

Ingebrigtsen, Simon  
Remøy, Sondre Westbø  
Runde, Simon Nedreberg  
Austnes, Einar Leite

## CosmoAI

A study of gravitational lensing through  
simulation and machine learning

Bachelor's thesis in Electrical Engineering - Automation and  
Robotics

Supervisor: Schaathun, Hans Georg  
Co-supervisor: Normann, Ben David

May 2022



Ingebrigtsen, Simon  
Remøy, Sondre Westbø  
Runde, Simon Nedreberg  
Austnes, Einar Leite

## **CosmoAI**

A study of gravitational lensing through simulation  
and machine learning

Bachelor's thesis in Electrical Engineering - Automation and Robotics  
Supervisor: Schaathun, Hans Georg  
Co-supervisor: Normann, Ben David  
May 2022

Norwegian University of Science and Technology  
Faculty of Information Technology and Electrical Engineering  
Department of ICT and Natural Sciences



**NTNU**

Kunnskap for en bedre verden





# Preface

This bachelor report is written by four students from “Automation and robotics” at NTNU Ålesund. The goal of this project is to serve as the first step towards a bigger, long term research endeavour planned to be conducted at NTNU.

The challenging nature of the problems and being part of a bigger research project is what intrigued us to choose this assignment. We were all eager to make a contribution to the field of cosmology and felt our background could bring a new perspective to the problems.

We would like to thank our supervisors, Hans Georg Schaathun and Ben David Normann, for providing crucial insight, critical thinking and motivating conversations. This project could not have been completed without their wisdom and guidance.

# Summary

Gravitational lensing serves as a tool for locating and mapping dark matter across the universe. Through the Roulette formalism it is hoped that the information hidden in the lenses can be deciphered with a common mathematical framework. The process of unveiling the lens' information could be more efficient and accurate through the use of machine learning. Cracking the machine learning problem could cause a paradigm shift in how we map dark matter. Tools capable of producing the large sets of data that is needed to train a machine learning model exist, however, these are not open source, nor able to simulate the Roulette formalism. In this thesis we have created an open source simulator able to produce these datasets of both traditional gravitational lensing and Roulette, in large quantities. This simulator enables the creation of hundreds of thousands of synthetic images. The simulator has also been made into an application with a user friendly interface. The application is easy to install and require no prior experience with programming to set up and use. Large datasets of distorted images with random lensing parameters were used for training a neural network with the goal of identifying the parameters. Several network architectures were tested, but a modified version of the AlexNet stood out as the most suited. The network was able to accurately predict the lens' parameters in all of the validation images.

# Contents

Preface . . . . .	i
Summary . . . . .	ii
Acronyms . . . . .	2
<b>1 Introductions</b>	<b>7</b>
1.1 Background . . . . .	7
1.2 Problem formulation . . . . .	8
1.3 Report structure . . . . .	8
<b>2 Theoretical basis</b>	<b>10</b>
2.1 Cosmology . . . . .	10
2.1.1 Gravitational lensing . . . . .	10
2.1.2 Einstein radius . . . . .	10
2.1.3 Strong lensing . . . . .	11
2.1.4 Weak lensing . . . . .	11
2.1.5 Roulette lensing . . . . .	12
2.1.6 Thin lens in the weak-field and flat-sky approximations . . . . .	12
2.1.7 Cosmological redshift . . . . .	13
2.1.8 Formulas . . . . .	13
2.2 Machine learning . . . . .	16
2.2.1 Artificial Neural Networks . . . . .	16
2.2.2 Convolutional Neural Network . . . . .	17
2.2.3 Previous work with ML and gravitational lensing . . . . .	17
<b>3 Method</b>	<b>18</b>
3.1 Project organisation . . . . .	18
3.2 Materials . . . . .	19
3.3 Software development . . . . .	20
3.3.1 Choice of software and libraries . . . . .	20
3.3.2 Simulator . . . . .	22
3.3.3 Machine learning . . . . .	22
<b>4 Results - Simulator</b>	<b>26</b>
4.1 Deriving equations for apparent position and R . . . . .	26
4.2 Exact point mass model . . . . .	27

4.2.1	First prototype	27
4.2.2	Second prototype - Simplified but correct implementation in C++	28
4.2.3	Third prototype - Including $\chi_L$ and $\chi_S$	29
4.2.4	Fourth prototype - Complete exact point mass simulator	30
4.2.5	Multithreading	31
4.3	Graphical User Interface (GUI)	32
4.3.1	Implementation of exact point mass simulator in GUI	35
4.3.2	Experimenting with different sources	35
4.4	Singular isothermal sphere model (SIS)	38
4.4.1	Analysing the equations	38
4.4.2	Generating expressions for the amplitudes	39
4.4.3	C++ and SymEngine	39
4.4.4	Implementation of SIS-model in simulator	41
4.5	Finite point mass model	43
<b>5</b>	<b>Results - Machine Learning</b>	<b>45</b>
5.1	Machine learning tasks	45
5.2	$\chi$ vs. Einstein radius	46
5.3	Generate and import data	46
5.4	Training models	47
5.4.1	First iteration: Three parameters	48
5.4.2	Second iteration: Increasing image resolution	48
5.4.3	Third iteration: AlexNet	48
5.4.4	Fourth iteration: Five parameters	49
5.4.5	Fifth iteration: Improving training data	49
5.4.6	Iteration six: Assuming that $\chi$ is known	50
5.4.7	Iteration seven: Multi modal network	51
5.4.8	Looking at the statistics	51
5.5	Inception3	53
5.6	Scheduler and Learning rate	55
<b>6</b>	<b>Discussion</b>	<b>56</b>
6.1	Project organisation	56
6.2	Prerequisites and learning goals	56
6.3	Simulator	58
6.3.1	Point mass model	58
6.3.2	Spherical model (SIS)	60
6.4	Machine learning	61
6.4.1	Possible to predict both $\chi$ and $R_E$ ?	61
6.4.2	Building networks from scratch	61
6.4.3	AlexNet	63
6.4.4	Inception3	63
6.4.5	Other architectures	63
6.4.6	Image resolution	63
6.4.7	Extending to Roulette images and real world applications	64

6.5	GUI . . . . .	64
6.6	Client feedback . . . . .	65
<b>7</b>	<b>Conclusions</b>	<b>66</b>
7.1	Simulator . . . . .	66
7.2	Machine learning . . . . .	66
7.3	Open problems . . . . .	67
	<b>Bibliography</b>	<b>68</b>
<b>A</b>	<b>Gravitational lensing notes by B.D. Normann</b>	<b>70</b>
<b>B</b>	<b>Pre-project report</b>	<b>91</b>
<b>C</b>	<b>Progress reports</b>	<b>103</b>
C.1	Progress report 02.02.2022 . . . . .	103
C.2	Progress report 02.03.2022 . . . . .	106
C.3	Progress report 25.03.2022 . . . . .	109
C.4	Progress report 06.04.2022 . . . . .	111
C.5	Progress report 27.04.2022 . . . . .	114
C.6	Progress report 11.05.2022 . . . . .	117
<b>D</b>	<b>Meeting minutes</b>	<b>120</b>
<b>E</b>	<b>Hour list</b>	<b>144</b>
E.1	Sondre . . . . .	144
E.2	Einar . . . . .	149
E.3	Simon I. . . . .	153
E.4	Simon R. . . . .	157
<b>F</b>	<b>Source code</b>	<b>161</b>
F1	Simulator . . . . .	161
F2	GUI . . . . .	179
F3	Machine Learning . . . . .	213

## Terminology

- Weight** Multiplies the activation from one neuron before feeding into next neuron.
- Bias** A term added to the weighted sum of previous layer before feeding into next neuron.
- Backpropagation** Algorithm used to train neural net by adjusting weights and biases
- Convolution** Discrete mathematical operation on two function that produces a third function.
- Kernel** Discrete filter used in convolution.
- Pooling** Down sampling inputs in a convolutional neural network.
- Loss** Used as a indication of the performance of a neural network.

