearby systems. The higher value in that range is still not very fitting to the data, but included as a firm upper bound. This shows that it can with some certainty be recovered that the metallicities of

these systems is much lower than the solar value: at least below 0.0045 or $\sim 0.3 Z_{\odot}$.

Ignored in this analysis are the strange features in the CMD that are best expressed as jumps to the right in the colour of groups of stars. These jumps where ignored because they are clear outliers to what the actual colours of the population should have been. Their origin is thus far unkown: they will be discussed further in section 6 below.

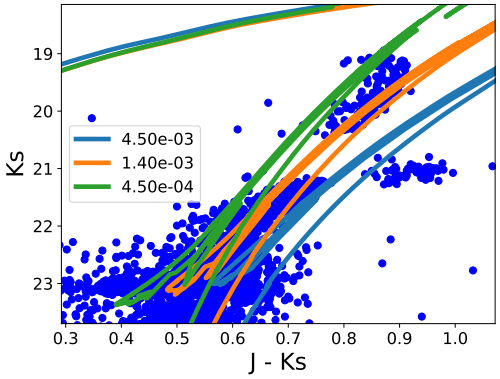

Figure 30: Different theoretical isochrones, for changing value of the metallicity, plotted over cluster 28.

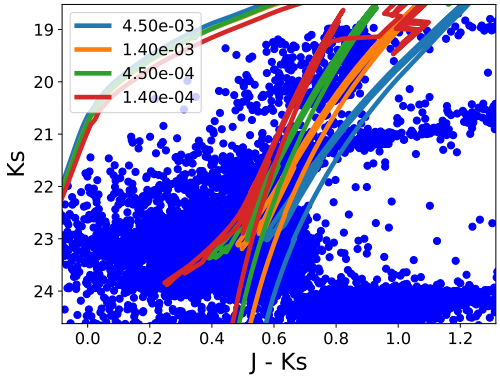

Figure 31: A slightly wider range of values for the metallicity, plotted over cluster 103.

6 Conclusion & Discussion

The results show that star positions can in principle be measured to sub-pixel precision. The analysis shows that this precision is not always representative of the accuracy when compared to the reference cluster. This bad representation mainly holds for the central region, below the HLR, where the light of many stars mixes together. The outer parts show accuracy that is well below the pixel scale.

The magnitude measurements show a similar trend: the precision is often optimistic compared to the accuracy, which is taken to be M_{phot} - M_{real}. There is no correlation between the measured precision and the obtained accuracy. The precision is heavily dependent on both the magnitude (brighter equals more precise) and on the radial distance from the centre (further means more precise). The spread in accuracy of the magnitudes is an even stronger function of radius, partly due to how stars are cross-matched between the photometry and the real cluster data. This matching takes the positions of the stars as a more important metric, with hard constraints: outliers are solely selected based on a bad position match. The magnitude cannot be regarded as a strong matching criterion, because in crowded stellar fields multiple stars can align to form a single, brighter point source. This is the reason for the strong dependence on radius for an accurate measurement. In the centre of these clusters, the stars can be many magnitudes too bright, while at the HLR this spread is usually already halved. At the further outskirts the magnitudes are recovered to within about half a magnitude, varying slightly from case to case. The magnitude accuracy was sufficient for good distance and metallicity estimates, although the colour reproduction limited a more precise value for the metallicity.

Distance estimates using the HB were only possible for the two closest distances, 0.8 and 4.0 Mpc. The estimated error in the distance is about 5 to 10 percent depending on whether the J-Ks or H-Ks CMD was used. The former colour measure did not allow for distance estimates of the 4.0 Mpc clusters, due to the lower magnitude limit for the J filter. The age and metallicity where kept constant for the distance estimates, to reduce the parameter space of possible matching isochrones. Using the AGB enabled estimating the distance of clusters up to and including 10.4 Mpc, although this method results in a much larger uncertainty (about ± 1 in distance modulus).

Distances are often a known quantity for GCs, due to their proximity to galaxies. In principle this allows a better estimate of the age and metallicity of the cluster, by setting the distance to a fixed, independently measured value. Age determination depends on what metallicity the cluster has, since the part of the stellar population used for this (the MS turn-off) is sensitive to both. However, in the case of the cluster images generated here, the MS turn-off point was not reached in terms of magnitude, so no age estimates were made. A longer total exposure time might enable age determination, under the condition that sufficient colour accuracy can be obtained. The spread in colour at the lowest magnitudes is so wide that no successful distinction would be possible between the various theoretical isochrones even if the magnitude of the MS turn-off was reached.

Metallicity estimates can be made using the (upper part of the) rad-giant branch, which is much less sensitive to the age. This is also where the very bright stars are, so this estimate is possible for all but the furthest of clusters. Depending on the amount of stars in the CMD, the spread in metallicity was determined to be 0.0014 to 0.00045 or - for when more stars were present - 0.0045 to 0.00014. The higher number of stars meant a larger spread in colours, making the match to different isochrones a bit more ambiguous.

The magnitude of stars was not only seen to increase in spread towards lower brightness (as well as towards lower radius, as explained above): there is also a systematic error for the faintest stars (see figures 20, 21, 23 and 24). This is most likely due to several stars forming one point source, as one measurement can only be assigned to one of the original stars. No matter to which star the magnitude measurement is attributed, it will always be too bright.