## Notation

- $\kappa$  Dimensionless surface mass density (or convergence) of the lens
- $\psi$  Lens potential
- $\alpha&\beta$  The even roulette amplitudes
- $\chi_L$  Distance from observer to lens plane
- $\chi_S$  Distance from observer to source plane
- $R_E$  Einstein radius, measure of the strength of the lens
- $\Gamma$  A different measure of the strength of the lens
- $(r, \theta)$  Polar coordinates in the lens plane
- $(x', y')$  Coordinates in the source plane
- $R$  Distance between lens and distorted image in the lens plane
- $P_{\text{act}}$  Actual position of source
- $P_{\text{app}}$  Apparent position of source

## Abbreviations

- GL** Gravitational lensing
- GR** General relativity
- ML** Machine learning
- ANN** Artificial Neural Network
- CNN** Convolutional Neural Network
- SIS** Singular Isothermal Sphere

**GUI** Graphical User Interface

**API** Application Programming Interface

# List of Figures

2.1	Image of the lensed quasar HE0435-1223 from the Hubble Telescope. A galaxy between the quasar and observer creates four almost evenly distributed images of the distant quasar around it. Credit: ESA/Hubble, NASA, Suyu et al. .	11
2.2	The relatively nearby galaxy cluster MACSJ0138.0-2155 has lensed a significantly more distant quiescent galaxy. Credit: ESA/Hubble NASA, A. Newman, M. Akhshik, K. Whitaker . . . . .	12
2.3	Illustration of the thin lens approximation. . . . .	13
2.4	Illustration of the thin lens approximation. . . . .	14
3.1	Illustration of work flow in agile development. . . . .	19
3.2	System diagram . . . . .	21
3.3	AlexNet architecture. . . . .	23
3.4	IDUN flow . . . . .	25
4.1	Screenshot of the first prototype . . . . .	28
4.2	Algorithm for exact point mass model, second prototype. . . . .	28
4.3	Second prototype of the simulator (undistorted left, distorted right) . . . . .	29
4.4	Simulator with $\frac{\chi_L}{\chi_S}$ implemented . . . . .	30
4.5	Artefacts from mapping from outside source image . . . . .	30
4.6	Complete simulator, exact point mass model . . . . .	31
4.7	Each thread is assigned its own range of rows. . . . .	32
4.8	Overview of GUI with numbered elements. . . . .	33
4.9	Overview of GUI menus with numbered elements. . . . .	33
4.10	Implementation of simulator in Qt. . . . .	36
4.11	Image of the Tintin rocket, distorted by the exact point mass model . . . . .	37
4.12	Image of the Tintin rocket, distorted by the finite point mass model . . . . .	37
4.13	Plot of $\frac{r^m}{m!}$ with $r = 10$ . . . . .	38
4.14	Algorithm for generating expressions along the diagonal from (0,1) to (n, n+1) . . . . .	40
4.15	Algorithm for generating expressions from $(m_{start}, s_{start})$ . . . . .	40
4.16	Expressions generated by <i>symPy</i> . . . . .	41
4.17	Algorithm for mapping from undistorted to distorted image . . . . .	41
4.18	SIS simulator $n = 6$ . . . . .	42
4.19	SIS simulator $n = 10$ . . . . .	42
4.20	SIS simulator $n = 14$ . . . . .	42



4.21	Algorithm for finite sum point mass model . . . . .	43
4.22	Simulator, finite point mass model . . . . .	44
5.1	Simulator output with $(R_E, \chi) = (30, 0.3)$ , $(80, 0.3)$ and $(80, 0.8)$ . . . . .	46
5.2	Program flow . . . . .	47
5.3	Images generated with lensing parameters shown in the filenames . . . . .	48
5.4	Artefacts at low resolutions . . . . .	49
5.5	Hard to predict . . . . .	50
5.6	Some of the worst losses with corresponding parameters and output . . . . .	51
5.7	$\chi$ is inserted directly into the network . . . . .	52
5.8	Distribution of the errors of the five parameters . . . . .	53
5.9	Distribution of the errors of the four parameters . . . . .	54
5.10	Distribution of the errors of the four parameters using multi modal network . . . . .	54
6.1	Simulator, finite point mass model . . . . .	59
6.2	Illustrations of Roulette lensing as presented in Clarksons work. Credit: C. Clarkson [1] . . . . .	59
6.3	Comparison of the output from the exact and the finite point mass model summing up to $m = 100$ . . . . .	60
6.4	$(\chi, R_E) = (30, 50)$ and $(40, 60)$ . . . . .	62
6.5	Possible "best guess" distribution of $\chi$ and $R_E$ if the network is able to predict the 'distance' between variables perfectly . . . . .	62

# List of Tables

3.1	Table of software used during development. . . . .	19
3.2	Table of the external libraries used . . . . .	20
4.1	Table explaining labels in 4.8 and 4.9 . . . . .	34
4.2	Example of alphas for $n = 5$ . . . . .	39

# Chapter 1

## Introduction

Today we collect information from the night sky in many different ways. A theoretical framework is the foundation for understanding collected information. By developing the framework, more information can be deciphered and in turn expand our knowledge of the universe. One such theoretical development came with the understanding that even light is effected by gravity, causing a lensing effect. This effect is known as *gravitational lensing*, a phenomenon rich with information. The theoretical framework used to interpret this information is divided in two regimes: *weak* and *strong* gravitational lensing. This separation is an artificial construct, in the sense that it is one of mathematics, and not physics. In 2016, C. Clarkson found a way of removing this divide by developing an algebraic framework that can model both weak and strong gravitational lensing, known as the *Roulette formalism*. Researchers at NTNU Ålesund seek to start a research project to further develop this framework. This bachelor thesis serves as a starting foundation for this future research.

### 1.1 Background

If we are to understand our universe, we must be able to map it. Everything we see around us is made of matter. This matter is observable because it reflects or emits light. This is “normal” matter, or more precisely baryonic matter, and mapping it is more or less straight forward. Surprisingly, all matter observed directly make up less than 5% of the observable universe. Roughly speaking, 68% is dark energy, and the remaining 27% is matter which does not reflect nor emit light [2]. This matter is called “dark matter”.

As dark matter is completely invisible, mapping it is challenging. Although, as with all matter, it does exert a pull proportional to its mass; gravity. When light passes a large object, the objects gravity will cause an alteration in the lights trajectory. This phenomenon is known as *gravitational lensing*. It allows us to observe the dark matter indirectly, and in turn, map it, giving us a better understanding of our universe.

## 1.2 Problem formulation

The problems tackled in this report are derived from two main goals. The first goal is to develop a tool for simulating different lensing models. By making this tool a simulator application, researchers can easily visualise mathematical lensing models without any programming knowledge, giving them more time to focus on their respective fields. A big challenge in developing this application is to make it user friendly while still including the functionalities a researcher might require.

The second goal is to use machine learning (ML) to analyse images with gravitational lensing. Gravitational lensing can be observed when light from distant galaxies pass through constellations of dark matter before reaching the observer. By training a ML-model, the goal is to identify the lens' mathematical model along with its parameters. Ideally this will be a functionality embedded in the application, allowing users to import images of distorted galaxies and receive an undistorted image of the galaxy along with lensing model and parameters. As the gravitational lensing effect is a function of the matter's mass, we can determine the the distribution of the mass based on the lens model. The distance can be determined from analysis of red shift. Thus, a map of the (indirectly) observed dark matter can be created.

Both these goals were quite loosely defined. There are several lens models to implement with increasing accuracy and hence increasing difficulty of implementing. The different lens models and suggestions of how to implement them can be found in appendix A. This appendix was written by our supervising cosmologist as an introduction to GL and Roulettes, and served as an assignment description. It was continuously revised throughout the project as the necessary equations were derived. There are also several goals to achieve in the machine learning part. Assumptions could be made regarding knowledge of the positions of the lens and light source, simplifying the task of predicting the remaining parameters. There is a clearly defined goal for the project as a whole, beyond the scope of this thesis, of being able to locate dark matter in the universe using the Roulette formalism and machine learning. Consequently, only partial results could be expected from this bachelor thesis.

## 1.3 Report structure

The rest of the report is structured as follows:

**Chapter 2 - Theoretical basis:** This chapter is divided in to two main parts; **cosmology** and **machine learning**. It serves as an introduction to the theoretical principles the following chapters are based on.

**Chapter 3 - Method:** Contains a description of the methodology for producing the simulator and the ML-model.

**Chapter 4 - Results - Simulator:** Contains all results and findings related to the simulator application.

**Chapter 5 - Results - Machine learning:** Contains all results and findings related to the machine learning.

**Chapter 6 - Discussion:** This chapter discuss and reflect around the project's organisation, as well as the results presented in chapter 4 and 5.

**Chapter 7 - Conclusions:** This chapter presents an overall conclusion to this thesis' findings along with the unanswered questions and open problems left for future research.

# Chapter 2

## Theoretical basis

This chapter serves as an introduction to the theoretical principles the following chapters are based on. It is divided in to two main parts, covering **cosmology** and **machine learning**.

### 2.1 Cosmology

This section will give a brief introduction to the concepts within the field of cosmology used in this report. It will strictly present the equations used in the implementation. For derivations and proofs, see Chris Clarkson's paper on Roulette lensing; *Roulettes: A weak lensing formalism for strong lensing - II. Derivation and analysis* [1].

#### 2.1.1 Gravitational lensing

Light rays are deflected when they propagate through a gravitational field. This phenomena is known as gravitational lensing, henceforth abbreviated to **GL**. It is even stated by P. Schneider, in the book *Gravitational Lensing: Strong, Weak and Micro*, that this gravitational light deflection occurs independently of the matters nature and state, making it equally sensitive to dark and luminous matter. The lensing phenomena is therefore an ideal tool for measuring the total mass of both dark and luminous astronomical bodies [3].

The theoretical algebraic framework to describe GL is typically divided in to two main regimes; *strong* and *weak* [3]. This divide is unfortunate because it is artificial, but has been necessary due to the lack of a common mathematical framework. In a paper published in 2016 by cosmologist C. Clarkson, a new perspective on GL was presented. Using the weak lensing formalism, Clarkson was able to accurately describe strongly lensed images through an expansion of the weak formalism. This gives a common theoretical framework and removes the artificial divide.

#### 2.1.2 Einstein radius

When an observer, a gravitational lens and a light source are all aligned, light from the source will be deflected around the lens and appear as a ring to the observer. This ring is called the Einstein



Figure 2.1: Image of the lensed quasar HE0435-1223 from the Hubble Telescope. A galaxy between the quasar and observer creates four almost evenly distributed images of the distant quasar around it. Credit: ESA/Hubble, NASA, Suyu et al.

ring, and the radius of said ring is the Einstein radius [4]. The Einstein radius represents the strength of the gravitational lens, as light from the source is deflected around it. The Einstein radius will be referred to as  $R_E$ .