Another effect that is less severe, but still noticeable in many clusters, is a slight underestimation of the magnitudes at the bright end. This might be caused by saturation of the detector for these bright stars. The central peak of the PSF is very sharp, and when this gets near the saturation limit or goes over it, the top of the peak is flattened in the image. The effect of this is partially mitigated by setting a saturation limit for pixel values in DAOPhot, but the PSF fitting may still be affected. If the central peak is lower, the overall flux measured by fitting will be lower as well. Something that hints at this is the fact that bright stars are not subtracted out very well: there are fairly high residuals in the wings of the PSF while the centre becomes negative. This can most likely be avoided by taking shorter exposures that do not come close to saturating for any of the stars and adding them up after the fact. This would significantly increase the simulation time for the images, unfortunately; although the `SimCADO` source objects can be re-used, saving some time[32].

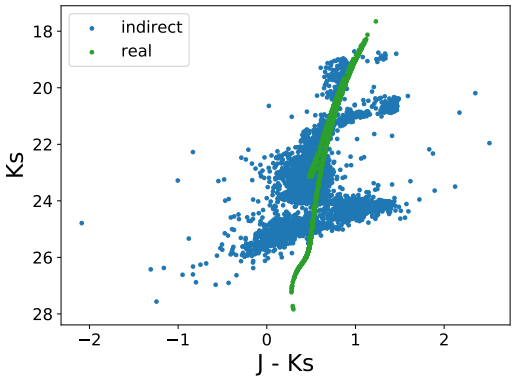

Figure 32: CMD of cluster 53 showing jumps in colour at certain magnitudes. The real data are plotted in green.

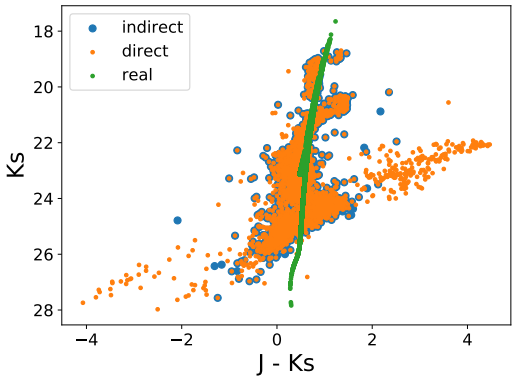

Figure 33: CMD of cluster 53 showing the direct matching between filters and the indirect matching via the real cluster.

6.1 Strange features

Finally, the jumps in colour at certain points in the CMD have to be adressed. Figure 32 shows the CMD of cluster 53, in which the jumps are particularly clear. There appear to be three distinct magnitude ranges where it happens, and the jumps in colour seem to consist of one or more steps. The reason why this happens is unclear so far. The same, or at least similar features are seen in the graphs of M_{phot} - M_{real} versus M_{real} (see figures 23 and 24). From the latter graphs it seems clear for all clusters that the Ks filter is not or insignificantly affected. The jumps are present in both the H and J filter, with the second one showing them most dominantly.

The first point of potential failure that could cause this is the matching algorithm that couples the photometry in different filters to eachother or couples the photometry to the cluster data. Different such algorithms where tried and all of them appear to reproduce these jumps. In figure 33 two of these algorithms are compared: one 'direct' meethod that simply cross-matches photometry data between filters and another 'indirect' method that first makes a match to the real stars for each filter and then connects photometry stars that were linked to the same cluster star. The latter is used throughout this project. From the comparison it is clear that most of the points lie directly on top of each other, although the direct method has a number of points that stray much further from ground truth.

The next potential point of failure would be the photometry, performed with IRAF. There are many variables going into this process that all have varying amounts of influence

[32] On the other hand, it is probably still faster than doing actual observations.

on the end result. It is difficult to say what exactly could be a cause of these features. Perhaps making PSF models from separate images, or not including flat-field images has this unexpected effect. The last possible failure point, if also the photometry would be ruled out, would be the image simulation process in `SimCADO`. It is not clear to me how exactly this part of the process could reproduce the observed effect, but since `SimCADO` was by no means in a finished state at the time of the simulations it is not unthinkable that this is where the strange features are produced.

6.2 Outlook

Hopefully, when the process of creating and analysing images of clusters is repeated in future projects taking into account some of the findings here, the problem of the strange colour/magnitude jumps can be either avoided or solved. This brings me to the point of future work: there are still many things that can be investigated in more detail. The parameter space explored is still limited in both range and resolution. With 150 clusters over three dimensions of properties, only 5 or 6 steps could be made in each dimension.

More importantly, it is definitely worth a whole second project to characterise the presented findings over a whole range of exposure times. Analysing the performance at 0.5, 1, 2, 4 and 6 hours of total exposure time would have a lot of added value, and could provide a graph similar to that in figure 3. Exposures of no more than a couple of minutes each would ensure the saturation limit is not reached in most scenarios.

Since MICADO will probably have to wait a bit longer for MCAO to arrive, it would also be usefull to look at the effect of the field-varying PSF of SCAO. This would most likely involve looking into different analysis algorithms for the photometry, because DAOPhot is not ideal for that kind of work.

Any future project that would make use of synthetic images of (groups of) stars would be able to make use of `StarPopSim`. This greatly reduces the amount of work that goes into making the images, potentially enabling more images to be made and/or a better analysis of the resulting data to be produced. Currently the focus of `StarPopSim` is naturally to produce life-like GCs. Other star clusters can easily be made too, with addition of some radial profiles if needed.

I enjoyed working on `StarPopSim`, and intend to keep working on it as a side project. The ability to model reasonably good looking spiral galaxies has been a wish from the start. Its modular nature allows different kinds of objects to be added in with relative ease. Other functionality could include modelling variability in the stars, or (perhaps more ambitiously), dynamic evolution of stars over short timescales to test astrometry methods.

Acknowledgements

I would like to thank Søren Larsen for giving me the opportunity to do this project. I was specifically looking for something that would involve building a substantial program from scratch, not to say that the content of the project was not important as well. I learned a lot about globular clusters, observing and writing code. Looking back at Python scripts from before I started this project, I sometimes feel urged to rewrite them properly.

Thanks also go to my second corrector, Paul Groot, for taking the time to grade my efforts.

Kieran Leschinski, who is developing `SimCADO`, also deserves a mention here. He has helped me with numerous problems and bugs when I was simulating the images. In return I was able to fix something here and there in `SimCADO` as well.

Thanks to Anne for supporting me even when I was sure I would not get it done in time.

To the amazing people at the astrophysics department, master students, PhD students, postdocs and all other staff: thanks for making my time at the department so enjoyable and unforgettable.

Also thanks to my parents for patiently waiting to see me again while I was frantically working on the last stretches of this thesis.

Making `StarPopSim` would not have been possible, or a lot slower, without the freely available packages: NumPy (T.E. Oliphant, 2006; Van der Walt, Colbert, and Varoquaux, 2011), SciPy (Jones, T. Oliphant, P. Peterson, et al., 2001), Matplotlib (Hunter, 2007) and Astropy (Astropy Collaboration, Robitaille, et al., 2013; Astropy Collaboration, Price-Whelan, et al., 2018).

References

Abel, N.H. (1826), "Auflösung einer mechanischen Aufgabe". In: *Journal für die reine und angewandte Mathematik,* 1, pp. 153–157.

Althaus, L.G. et al. (2010), "Evolutionary and pulsational properties of white dwarf stars". In: *A&ARv* 18.4, pp. 471–566. DOI: 10.1007/s00159-010-0033-1.

Arcidiacono, C. et al. (2014), "End to end numerical simulations of the MAORY multiconjugate adaptive optics system". In: *Proc. SPIE* 9148, 91486F, p. 8. DOI: 10.1117/12.2055608.

Astropy Collaboration, A. M. Price-Whelan, et al. (2018), "The Astropy Project: Building an Open-science Project and Status of the v2.0 Core Package". In: *AJ* 156.3, id. 123, p. 19. DOI: 10.3847/1538-3881/aabc4f.

Astropy Collaboration, T. P. Robitaille, et al. (2013), "Astropy: A community Python package for astronomy". In: *A&A* 558, id. A33, p. 9. DOI: 10.1051/0004-6361/201322068.

Becker, A.C. et al. (2007), "In Pursuit of LSST Science Requirements: A Comparison of Photometry Algorithms". In: *PASP* 119.862, pp. 1462–1482. DOI: 10.1086/524710.

Bonaccini Calia, D. et al. (2010), "Laser Development for Sodium Laser Guide Stars at ESO". In: *The Messenger* 139, pp. 12–19.

Bradley, L. et al. (2019), "astropy/photutils: v0.6". In: -. DOI: 10.5281/zenodo.2533376.

Choi, J. et al. (2016), "MESA Isochrones and Stellar Tracks (MIST). I. Solar-scaled Models". In: *ApJ* 823.2, id. 102, p. 48. DOI: 10.3847/0004-637X/823/2/102.

Cummings, J.D. et al. (2018), "The White Dwarf Initial-Final Mass Relation for Progenitor Stars from 0.85 to 7.5 $M_\odot$". In: *ApJ* 866.1, id. 21, p. 14. DOI: 10.3847/1538-4357/aadfd6.

Davies, R. et al. (2010), "MICADO: the E-ELT adaptive optics imaging camera". In: *Proc. SPIE* 7735.77352A. DOI: 10.1117/12.856379.

Dotter, A. (2016), "MESA Isochrones and Stellar Tracks (MIST) 0: Methods for the Construction of Stellar Isochrones". In: *ApJS* 222.1, id. 8, p. 11. DOI: 10.3847/0067-0049/222/1/8.

ESO (website 2019), *ESO's Extremely Large Telescope*. eso.org.

— (2011), *E-ELT Enclosure*. eso.org.

— (2018), *First ELT Main Mirror Segments Successfully Cast*. eso.org.

Fiorentino, G. et al. (2017), "MAORY science cases white book". In: arXiv:1712.04222.

Fryer, C.L. et al. (2012), "Compact Remnant Mass Function: Dependence on the Explosion Mechanism and Metallicity". In: *ApJ* 749.1, id. 91, p. 14. DOI: 10.1088/0004-637X/749/1/91.

Gieles, M. et al. (2018), "Concurrent formation of supermassive stars and globular clusters: implications for early self-enrichment". In: *MNRAS* 478.2, pp. 2461–2479. DOI: 10.1093/mnras/sty1059.

Gilmozzi, R. and J. Spyromilio (2007), "The European Extremely Large Telescope (E-ELT)". In: *The Messenger* 127, p. 11.