### 2.1.3 Strong lensing

Strong gravitational lensing is in the literature described as  $\kappa > 1$ , where  $\kappa$  is the dimensionless surface mass density (or convergence) of the lens [3, p. 21]. A more tangible way of describing strong lenses are that they produce multiple images of point-like objects, i.e. the light from one object can be distorted so heavily that, to the observer, it appears as several separate objects. For more extended sources, the object can appear as an arc or even an Einstein ring [5]. Figure 2.1 shows an image of the lensed quasar HE0435-1223 where the lensing has caused the quasar to appear as four images around the lens. Figure 2.2 shows another case of strong lensing where clear arcs appears due to the lensing of galaxy cluster MACSJ0138.0-2155.

### 2.1.4 Weak lensing

When  $\kappa < 1$ , the lensing is categorised as *weak*. Opposed to *strong lensing*, this phenomenon does not cause multiple images, clear arcs or rings. As the name implies, the distortions are weak and more subtle. This causes the study of weak lensing to be a matter of statistical analysis, and the higher number of sources will yield more accurate results. Distant galaxies observed in the optical or near-infrared pass-band form the densest population of distant objects in the sky. Because of this, almost all weak lensing studies up to now are of these galaxies [3].

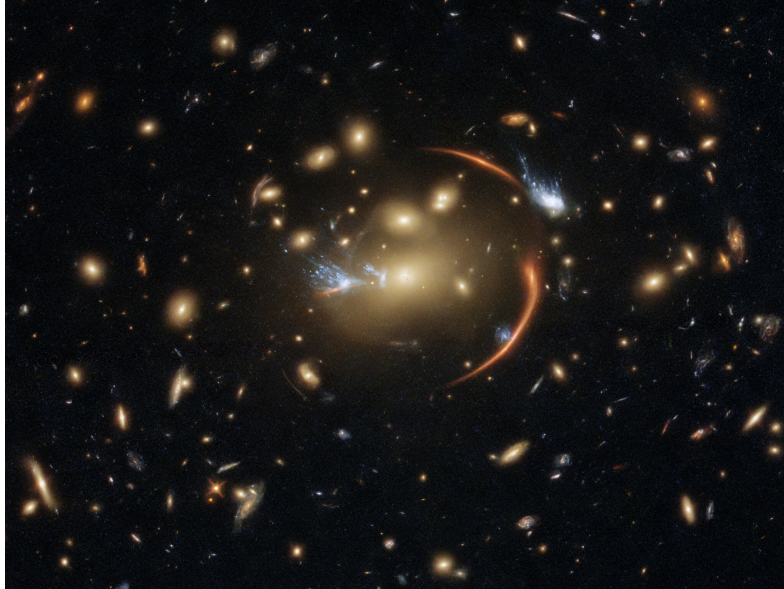


Figure 2.2: The relatively nearby galaxy cluster MACSJ0138.0-2155 has lensed a significantly more distant quiescent galaxy. Credit: ESA/Hubble NASA, A. Newman, M. Akhshik, K. Whitaker

### 2.1.5 Roulette lensing

It is unfortunate to divide a natural phenomena into two separate regimes with their own mathematical frameworks. The universe does not have such divides. In 2015, C. Clarkson at Queen Mary University of London found a way of removing this divide. By expanding the existing weak lensing formalism and taking higher-order effects into account, he found that strong gravitational lensing could be modelled. This method is called the Roulette formalism. The mathematical details of Roulette lensing is beyond the scope of this thesis, but can be found in C. Clarkson's work [1].

### 2.1.6 Thin lens in the weak-field and flat-sky approximations

By making some approximations, the mathematics can be simplified. These approximations is what we concern ourselves with for the rest of this report. The first approximation is that the lens can be thought of as optically thin. This removes the complicated physics involved with describing the lensing effect in thicker GL-lenses. Furthermore, we may consider the observer sphere as a flat sky because of the vast distance between observer and the lens. And finally, we consider the lens to be weak, in the sense that it is described by by the weak-field approximation. These approximations allows the setup to be drawn as shown in figure 2.3. The lens and source plane's coordinate systems can be defined as shown in figure 2.4. The inquisitive reader is encouraged to read appendix A for further elaboration and proof.

It is important to note that notation varies in the field of GL, in this report the notation is coherent with C. Clarkson's publications. In total, there are three coordinate systems at play, each with its own origin. Coordinates centred at the image in the lens plane are  $(x, y)$ , written as  $r(\cos\theta, \sin\theta)$  in polar coordinates, while the second coordinate system  $(X, Y)$  is centred at the



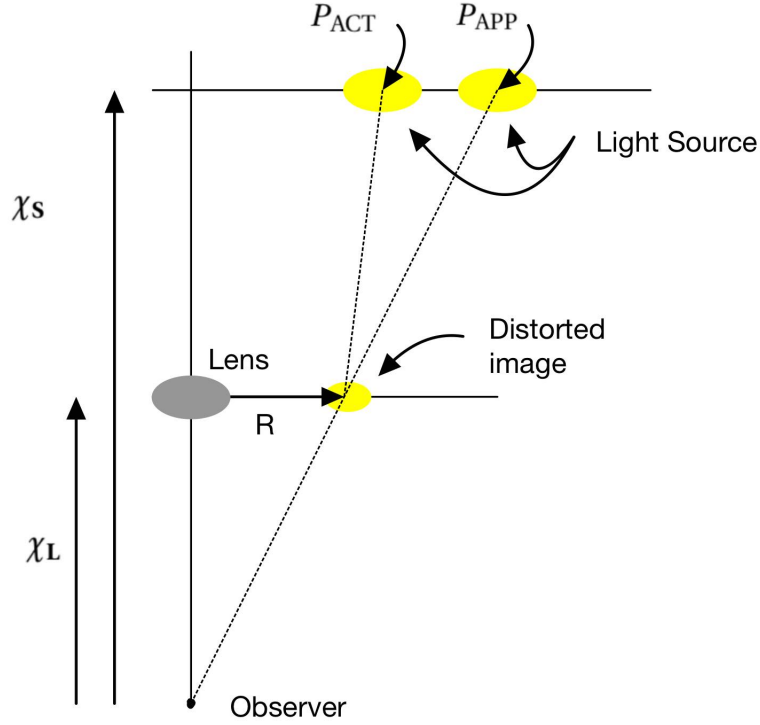


Figure 2.3: Illustration of the thin lens approximation.

lens, also in the lens plane. The third and final coordinate system is centred at the apparent source in the source plane are  $(x', y')$ .  $\chi_L$  is the distance from the observer to the lens plane.  $\chi_S$  is the distance from the observer to the source plane.  $R$  is the absolute value of the  $(X, Y)$  position.

### 2.1.7 Cosmological redshift

Light from distant sources in the universe will appear more red the further from the observer they are. This is due to the expansion of the universe stretching the light waves while they are travelling towards the observer. The reason it turns red is because red has the longest wavelength in the light spectrum [6]. The redshift can be used to measure distances to distant galaxies.

### 2.1.8 Formulas

#### Gravitational lens

The GL-simulator must map each pixel of a source image onto a distorted image. Using thin lens approximation this mapping from coordinates  $(r, \theta)$  in the lens plane to  $(x', y')$  in the source plane can be described by equation 2.1. This is equation (8), presented in appendix A.

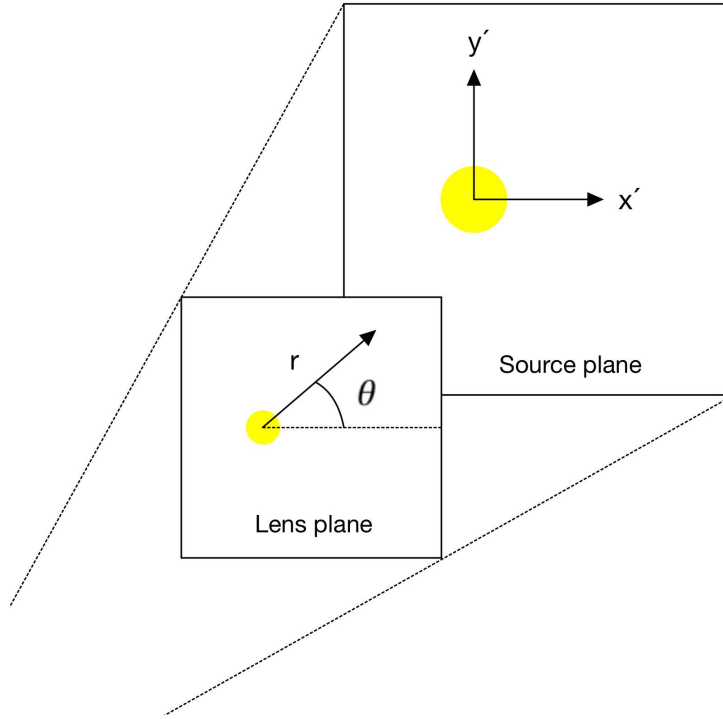


Figure 2.4: Illustration of the thin lens approximation.