Gratton, R.G., E. Carretta, and A. Bragaglia (2012), "Multiple populations in globular clusters". In: *A&A Rev.* 20, id. 50. DOI: 10.1007/s00159-012-0050-3.

Hippler, S. (2019), "Adaptive Optics for Extremely Large Telescopes". In: *JAI* 8.2, id. 1950001-322. DOI: 10.1142/S2251171719500016.

Hunter, J.D. (2007), "Matplotlib: A 2D Graphics Environment". In: *Computing in Science & Engineering* 9.3, pp. 90–95. DOI: 10.1109/MCSE.2007.55.

Johns, M et al. (2012), "Giant Magellan Telescope: overview". In: *Proc. SPIE* 8444.84441H. DOI: 10.1117/12.926716.

Jones, E., T. Oliphant, P Peterson, et al. (2001), *SciPy: Open source scientific tools for Python*. URL: http://www.scipy.org/.

King, I. (1962), "The structure of star clusters. I. an empirical density law". In: *AJ* 67, p. 471. DOI: 10.1086/108756.

Kroupa, P. (2001), "On the variation of the initial mass function". In: *MNRAS* 322.2, pp. 231–246. DOI: 10.1046/j.1365-8711.2001.04022.x.
Larsen, S. S. (2011), "Young Star Clusters in External Galaxies". In: *Workshop Proc. "Stellar Clusters & Associations: A RIA Workshop on Gaia"*.
— (2013), "Star clusters and Extragalactic Stellar Populations". In: *Conference Proc. "370 Years of Astronomy in Utrecht"*.
Larsen, S. S., J. P. Brodie, and J. Strader (2017), "Detailed abundances from integrated-light spectroscopy: Milky Way globular clusters". In: *A&A* 601, id. A96, p. 23. DOI: 10.1051/0004-6361/201630130.
Leschinski, K. et al. (2016), "SimCADO - an instrument data simulator package for MICADO at the E-ELT". In: *Proc. SPIE* 9911, id. 991124, p. 17. DOI: 10.1117/12.2232483.
Lewin, S (2017), *Making the Giant Magellan Telescope's Massive, Incredibly Precise Mirrors*. space.com.
Li, C.-Y., R de Grijs, and L.-C. Deng (2016), "Stellar populations in star clusters". In: *RAA* 16.12. DOI: 10.1088/1674-4527/16/12/179.
Minniti, D. (1996), "Field Stars and Clusters of the Galactic Bulge: Implications for Galaxy Formation". In: *ApJ* 459, p. 175. DOI: 10.1086/176879.
NASA (website 2019[a]), *About the James Webb Space Telescope*. jwst.nasa.gov.
— (website 2019[b]), *Hubble Servicing Missions Overview*. nasa.gov.
Odenkirchen, M. et al. (2001), "Detection of Massive Tidal Tails around the Globular Cluster Palomar 5 with Sloan Digital Sky Survey Commissioning Data". In: *ApJ* 548.2, pp. L165–L169. DOI: 10.1086/319095.
Oliphant, T.E. (2006), *A guide to NumPy*. URL: http://www.numpy.org/.
Paxton, B. et al. (2011), "Modules for Experiments in Stellar Astrophysics (MESA)". In: *ApJS* 192.1, id. 3, p. 35. DOI: 10.1088/0067-0049/192/1/3.
— (2013), "Modules for Experiments in Stellar Astrophysics (MESA): Planets, Oscillations, Rotation, and Massive Stars". In: *ApJS* 208.1, id. 4, p. 42. DOI: 10.1088/0067-0049/208/1/4.
— (2015), "Modules for Experiments in Stellar Astrophysics (MESA): Binaries, Pulsations, and Explosions". In: *ApJS* 220.1, id. 15, p. 44. DOI: 10.1088/0067-0049/220/1/15.
Peng, E. W. et al. (2006), "The ACS Virgo Cluster Survey. IX. The Color Distributions of Globular Cluster Systems in Early-Type Galaxies". In: *ApJ* 639.1, pp. 95–119. DOI: 10.1086/4982109.
Peterson, C.J. (1987), "Ages of globular clusters". In: *PASP* 99, pp. 1153–1160. DOI: 10.1086/132098.
Sanders, G.H. (2013), "The Thirty Meter Telescope (TMT): An International Observatory". In: *J. Astrophys. Astr.* 34.2, pp. 81–86. DOI: 10.1007/s12036-013-9169-5.
Shapley, H. (1916), "Studies based on the colors and magnitudes in stellar clusters. I. The general problem of clusters". In: *Contributions of the Mount Wilson Solar Observatory* 115, p. 22.
Stetson, P.B. (1987), "DAOPHOT - A computer program for crowded-field stellar photometry". In: *PASP* 99, pp. 191–222. DOI: 10.1086/131977.
Tody, D. (1986), "The IRAF Data Reduction and Analysis System". In: *Proc. SPIE* 627, p. 733. DOI: 10.1117/12.968154.
Van der Walt, S, S.C. Colbert, and G. Varoquaux (2011), "The NumPy Array: A Structure for Efficient Numerical Computation". In: *Computing in Science & Engineering* 13.2, pp. 22–30. DOI: 10.1109/MCSE.2011.37.
VandenBerg, D.A. et al. (2013), "The Ages of 55 Globular Clusters as Determined Using an Improved ΔV_{TO}^{HB} Method along with Color-Magnitude Diagram Constraints, and Their Implications for Broader Issues". In: *ApJ* 775.2, id. 134, p. 47. DOI: 10.1088/0004-637X/775/2/134.