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \sum_{m=1}^{\infty} \frac{r^m}{m!} \sum_{s=0}^{m+1} \frac{1 - (-1)^{m+s}}{4} \left( \alpha_s^m \begin{bmatrix} \cos(s-1)\theta & \cos(s+1)\theta \\ -\sin(s-1)\theta & \sin(s+1)\theta \end{bmatrix} + \beta_s^m \begin{bmatrix} \sin(s-1)\theta & \sin(s+1)\theta \\ \cos(s-1)\theta & -\cos(s+1)\theta \end{bmatrix} \right) \begin{bmatrix} C^+ \\ C^- \end{bmatrix} \quad (2.1)$$

with:

$$C^{\pm} = \pm \frac{s}{m+1} \quad (2.2)$$

Equation 2.1 is quite complicated due to the infinite sum,  $\alpha$ ,  $\beta$  and  $\psi$ .  $\alpha$  and  $\beta$  are functions containing sums of combinations of partial derivatives of the lens potential  $\psi$ . The lens potential's complexity will depend on the profile. However, simplifications can be made due to the approximations in section 2.1.6. The lens potential  $\psi$  for singular isothermal sphere profile is shown in equation 2.3. In this equation  $\Gamma$  is used to describe the strength of the lens, as an alternative to  $R_E$ .

$$\psi(R) = \frac{2\Gamma R}{\chi_L^2} \quad (2.3)$$

### Point mass lens

By assuming that the mass of the gravitational lens is concentrated in one point, a considerable amount of simplification can be done. Without loss of generality, one may assume that the

centre of mass of the source is located on the positive x-axis. The map from the source image to the distorted image can then be described by equation 2.4:

$$\frac{\chi_L}{\chi_S} \begin{bmatrix} x' \\ y' \end{bmatrix} = r \begin{bmatrix} \cos\theta \\ \sin\theta \end{bmatrix} - \frac{R_E^2}{R} \sum_{m=1}^{\infty} (-1)^m \left[ \frac{r}{R} \right]^m \begin{bmatrix} \cos(m\theta) \\ -\sin(m\theta) \end{bmatrix} \quad (2.4)$$

An approximation of the mapping can be calculated using the sum from  $m = 1$  to some finite number  $n$  with increasing accuracy as  $n \rightarrow \infty$ . This will produce a distorted image with  $(n + 1)$  spurious images and will only be valid within  $r < R$ . This model will be referred to as the *finite point mass model*. Using analytic continuation, the infinite sum can be calculated and extended outside this region. Using geometric series, it can be written in closed form as follows [1]:

$$\frac{\chi_L}{\chi_S} \begin{bmatrix} x' \\ y' \end{bmatrix} = r \begin{bmatrix} \cos\theta \\ \sin\theta \end{bmatrix} + \frac{R_E^2 r}{r^2 + R^2 + 2rR\cos(\theta)} \begin{bmatrix} \frac{r}{R} \cos(\theta) \\ -\sin(\theta) \end{bmatrix} \quad (2.5)$$

This is a standard result in the case of a point mass, and is not a result unique to the Roulette formalism. This model will be referred to as the exact point mass model. Furthermore, one may show that the apparent position is equal to the actual position shifted by:

$$\frac{\chi_S R_E^2}{\chi_L R} \quad (2.6)$$

### The Singular Isothermal Sphere (SIS)

The SIS model is somewhat similar to the point mass model as they both have circular symmetry. They both use the thin lens in the weak-field and flat-sky approximations. The SIS-model however, assumes that the mass of the GL is distributed in a spherical shape rather than concentrated at a single point. Unfortunately, this mean that the final simplifications that was used to get the simple equations 2.4 and 2.5 can not be used for the SIS model. In this case we can not avoid using the infinite sum equation 2.1. Luckily there are still some shortcuts that can be made to simplify things a bit.

A function for the amplitudes  $\alpha_1^0$  and  $\beta_1^0$  is given by the equation 2.7 and 2.8 from [7]

$$\alpha_1^0 = -\chi_L \frac{\partial \psi}{\partial X} \quad (2.7) \quad \beta_1^0 = -\chi_L \frac{\partial \psi}{\partial Y} \quad (2.8)$$

The amplitudes  $\alpha_s^m$  and  $\beta_s^m$  can be shown to be related through recursion relations [7]. The relations are:

$$\alpha_{s+1}^{m+1} = (C_+^+)^{m+1} \left( \frac{\partial \alpha_s^m}{\partial X} - \frac{\partial \beta_s^m}{\partial Y} \right) \quad (2.9)$$

$$\beta_{s+1}^{m+1} = (C_+^+)^{m+1} \left( \frac{\partial \beta_s^m}{\partial X} + \frac{\partial \alpha_s^m}{\partial Y} \right) \quad (2.10)$$

$$\alpha_{s-1}^{m+1} = (C_-^+)^{m+1} \left( \frac{\partial \alpha_s^m}{\partial X} + \frac{\partial \beta_s^m}{\partial Y} \right) \quad (2.11)$$

$$\beta_{s-1}^{m+1} = (C_-^-)^{m+1} \left( \frac{\partial \beta_s^m}{\partial X} - \frac{\partial \alpha_s^m}{\partial Y} \right) \quad (2.12)$$

with:

$$(C_+^+)^m = 2^{\delta_{0(s-1)}} \frac{m+1}{m+1+s} \chi_L \quad (C_-^+)^m = 2^{-\delta_{0s}} \frac{m+1}{m+1+s} \chi_L \quad (2.13)$$

The astute reader may notice that all amplitudes, when  $s + m$  is odd, can not be found through these relations. However, the contribution from these terms are equal to zero, as shown in equation 2.1. In other words, one can calculate all the amplitudes needed from the aforementioned relations.

Looking at equation 2.15 and equation 2.3, one can observe the relation between the two.

$$\psi(R) = \frac{R_E R}{\chi_L^2} \quad (2.15)$$

## 2.2 Machine learning

Machine learning is the science of making computers learn and improve automatically from experience. Machine learning is an important part of artificial intelligence and data science. It is widely used in fields such as computer vision and image recognition. A common solution for machine learning problems are artificial neural networks.

### 2.2.1 Artificial Neural Networks

An artificial neural network (ANN), sometimes simply called neural networks, is a data structure inspired by the neurons in a brain. Such data structures are well suited for identifying relations between information which is otherwise difficult to identify using mathematical formulations, e.g. image processing [8].

The network consists of connected data elements called neurons. The neuron can be viewed as a function which gives out a numeric value. This value is referred to as the neuron's activation. The neurons can send and receive numeric values between each other.

The neurons are in turn organised in layers. The first layer receives the inputs and the last layer gives out the network's output. Between these two layers, the network can have several "hidden" layers. The activations in one layer will determine the activations of the next layer. The number of neurons vary depending on the network's intended task, e.g. for processing images, each pixel in the image will be represented by a neuron in the input-layer. For an image with the resolution 400x400 this will result in 160 000 pixels.

The network is able to make decisions based on weights and biases. Each neuron will receive an activation value. When a neuron receive a value, the value is multiplied with positive or negative weights depending on the represented properties. A bias can be added to ensure the weighted sum needs to be above a certain threshold before a neuron is meaningfully active. Each neuron is connected with all of the neurons in the previous layer and the next layer. Each of these connections has its own weight, and every neuron it's own bias. These weights and biases get adjusted using an algorithm called back-propagation to minimise the error in the output. [9].

### 2.2.2 Convolutional Neural Network

For pattern recognition in images the Convolutional Neural Network (CNN) is commonly used. This particular version of ANN uses convolution between a set of kernels and the image. In CNN, the weights are the elements of the filter kernel and these are adjusted during backpropagation. The CNN takes advantage of the relations between pixels that are close to each other. This limits the massive amounts of neurons one would get using a traditional ANN and results in high performance with a relatively light network.

Three types of layers typically make up the CNN's architecture. Convolutional layers, pooling layers and fully-connected layers [10].

- The convolutional layers convolve the inputs given to it using a kernel of selected size, and gives the results to the next layer.
- Pooling layers reduce the data down by combining  $n \times n$  regions of the input data using either average or max values.
- Fully-connected layers does the same in CNNs as they do in traditional ANNs. They connect all neurons in one layer to all neurons in the next layer.

### 2.2.3 Previous work with ML and gravitational lensing

Previous work conducted on training networks to predict the parameters of gravitational lenses have done so using a strong lens approach [11] [12]. A 2017 paper implemented a classification neural network[13]. Unlike the previous work conducted, this project's end goal is to utilise the Roulette formalism to train the networks.

# Chapter 3

## Materials and methods

This chapter gives an outline of how the group went about to solve the two main problems presented in 1.2 - Problem formulation. The chapter starts with a detailed description of how the members distributed roles, responsibilities and main tasks. The rest of the chapter covers documentation, mathematical deduction, as well as the development process of both simulator and machine learning.

### 3.1 Project organisation

The group has been organised with a project leader and a secretary. The leader had the responsibility to organise the project, delegate responsibility, handle disputes, approve documentation and plan meetings. The secretary had the responsibility to take notes during meetings, write meeting reports and finish documentation. In practise, the group had a flat leading structure and major decisions were taken through discussion.