Vidal, F. et al. (2018), "End-to-End simulations for the MICADO-MAORY SCAO mode". In: *Proc. AO4ELT5* 43, p. 12. DOI: 10.26698/AO4ELT5.0043.

Wizinowich, P. et al. (2000), "First Light Adaptive Optics Images from the Keck II Telescope: A New Era of High Angular Resolution Imagery". In: *PASP* 112.769, pp. 315–319. DOI: 10.1086/316543.

Zinn, R. (1985), "The Globular Cluster System of the Galaxy. IV. The Halo and Disk Subsystems". In: *ApJ* 293, p. 424. DOI: 10.1086/163249.

A StarPopSim functions

The functions that enable the user to get most of the functionality out of `StarPopSim` are listed below. There are a number of core functions that are tied to the `AstObject` object: the hub for all the data and information about the star cluster (or other astronomical objects in the future). Then there are functions that are mainly meant to aid in the visualisation of the object in various ways. `StarPopSim` is independent from any telescope-optical-train simulating codes: there is a separate module linking the program to `SimCADO` that could easily be expanded to simulation software of other telescopes.

```
objectgenerator.AstObject(struct='ellipsoid', N_stars=0, M_tot_init=0,
   age=None, metal=None, rel_num=None, distance=0, d_type='l', imf_par=None,
   sf_hist=None, extinct=0, incl=None, r_dist=None, r_dist_par=None,
   ellipse_axes=None, spiral_arms=0, spiral_bulge=0, spiral_bar=0, )
   compact=False, cp_mode='num', mag_lim=None)
```

"Generates the astronomical object and contains all the information about the object. Also functions that can be performed on the object are defined here. "

```
objectgenerator.AstObject.CurrentMasses(realistic_remnants=True)
```

"Gives the current masses of the stars in Msun. Uses isochrone files and the given initial masses of the stars. Stars should not have a lower initial mass than the lowest mass in the isochrone file. "

```
objectgenerator.AstObject.LogLuminosities(realistic_remnants=True)
```

"Gives the logarithm of the luminosity of the stars in Lsun. Uses isochrone files and the given initial masses of the stars. realistic_remnants gives estimates for remnant luminosities. Set False to save time. Stars should not have a lower initial mass than the lowest mass in the isochrone file. "

```
objectgenerator.AstObject.LogTemperatures(realistic_remnants=True)
```

"Gives the logarithm of the effective temperature of the stars in K. Uses isochrone files and the given initial masses of the stars. realistic_remnants gives estimates for remnant temperatures. Set False to save time. Stars should not have a lower initial mass than the lowest mass in the isochrone file. "

```
objectgenerator.AstObject.AbsoluteMagnitudes(filter='all')
```

"Gives the absolute magnitudes of the stars. Uses isochrone files and the given initial masses of the stars. Stars should not have a lower initial mass than the lowest mass in the isochrone file. "

```
objectgenerator.AstObject.ApparentMagnitudes(filter='all')
```

"Compute the apparent magnitude from the absolute magnitude and the individual distances (in pc!). The filter can be specified. 'all' will give all the filters. "

```
objectgenerator.AstObject.SpectralTypes(realistic_remnants=True)
```

"Gives the spectral types (as indices, to conserve memory) for the stars and the corresponding spectral type names. Uses isochrone files and the given initial masses of the stars. Stars should not have a lower initial mass than the lowest mass in the isochrone file. "

```
objectgenerator.AstObject.Remnants()
```

"Gives the indices of the positions of remnants (not white dwarfs) as a boolean array. (WD should be handled with the right isochrone files, but NS/BHs are not) "

`objectgenerator.AstObject.TotalCurrentMass()`
"Returns the total current mass in Msun."

`objectgenerator.AstObject.TotalLuminosity()`
"Returns log of the total luminosity in Lsun."

`objectgenerator.AstObject.CoordsArcsec()`
"Returns coordinates converted to arcseconds (from pc)."

`objectgenerator.AstObject.Radii(unit='pc', spher=False)`
"Returns the radial coordinate of the stars (spherical or cylindrical) in pc/as."

`objectgenerator.AstObject.HalfMassRadius(spher=False)`
"Returns the (spherical or cylindrical) half mass radius in pc."

`objectgenerator.AstObject.HalfLumRadius(spher=False)`
"Returns the (spherical or cylindrical) half luminosity radius in pc."

```
objectgenerator.AstObject.Plot2D(title='Scatter', xlabel='x', ylabel='y',
   axes='xy', colour='blue', filter='V', theme='dark1')
```
"Make a plot of the object positions in two dimensions Set colour to 'temperature' for a temperature representation. Set filter to None to avoid markers scaling in size with magnitude. Set theme to 'dark1' for a fancy dark plot, 'dark2' for a less fancy but saveable dark plot, 'fits' for a plot that resembles a .fits image, and None for normal light colours. "

```
objectgenerator.AstObject.Plot3D(title='Scatter', xlabel='x', ylabel='y',
   axes='xy', colour='blue', filter='V', theme='dark1')
```
"Make a plot of the object positions in three dimensions. Set colour to 'temperature' for a temperature representation. Set filter to None to avoid markers scaling in size with magnitude. Set theme to 'dark1' for a fancy dark plot, 'dark2' for a less fancy but saveable dark plot, and None for normal light colours. "

```
objectgenerator.AstObject.PlotHRD(title='HRD', colour='temperature',
   theme='dark1')
```
"Make a plot of stars in an HR diagram. Set colour to 'temperature' for a temperature representation. Set theme to 'dark1' for a fancy dark plot, 'dark2' for a less fancy but saveable dark plot, and None for normal light colours. "

```
objectgenerator.AstObject.PlotCMD(x='B-V', y='V', title='CMD',
   colour='blue', theme='dark1')
```
"Make a plot of the stars in a CMD Set x and y to the colour and magnitude to be used (x needs format 'A-B') Set colour to 'temperature' for a temperature representation. "