Before the project started, a pre-project report was created. The purpose of the pre-project report was to give clear outlines of the projects goals and how the group would organise to meet the goals. The report is presented in appendix B (*in Norwegian*). Along with the report, a written agreement of cooperation was signed by all group members.

In the development of both the simulator and the machine learning model, an agile development methodology was used. Both the group and the supervisors agreed that some of the research will be uncharted territory, resulting in a high degree of uncertainty regarding expected results. With this in consideration, a traditional plan with a clear end goal was not feasible, and the group decided the agile methodology was better suited. This method is based on an iterative development and is intended for rapid delivery of new software in smaller groups.

All unperformed work was placed in a *backlog* and the entire project period divided in to *sprints*. The sprints was supposed to be approximately two week work periods, but some deviations occurred. At the start of a sprint, the group chose tasks from the backlog and planned the execution of these tasks over the course of one sprint. During the sprint, the group had weekly, or sometimes daily, meetings to coordinate tasks and discuss problems. After each sprint, the group

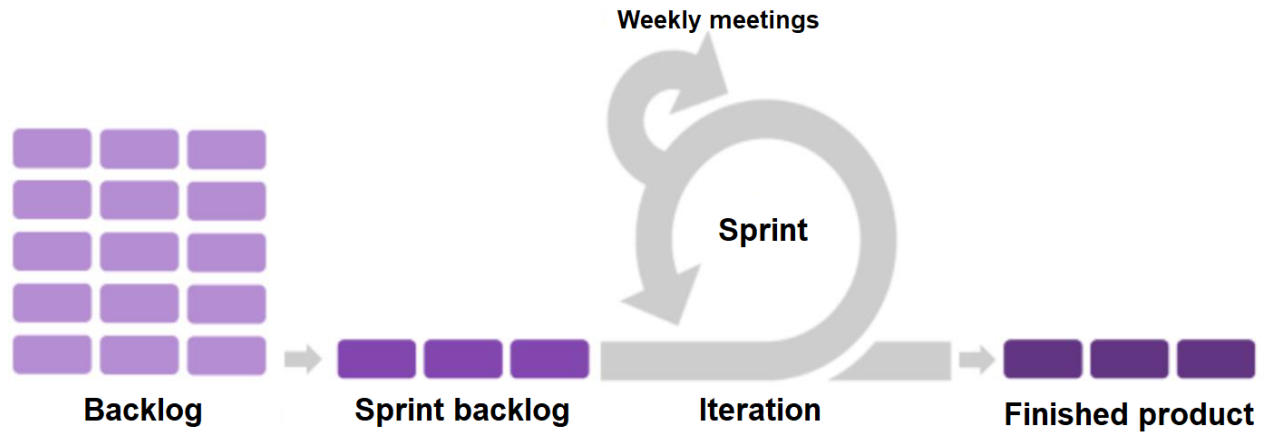


Figure 3.1: Illustration of work flow in agile development.

met with the supervisors, presenting completed work in a progress report, and discussed the next sprint. Larger tasks would be divided in to manageable chunks to fit into a two week sprint. Figure 3.1 illustrates the work flow from backlog to finished product. See progress reports in appendix C.1-C.6.

## 3.2 Materials

This section gives an overview of the materials used to develop the simulator presented in this thesis.

Software used		
Program	Version	Description
Jetbrains CLion	2022.1	A cross-platform IDE for C and C++.
Jetbrains Py-Charm	2022.1	A python development IDE.
Qt Creator	7.0.0	A cross-platform IDE for creating GUI applications.
MS Visual Studio	2019 V16.11	IDE supporting C++ and Python.
linuxdeployqt	28.01.2022	Linux deployment tool
MSVC Compiler	2019	C++ compiler for Windows platform.
MinGW	11.2.0	C++ compiler for Windows platform.
g++	9.4.0	gnu C++ compiler for Linux platform
Apple clang	12.0.0	Apple C++ compiler for MacOS platform

Table 3.1: Table of software used during development.

External libraries		
Library	Version	Description
SymEngine	0.9.0	A library for symbolic manipulation in C++. This was used to generate symbolic functions and callable functions from the sympy-generated functions
OpenCV	4.5.3	Open source library for computer vision. This library was used for pixel manipulation and visualisation in both Python and C++.
Qt	6.2.4	A framework which includes all modules needed for creating a GUI application. Bundled with Qt Creator.
PyTorch	1.11.0	An open source framework for machine learning in Python.
tqdm	4.64.0	Python library for creating smart progress bars.
Matplotlib	3.5.2	Python library used for visualising data and creating figures.
SymPy	1.10.1	A Python library for symbolic mathematics. Used to create mathematical functions which can be read in C++ using Symengine.
NVIDIA CUDA	11.6.55	NVIDIA GPU Toolkit, required for using GPU with PyTorch.

Table 3.2: Table of the external libraries used

### 3.3 Software development

The first thing that needed to be made was a working simulator of the exact point mass model. Put shortly, that meant to interpret and implement the theoretical mathematical models into a working simulator. Once that was done, the simulator could be used to generate data for the ML part of the project. As well as creating a GUI for the simulator. Work on the finite point mass model and spherical model (SIS) was also started after the exact point mass model was up and running.

#### 3.3.1 Choice of software and libraries

##### Python programming language

For the initial prototyping of both the lens model and spherical model, the programming language Python was used. This is perhaps the programming language all the group members are most familiar with. Python is also a very high level and intuitive language with many useful libraries. One such library is Pytorch, used for the machine learning. Python was also used for generating the symbolic functions in the spherical model as well as for analysing the performance of the neural net. Python is well suited for prototyping because it allows fast and easy



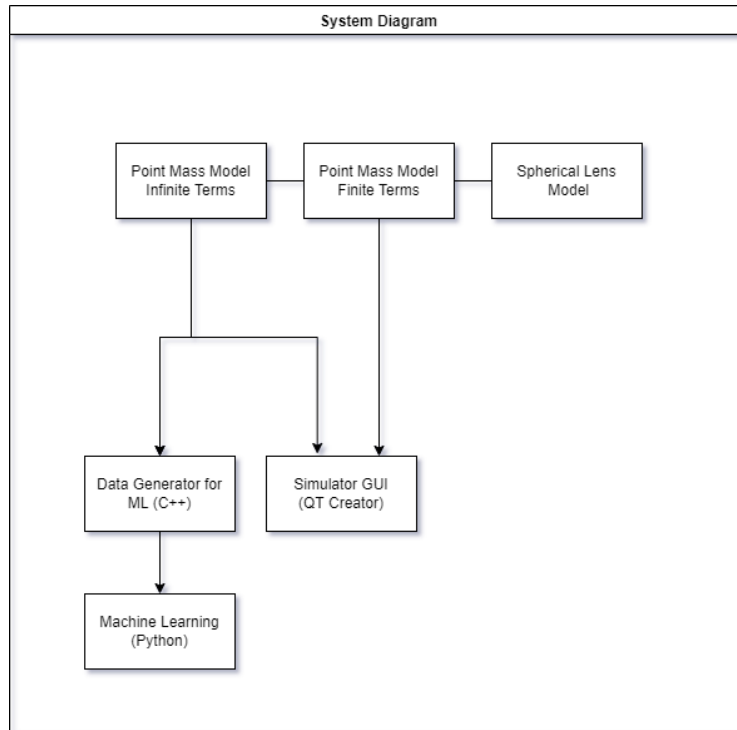


Figure 3.2: System diagram

development and debugging. The language has excellent error messages, compared to C++, and is very forgiving in nature.

### C++ programming language

As the calculations' complexity increased, it became apparent that the speed of Python was insufficient. Heavier calculations led to a slow and non user friendly interface. To increase the performance, it was necessary to make a switch a faster programming language. C++ is well known for its fast computing speed. This computing speed comes at a price of generally slower development time, due to it being a low level language and its less forgiving nature. This is also a language that the group members had somewhat familiarity with, although it is much less user friendly.

### Qt GUI framework

The GUI was implemented in Qt using the Qt Creator IDE. Qt is a development framework for creating GUI applications, and makes it easier to design functional and user-friendly GUIs with cross-platform support. Qt was chosen because it has support for C++ and several options for image- and pixel manipulation and visualisation and is very well documented.

Qt widgets allows for easily dragging and dropping features such as sliders and buttons to the GUI form which can then call functions in the code when a slider is changed or button is pressed. This allows for easy expansion of new features or changing the layout around without doing any changes to previously written source code.

Some useful deployment tools come bundled with Qt. These are used to create executable files which include all dependencies. This means we can deliver an executable application without the end user needing to do any compiling, downloading dependencies or have any know-how at all, it just works. The bundled deployment tools are only for Windows and MacOS systems, so a third-party deployment tool for Linux called *linuxdeployqt* was also used.

The Windows and MacOS deployment tools are simple to use and only needs to be executed in the project build folder to create an executable. However, the tool for Linux requires a specific directory structure to create an executable AppImage file, so a script was created to automate the Linux deployment process.

### **PyTorch machine learning framework**

The PyTorch machine learning framework in Python was chosen for this project. It was chosen because it is open source, has widespread use and is well documented. Also, some of the group members had prior experience with the framework. It offers support for advanced functions like multiple GPU training and tensor cores.

### **SymPy and SymEngine**

symPy is a symbolic math library for Python. It was used to generate the amplitude-expressions for the spherical model. The main reason it was used for this task was for its ability of differentiating and simplifying expressions. SymEngine is a C++ library for symbolic math. It was chosen for its ability to evaluate expressions numerically and to extract expressions from strings, making it possible to load expressions from a text file generated by symPy.

## **3.3.2 Simulator**

What is referred to as “the simulator” is the C++ code that is the the basis for the GL implementation in the GUI. This code is also responsible for generating training data for the neural net. It generates images of an undistorted light-source and creates a distorted image by using equations describing the lens model. The distortion will depend on the chosen lensing model as well as randomly generated variables. This image manipulation gives an accurate representation of how the given lens would affect light from a real source in the sky, at the same time allowing creation of data in a controlled environment.

## **3.3.3 Machine learning**

### **Data generation**

The group generated synthetic data using the exact point mass simulator. The images generated used random lens parameters for each image. Generating data like this meant the group had access to as much training data as could possibly be needed. This process is described in more detail in section 5.3.

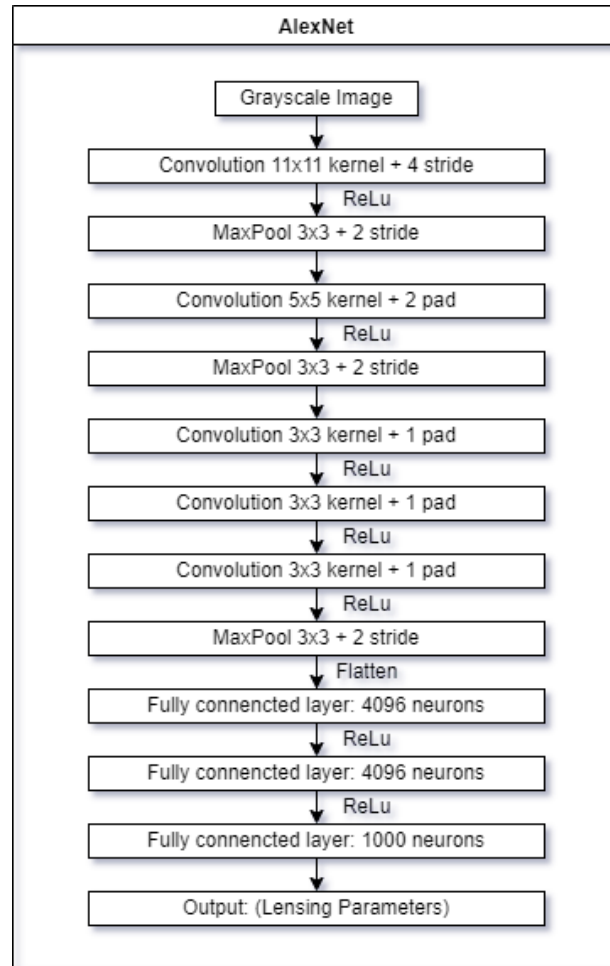


Figure 3.3: AlexNet architecture.

### AlexNet

AlexNet[14] is a convolutional neural net that is specialised on object recognition. It competed in the ImageNet large scale visual recognition challenge 2012. It achieved a top-5 error of 15.3% more than 10% better than the runner up [15]. The top five error is a measure of how often the network does not have the correct class in its top five guesses. Figure 3.3 shows the network that consists of five convolutional layers and three fully-connected layers. It uses the ReLu activation function [16]. AlexNet includes two drop out layers, these layers randomly zeroes out some of the neurons in the fully connected layers to prevent the model from over fitting and increasing generalisation [17]. This network is normally used for images with resolution similar to the the images used in this project. It is also a reasonably light weight network allowing for fast experimenting and parameter-tuning.

### Inception3

Inception3 is a more modern and more advanced convolutional neural network than the previously mentioned AlexNet. It achieved a top-5 error of 5.6% on the ILSVRC 2012 classification

challenge validation set[18]. Substantially better results than AlexNet. With such good credentials the network seemed like a good choice for our project. It would provide a grounds for comparison against AlexNet.

### **Idun - High Performance Computing Group**

Training of machine learning models can require large amounts of computing power. To achieve a reasonable training speed the machine learning model need to be trained using one or more GPUs. For this purpose the group was given access to the IDUN cluster for this project.

The IDUN cluster is a group of high performance compute nodes operated by NTNUs faculties and the IT division. When using large amounts of training data or very large CNN models like AlexNet or Inception3, the demands for GPU power and memory becomes to big for a normal PC. IDUN have GPUs with memory up to 80 GB, allowing loading large batches of data to the GPUs memory.

Training CNN models can certainly be done on a normal PC equipped with a modern GPU, and the group used PCs for easy experimenting. Especially Inception3's large architecture demanded much memory, meaning batch sizes would have to be low. On a GPU with 8 GB memory, batch size of approximately 64 images was the most possible using Inception3. AlexNet was much more memory friendly, allowing batches of 512 to be loaded on the same system. Batch size parameter has a crucial effect on accuracy of image recognition. And according to this 2017 study optimal batch size varies from 200 and greater [19]. Meaning optimal results would not be possible to get using only personal computers. Therefore IDUN was needed.

When given access to IDUN, login nodes can be logged in to via SSH protocol, and every user is given its own storage space. Training jobs are submitted to a queue system (Slurm workload Manager), where running time and needed resources are specified. When available, the requested resources are allocated and the job is started.

Jobs are submitted using a SLURM script. The SLURM script specifies needed resources and modules, as well as specifies the file path to the program or script to run. Appendix E.3 shows one of the scripts used to initiate a job on IDUN. Required nodes, GPUs and run time are some of the parameters defined in the job script. If the run time is exceeded the job is cancelled.

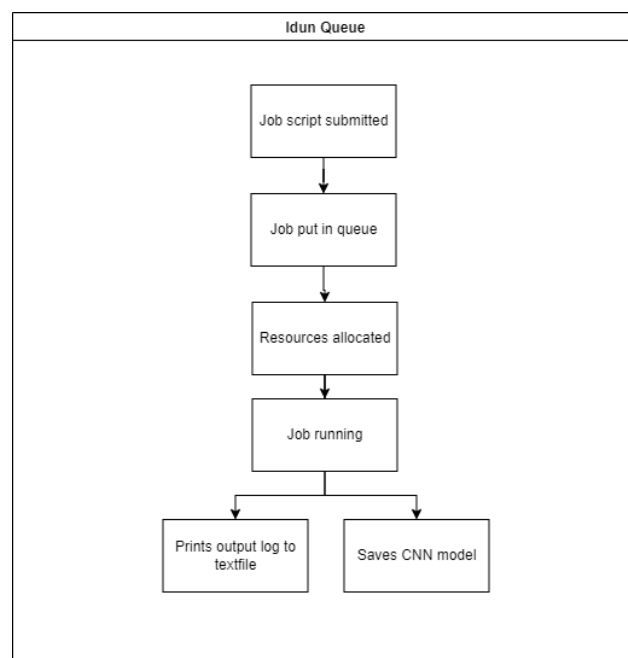


Figure 3.4: IDUN flow

# Chapter 4

## Results - Simulator

This chapter will highlight the findings related to the first goal presented in 1.2;

*“The first goal is to develop a tool for simulating different lensing models. By making this tool a simulator application, researchers can easily visualise their mathematical models without any programming knowledge, giving them more time to focus on their respective fields. A big challenge in developing this application is to make it user friendly while still including the functionalities a researcher might require.”*

### 4.1 Deriving equations for apparent position and R

Before the implementation could commence, some equations had to be derived. This section will cover these equations to give the crucial insight needed to fully understand the algorithm’s implementation, however, readers not concerned with the mathematical details can skip this section just fine.

Equation 2.5 presented in chapter 2.1.8 is the standard result for a point mass model for mapping from the source image to the distorted image. In order for a computer to run this mapping algorithm and solve equation 2.5, quite a few variables are needed. The variables *Einstein radius* ( $R_E$ ), *distance from observer to lens* ( $\chi_L$ ), *distance from observer to source* ( $\chi_S$ ), and *source’s actual position* ( $P_{\text{act}}$ ), will be taken as user inputs. Still, the two variables *apparent position of the source* ( $P_{\text{app}}$ ) and *distance between the lens and the distorted image* ( $R$ ) must be calculated. This is needed both for establishing the relative positions between the three coordinate systems and for solving equation 2.5. The simplified point mass model assumes that the centre of mass of the source is located on the positive x-axis, so for now we can treat  $P_{\text{act}}$ ,  $P_{\text{app}}$  and  $R$  as scalar values.

Given  $R_E$ ,  $\chi_L$ ,  $\chi_S$  and  $P_{\text{act}}$ , we can start by solving equation 2.6 for  $P_{\text{app}}$ :

$$P_{\text{app}} = P_{\text{act}} + \frac{\chi_S}{\chi_L} \frac{R_E^2}{R} \quad (4.1)$$

As  $R$  is a projection of  $P_{\text{app}}$  onto the lens plane, we know that

$$R = P_{\text{app}} \frac{\chi_L}{\chi_S} \quad (4.2)$$

We can then substitute for  $R$  in equation 4.1 and solve for  $P_{\text{app}}$ :

$$P_{\text{app}} = \frac{P_{\text{act}} \pm \sqrt{P_{\text{act}}^2 + 4\left(\frac{\chi_S}{\chi_L}\right)^2 R_E^2}}{2} \quad (4.3)$$

To get this result in a more useful format we divide it by  $P_{\text{act}}$  to get the ratio between  $P_{\text{app}}$  and  $P_{\text{act}}$ . Simplifying gives this equation:

$$\frac{P_{\text{app}}}{P_{\text{act}}} = \frac{1}{2} \pm \sqrt{\frac{1}{4} + \frac{R_E^2}{\chi^2 P_{\text{act}}^2}} \quad (4.4)$$

This equation gives a scaling term that can be multiplied with  $P_{\text{act}}$  to get  $P_{\text{app}}$ .