`objectgenerator.AstObject.SaveTo(filename)`
"Saves the class to a file."

`objectgenerator.AstObject.LoadFrom(filename)`
"Loads the class from a file."

`objectgenerator.Ellipsoid(N_stars, dist_type='Normal_r', axes=None, **param)`
"Make a spherical distribution of stars using the given 1d radial distribution type.
If axes-scales are given [a*x, b*y, c*z], then shape is elliptical instead of spherical.

Takes additional parameters for the r-distribution function. ”

```
objectgenerator.StarMasses(N_stars=0, M_tot=0, imf=[0.08, 150])
```
”Generate masses using the Initial Mass Function.
Either number of stars or total mass should be given.
imf defines the lower and upper bound to the masses generated in the IMF.
Also gives the difference between the total generated mass and the input mass
(or estimated mass when using N_stars). ”

```
objectgenerator.OpenIsochrone(age, Z, columns='all')
```
”Opens the isochrone file and gives the relevant columns.
columns can be: 'all', 'mini', 'mcur', 'lum', 'temp', 'mag' (and 'filters')
age can be either linear or logarithmic input. ”

```
visualizer.DistHist(dist, title='Histogram', xlabel='parameter',
   ylabel='relative number', step=True, type='linear', labels=[])
```
”Display the histogram for some distribution. Can handle multiple distributions. type can be linear, linlog, loglin (x,y) and loglog. Step=False will give a plot instead. ”

```
fitshandler.GetData(filename, index=0)
```
”Returns the requested data. [NOTE: data[1, 4] gives pixel value at x=5, y=2.] Optional arg: HDUlist index. ”

```
fitshandler.PlotFits(filename, index=0, colours='gray', scale='lin',
   grid=False, chip='single')
```
”Displays the image in a fits file. Optional args: HDUlist index, colours.
Can also take image objects directly.
scale can be set to 'lin', 'sqrt', and 'log'
chip='single': plots single data array at given index.
='full': expects data in index 1-9 and combines it. ”

```
fitshandler.SaveFitsPlot(filename, index=0, colours='gray', scale='lin',
   grid=False, chip='single')
```
”Saves the plotted image in a fits file. Optional args: HDUlist index, colours.
Can also take image objects directly.
scale can be set to 'lin', 'sqrt', and 'log'
chip='single': plots single data array at given index.
='full': expects data in index 1-9 and combines it. ”

```
imagegenerator.MakeSource(astobj, filter='V')
```
”Makes a SimCADO Source object from an AstObj. filter determines what magnitudes are used (corresponding to that filter). ”

```
imagegenerator.MakeImage(src, exposure=60, NDIT=1, view='wide',
   chip='centre', filter='V', ao_mode='scao',
   filename=default_image_file_name, internals=None, return_int=False)
```
”Make the image with SimCADO.
exposure = time in seconds, NDIT = number of exposures taken.
view = mode = ['wide', 'zoom']: fov 53 arcsec (4 mas/pixel) or 16 (1.5 mas/pixel)
chip = detector_layout = ['small', 'centre', 'full']:
1024x1024 pix, one whole detector (4096x4096 pix) or full array of 9 detectors
filter = the filter used in the 'observation'

ao_mode = PSF file used [scao, ltao, (mcao not available yet)] ”

`constructor.DynamicConstruct()`
”Dynamically give parameters for construction via this interactive function.”

`imager.DynamicImage(astobj=None, name=None)`
”Dynamically give parameters for imaging via this interactive function.”

Additional notes

Of course there are many many more functions that make the functions above work, but these are in principle not necessary for the end user and might never be seen. The `constructor.py` and `imager.py` modules enable usage through the command line; the above two 'dynamic' functions can also be called that way (for example: `<python3 constructor -inter>` will start the interactive construction mode). There is an entire module containing distribution functions for certain variables like the mass and spatial positions. More of these can be added by the user where needed for personal use. New code can always be submitted via a pull request on the GitHub page (github.com/LucIJspeert/StarPopSim). Alternatively, requests for new functionality or fixes can also be communicated through the GitHub page.

B Cluster input properties

Table 3: The individual cluster input parameters. All clusters have an age of 10 Gyr and use the radial profile by King (see 4.1.2). The given properties are the total mass (M) of the cluster, the distance (D) to it and its half-light radius (hlr).

#	M ($M_\odot$)	D (Mpc)	hlr (pc)	#	M ($M_\odot$)	D (Mpc)	hlr (pc)
1	10^5	0.8	1	76	$6.1 \cdot 10^6$	0.8	1
2	10^5	0.8	6	77	$6.1 \cdot 10^6$	0.8	6
3	10^5	0.8	11	78	$6.1 \cdot 10^6$	0.8	11
4	10^5	0.8	16	79	$6.1 \cdot 10^6$	0.8	16
5	10^5	0.8	21	80	$6.1 \cdot 10^6$	0.8	21
6	10^5	4.0	1	81	$6.1 \cdot 10^6$	4.0	1
7	10^5	4.0	6	82	$6.1 \cdot 10^6$	4.0	6
8	10^5	4.0	11	83	$6.1 \cdot 10^6$	4.0	11
9	10^5	4.0	16	84	$6.1 \cdot 10^6$	4.0	16
10	10^5	4.0	21	85	$6.1 \cdot 10^6$	4.0	21
11	10^5	7.2	1	86	$6.1 \cdot 10^6$	7.2	1
12	10^5	7.2	6	87	$6.1 \cdot 10^6$	7.2	6
13	10^5	7.2	11	88	$6.1 \cdot 10^6$	7.2	11
14	10^5	7.2	16	89	$6.1 \cdot 10^6$	7.2	16
15	10^5	7.2	21	90	$6.1 \cdot 10^6$	7.2	21
16	10^5	10.4	1	91	$6.1 \cdot 10^6$	10.4	1
17	10^5	10.4	6	92	$6.1 \cdot 10^6$	10.4	6
18	10^5	10.4	11	93	$6.1 \cdot 10^6$	10.4	11
19	10^5	10.4	16	94	$6.1 \cdot 10^6$	10.4	16
20	10^5	10.4	21	95	$6.1 \cdot 10^6$	10.4	21
21	10^5	13.6	1	96	$6.1 \cdot 10^6$	13.6	1
22	10^5	13.6	6	97	$6.1 \cdot 10^6$	13.6	6
23	10^5	13.6	11	98	$6.1 \cdot 10^6$	13.6	11
24	10^5	13.6	16	99	$6.1 \cdot 10^6$	13.6	16
25	10^5	13.6	21	100	$6.1 \cdot 10^6$	13.6	21
26	$2.1 \cdot 10^6$	0.8	1	101	$8.1 \cdot 10^6$	0.8	1
27	$2.1 \cdot 10^6$	0.8	6	102	$8.1 \cdot 10^6$	0.8	6
28	$2.1 \cdot 10^6$	0.8	11	103	$8.1 \cdot 10^6$	0.8	11
29	$2.1 \cdot 10^6$	0.8	16	104	$8.1 \cdot 10^6$	0.8	16
30	$2.1 \cdot 10^6$	0.8	21	105	$8.1 \cdot 10^6$	0.8	21
31	$2.1 \cdot 10^6$	4.0	1	106	$8.1 \cdot 10^6$	4.0	1
32	$2.1 \cdot 10^6$	4.0	6	107	$8.1 \cdot 10^6$	4.0	6
33	$2.1 \cdot 10^6$	4.0	11	108	$8.1 \cdot 10^6$	4.0	11
34	$2.1 \cdot 10^6$	4.0	16	109	$8.1 \cdot 10^6$	4.0	16
35	$2.1 \cdot 10^6$	4.0	21	110	$8.1 \cdot 10^6$	4.0	21
36	$2.1 \cdot 10^6$	7.2	1	111	$8.1 \cdot 10^6$	7.2	1
37	$2.1 \cdot 10^6$	7.2	6	112	$8.1 \cdot 10^6$	7.2	6
38	$2.1 \cdot 10^6$	7.2	11	113	$8.1 \cdot 10^6$	7.2	11
39	$2.1 \cdot 10^6$	7.2	16	114	$8.1 \cdot 10^6$	7.2	16