Equation 4.4 will give two solutions for  $P_{\text{app}}$ . The two solutions shows up in the distorted image, as can be seen in figure 4.6. The solution with largest absolute value corresponds to the larger portion of light that is shifted away from the centre of the lens, i.e., the big blob. And the solution with the smallest absolute value corresponds to the “reflection” of the source inside the Einstein radius, i.e., the small blob. This result was discovered quite inadvertently, but the literature [1] repeatedly mentions “the secondary image”, so this must be it.

Knowing  $P_{\text{app}}$  we can solve for  $R$  using equation 4.2. Now we have all the information needed to go on and solve equation 2.5 for each point in the distorted image.

## 4.2 Exact point mass model

This section documents the development of the prototypes of the exact point mass model. This model maps image pixels from the undistorted image to the distorted image through the mapping presented in equation 2.5.

### 4.2.1 First prototype

The very first prototype was written in Python. It took in user input in form of sliders for the variables *source\_size*,  $P_{\text{app}}$  and  $R_E$ . This implementation was made before a proper understanding of the mathematics was established. There were still some misunderstandings regarding the various variables, and the output did not look as expected, as seen in figure 4.1. The program also ran quite slow, although the resolution of the images were quite low, so to speed up the rendering, it was decided to continue the work in C++.

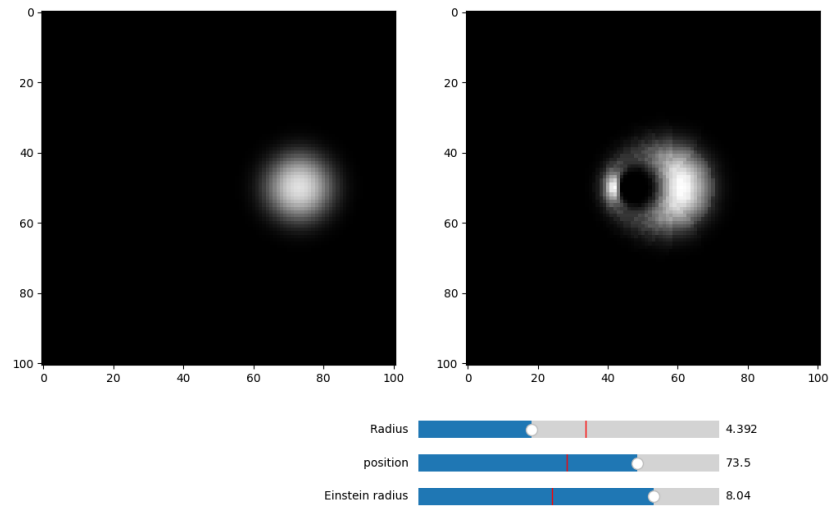


Figure 4.1: Screenshot of the first prototype

---

```

for each pixel in imgDistorted do
  translate matrix indices to (x, y) by shifting and scaling
  calculate polar coordinates (r,  $\theta$ )
  calculate (x', y') using equation 2.5
  translate (x', y') to matrix indices in imgApparent
  copy pixel value from imgApparent to corresponding pixel in imgDistorted
end for

```

---

Figure 4.2: Algorithm for exact point mass model, second prototype.

### 4.2.2 Second prototype - Simplified but correct implementation in C++

The second implementation was written in C++ after getting a fuller understanding of the various variables, but before deriving proper equations for calculating  $R$  and  $P_{app}$ . To simplify things, the ratio between  $\chi_S$  and  $\chi_L$  was assumed to be one, which places lens and source on the same plane. This simplified things quite a bit as only one coordinate plane had to deal with. The equations 4.3 and 4.2 was not yet derived but a simplified version with  $\chi_L/\chi_S = 1$  was used. The simulator took in user input for the three aforementioned variables, calculated  $P_{app}$  and  $R$  (which are equal). Next, a matrix *imgApparent* was created with a light source drawn at the apparent position. An empty matrix *imgDistorted* with the same size was created to contain the distorted image. Then the mapping algorithm 7 could run. Screenshot of the user interface and the images produced by the second prototype is shown in figure 4.3.



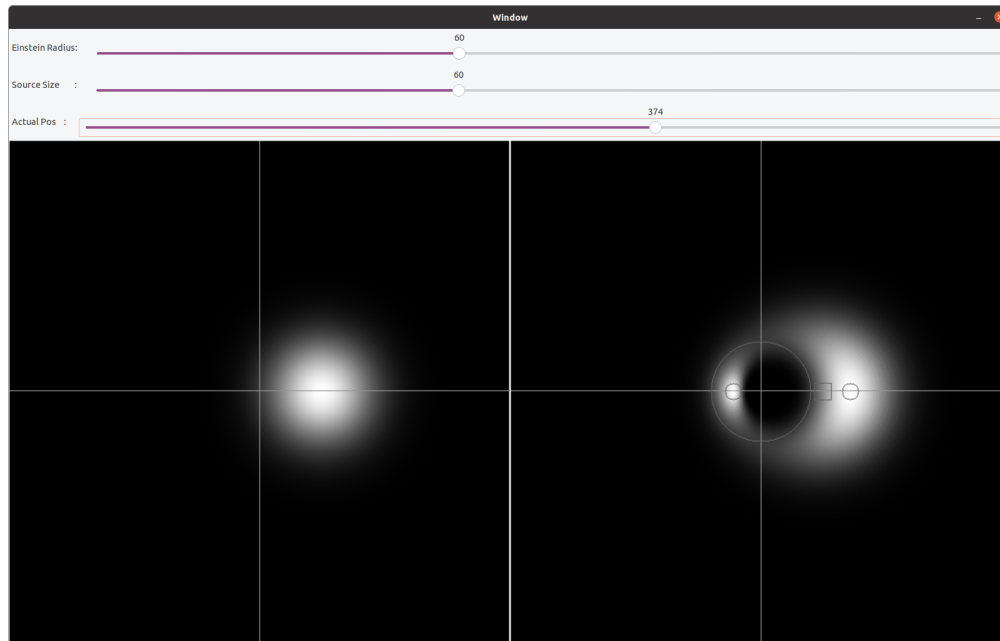


Figure 4.3: Second prototype of the simulator (undistorted left, distorted right)

### 4.2.3 Third prototype - Including $\chi_L$ and $\chi_S$

The next version took  $\chi_L/\chi_S$  into account. This placed the source on a separate plane from the lens. As we are only interested in the ratio between the two, the implementation of the ratio is simply a floating point variable  $0 < \chi \leq 1$ . The ratio can then be adjusted by the user. Given *actual position*, *Einstein radius*, *source size*, and  $\chi$ , it is possible to calculate the *apparent position* and  $R$  with equations 4.3 and 4.2. Just like before, images *imgApparent* and *imgDistorted* were made. This time however, as *imgDistorted* is a projection from the image plane onto the lens plane and  $\chi$  is not necessarily equal to one, the *imgDistorted* matrix had to be scaled by  $\chi$ . Now the mapping algorithm can run. The fraction  $\chi_L/\chi_S$  in equation 2.5 must be replaced by  $1/\chi$  and after the mapping, the distorted image must be scaled by  $1/\chi$  to get the distorted and the undistorted image to the same size for comparison.

Figure 4.4 is a screenshot of the third prototype. At this point, the two intermediate images was shown on screen for debugging purposes. From the left: undistorted image with source at actual position, undistorted image with source at apparent position, distorted image (projected onto lens plane), and resized distorted image. The large circle shows the Einstein radius, the small circle the apparent position and the small square the actual position. In this prototype the origin of the lens was moved to the left to simplify the coordinate transformations, but it is moved to the centre in the final prototype.