Continued on next page

Table 3 – *Continued from previous page*

#	M ($M_\odot$)	D (Mpc)	hlr (pc)	#	M ($M_\odot$)	D (Mpc)	hlr (pc)
40	$2.1 \cdot 10^6$	7.2	21	115	$8.1 \cdot 10^6$	7.2	21
41	$2.1 \cdot 10^6$	10.4	1	116	$8.1 \cdot 10^6$	10.4	1
42	$2.1 \cdot 10^6$	10.4	6	117	$8.1 \cdot 10^6$	10.4	6
43	$2.1 \cdot 10^6$	10.4	11	118	$8.1 \cdot 10^6$	10.4	11
44	$2.1 \cdot 10^6$	10.4	16	119	$8.1 \cdot 10^6$	10.4	16
45	$2.1 \cdot 10^6$	10.4	21	120	$8.1 \cdot 10^6$	10.4	21
46	$2.1 \cdot 10^6$	13.6	1	121	$8.1 \cdot 10^6$	13.6	1
47	$2.1 \cdot 10^6$	13.6	6	122	$8.1 \cdot 10^6$	13.6	6
48	$2.1 \cdot 10^6$	13.6	11	123	$8.1 \cdot 10^6$	13.6	11
49	$2.1 \cdot 10^6$	13.6	16	124	$8.1 \cdot 10^6$	13.6	16
50	$2.1 \cdot 10^6$	13.6	21	125	$8.1 \cdot 10^6$	13.6	21
51	$4.1 \cdot 10^6$	0.8	1	126	$1.01 \cdot 10^7$	0.8	1
52	$4.1 \cdot 10^6$	0.8	6	127	$1.01 \cdot 10^7$	0.8	6
53	$4.1 \cdot 10^6$	0.8	11	128	$1.01 \cdot 10^7$	0.8	11
54	$4.1 \cdot 10^6$	0.8	16	129	$1.01 \cdot 10^7$	0.8	16
55	$4.1 \cdot 10^6$	0.8	21	130	$1.01 \cdot 10^7$	0.8	21
56	$4.1 \cdot 10^6$	4.0	1	131	$1.01 \cdot 10^7$	4.0	1
57	$4.1 \cdot 10^6$	4.0	6	132	$1.01 \cdot 10^7$	4.0	6
58	$4.1 \cdot 10^6$	4.0	11	133	$1.01 \cdot 10^7$	4.0	11
59	$4.1 \cdot 10^6$	4.0	16	134	$1.01 \cdot 10^7$	4.0	16
60	$4.1 \cdot 10^6$	4.0	21	135	$1.01 \cdot 10^7$	4.0	21
61	$4.1 \cdot 10^6$	7.2	1	136	$1.01 \cdot 10^7$	7.2	1
62	$4.1 \cdot 10^6$	7.2	6	137	$1.01 \cdot 10^7$	7.2	6
63	$4.1 \cdot 10^6$	7.2	11	138	$1.01 \cdot 10^7$	7.2	11
64	$4.1 \cdot 10^6$	7.2	16	139	$1.01 \cdot 10^7$	7.2	16
65	$4.1 \cdot 10^6$	7.2	21	140	$1.01 \cdot 10^7$	7.2	21
66	$4.1 \cdot 10^6$	10.4	1	141	$1.01 \cdot 10^7$	10.4	1
67	$4.1 \cdot 10^6$	10.4	6	142	$1.01 \cdot 10^7$	10.4	6
68	$4.1 \cdot 10^6$	10.4	11	143	$1.01 \cdot 10^7$	10.4	11
69	$4.1 \cdot 10^6$	10.4	16	144	$1.01 \cdot 10^7$	10.4	16
70	$4.1 \cdot 10^6$	10.4	21	145	$1.01 \cdot 10^7$	10.4	21
71	$4.1 \cdot 10^6$	13.6	1	146	$1.01 \cdot 10^7$	13.6	1
72	$4.1 \cdot 10^6$	13.6	6	147	$1.01 \cdot 10^7$	13.6	6
73	$4.1 \cdot 10^6$	13.6	11	148	$1.01 \cdot 10^7$	13.6	11
74	$4.1 \cdot 10^6$	13.6	16	149	$1.01 \cdot 10^7$	13.6	16
75	$4.1 \cdot 10^6$	13.6	21	150	$1.01 \cdot 10^7$	13.6	21

C IRAF parameters

Here all the altered parameters are listed that were used in the image analysis. Wherever the following options occur, they are set to the values in the list below.

- cache = yes
- verify = no
- update = no
- verbose = yes
- graphic = stdgraph
- display = stdimage

This is predominantly done to ensure `IRAF` stays consistent and does not behave unexpectedly.[33] The first file to be altered, `datapars`, holds some specific quantities for the data.

- scale = 1
- fwhmpsf = 5
- sigma = 238
- readnoise = 4

Next, the file `findpars` is updated. It contains the parameters that are used in the star finding algorithm, and will determine how well the stars are distinguished from each other and from the noise.

- thresho = 5
- nsigma = 2
- sharplo = 0
- sharphi = 20
- roundlo = -10
- roundhi = 10

The code has to be told what files to use and make by setting the `daofind` keywords:

- image = <your PSF image file>
- output = <found stars file>

Make sure the names make sense so that they can always be recognised afterwards. Now `daofind` is run and the next parameters to be changed are for the aperture photometry. In `photpars`:

- apertures = 6

In `centerpars`:

- calgorithm = centroid
- cbox = 5.0

In `fitskypars`:

- salgorithm = mode
- annulus = 7
- dannulus = 10

[33] It has a tendency not to do what the user wants when not specifically told otherwise.

And finally, before running `phot`, its parameters are set:

- image = <your PSF image file>
- coords = <found stars file>
- output = <aperture photometry file>

We will now have our first estimates of the flux that are needed in order to make the PSF models. The last task before this can be done will select the PSF stars to be used. Since we have an image *only* containing PSF stars, it is fairly safe to let the software figure this out itself and check the output file afterwards. This saves the time of selecting every star by hand, which can require some patience. The file `daopars` is edited:

- function = auto
- psfrad = 16
- fitrad = 5
- fitsky = yes
- sannul = 5
- wsannul = 16
- flaterr = 0
- maxgroup = 100

As well as `pstselect`:

- image = <your PSF image file>
- photfile = <aperture photometry file>
- pstfile = <PSF star selection file>
- maxnpsf = 36

Whereafter `pstselect` is run. Finally, to finish the PSF modelling, the task `psf` is run with the following parameters:

- image = <your PSF image file>
- photfile = <aperture photometry file>
- pstfile = <PSF star selection file>
- psfimage = <PSF model image>
- opstfile = <file for the used stars>
- groupfile = <file for the star groups made>

To be able to plot a nice image of the PSF model, one can use the task `seepsf`. The model itself looks rather odd, more representative of a negative picture. It is a good idea to have a look at the PSF model to check for deviations from the expected result. This can be done in the program `DS9`, which can work together with certain `IRAF` tasks.