An issue with this prototype was that the areas of the *imgDistorted* that maps to outside of the frame of *imgApparent* stayed black no matter what. When part of the source ends up outside of the frame in *imgApparent*, we get sharp edges in *imgDistorted* corresponding to the edges of *imgApparent*. This is shown in figure 4.5. To solve this, the width of *imgApparent* was doubled, so that no part of the light source ever reached the edge. Of course the edge was still there, only farther away, so that the area at the edge was black anyway. This change in size had to

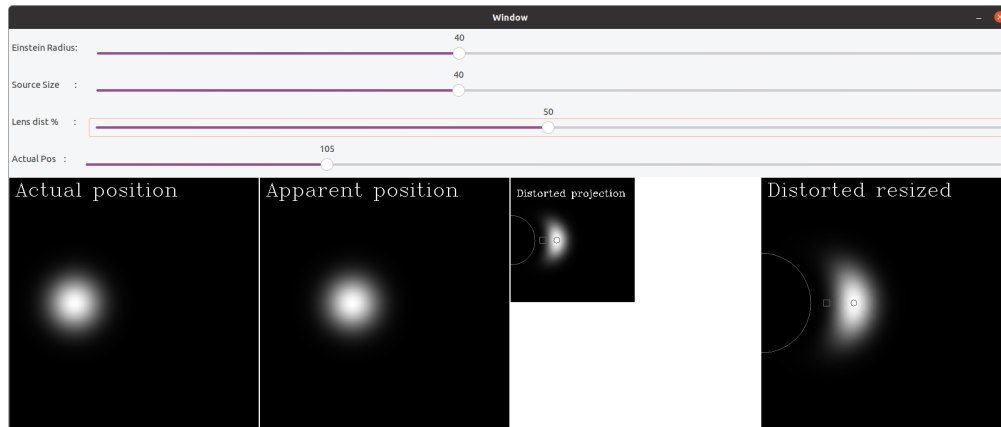


Figure 4.4: Simulator with  $\frac{\chi_L}{\chi_S}$  implemented

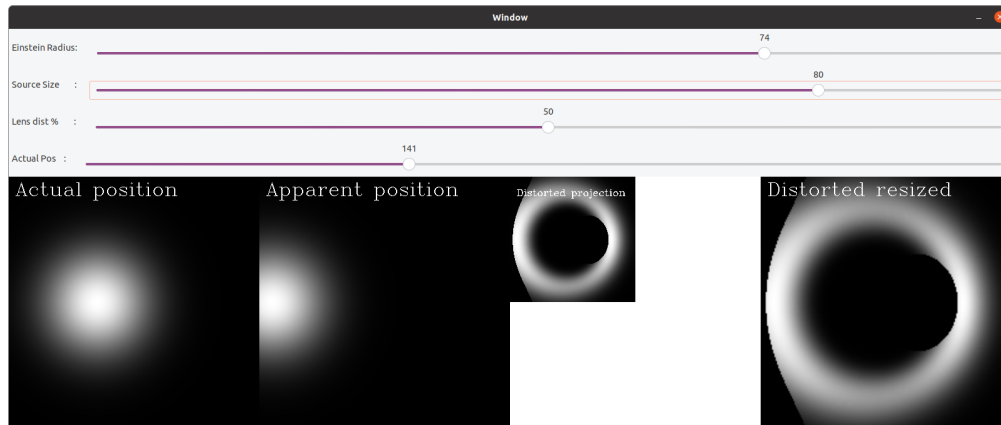


Figure 4.5: Artefacts from mapping from outside source image

be taken into account when doing the mapping and later when implementing rotation of the image.

#### 4.2.4 Fourth prototype - Complete exact point mass simulator

The previous prototypes had some shortcomings that needed to be addressed. The first was that it assumed the source is on the positive x-axis of our coordinate system. In reality, the source would most likely also have a y-component. To implement the variable *y-position*, a slider in the prototype's interface was added to allow the user to adjust the source's y-position. The polar coordinates (lets call them  $K$  and  $\phi$ ) was calculated and  $K$  used as  $P_{act}$  (still on the x-axis). The distorted image was calculated just as before. Due to the circular symmetry of the lens *and* the source, the image could simply be rotated by the angle  $\phi$  to reveal the the correct distorted image.

The second shortcoming was the mapping from the image plane onto the "smaller" lens plane and scaling of the distorted image. This process led to lower resolution on the final image when using small  $\chi$ . To avoid the "compression" of the image as *imgApparent* is projected onto the

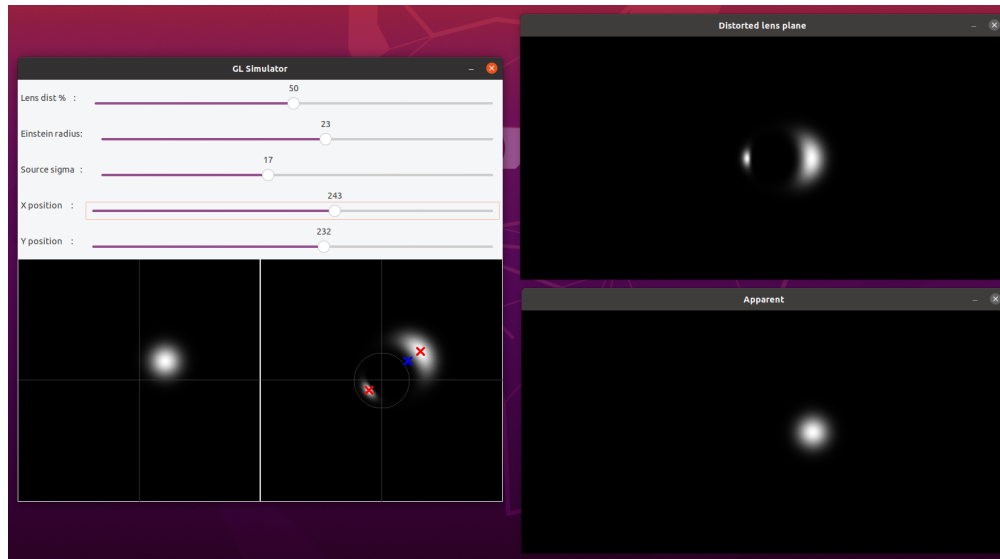


Figure 4.6: Complete simulator, exact point mass model

lens plane, *imgDistorted* is not scaled down by  $\chi$ . Instead, the x-, and y-position are scaled by  $\chi$  when going from matrix indices to  $(x, y)$  position in the lens plane.

Figure 4.6 shows the complete simulator for the exact point mass model. The main window includes the undistorted image at actual position on the left, as well as the distorted image on the right. The intermediate images are shown in separate windows to the right. At the bottom: undistorted image at apparent position. At the top: distorted, but not rotated image. Notice that the intermediate images has the source along the x-axis. Notice also that the distorted image is never scaled down as it was before. The blue marker show  $P_{act}$ . The red markers shows the two solutions to  $P_{app}$  as mentioned in the beginning of the chapter. Surprisingly, both solutions for  $P_{app}$  could be used for calculating R and placing the source in *imgApparent* and they actually give the same distorted image! When using the smallest version however, when  $P_{act}$  is large,  $P_{app}$  goes towards the centre of the lens. A small change in  $P_{act}$  would therefore not necessarily change  $P_{app}$  enough to move it one pixel and there would be no visible change in the output. This is why the largest solution is used.

### 4.2.5 Multithreading

To achieve a responsive and smooth user experience for the simulator GUI, and fast data generation for the machine learning, the program was parallelized. The task performed by the simulator could be parallelized without much difficulty concerning thread safety. The different threads would access the same image matrix concurrently, but threads would never access the same indices (pixels). This can be guaranteed because each thread is assigned its own section of the image. The program detects the number of logical threads available from the CPU, and divides the image matrix into the same amount of sections. Figure 4.7 displays how the threads would divide the workload. In cases where the number of rows in the image could not be divided by number of threads, the last thread would be assigned the remainder. The multithreaded implementation gave considerable improvement in simulator performance.

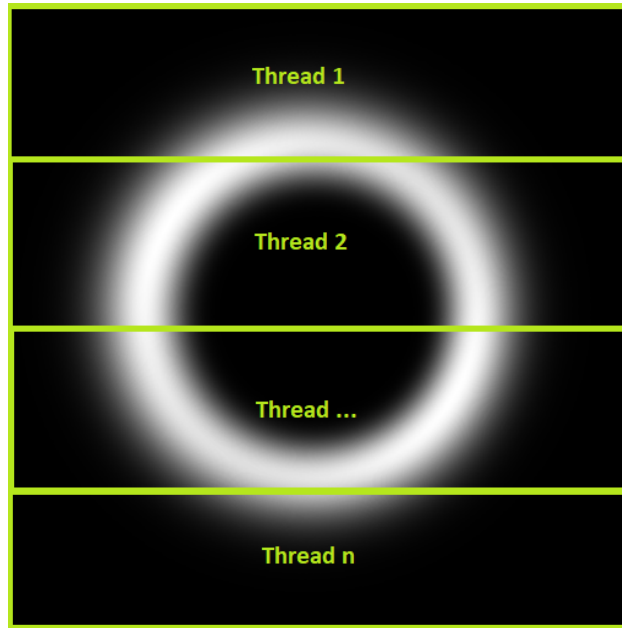


Figure 4.7: Each thread is assigned its own range of rows.

### 4.3 Graphical User Interface (GUI)

The goal for the simulator's usage emphasise user friendliness. Presenting a script in Python or C++ to a user without any prior programming knowledge does not bode well for the accomplishment of that goal. The GUI will allow the user to see the visualisation of the algorithm's results in real time as parameters are adjusted, without having to know anything about the code, nor the algorithms. As it should be with all modern applications.

The simulator originally used OpenCV for processing images and visualising them in a simple user interface with sliders to tweak variables. While it is possible to create a more advanced GUI in OpenCV, it is not a simple task to implement a functional yet good-looking user interface using this library. This led to switching OpenCV for Qt.

The GUI application was made using Qt Widgets and the source code is written in C++. It utilises the simulator code for creating images and allows the user to determine the parameters of the source and lens model. It then shows the images to the user, updating every time the user tweaks a parameter. This gives the possibility to see the parameters effects in real time. A rough sketch of the overall look and feel of the GUI was created to be used for designing the final GUI in Qt. After a base version of the GUI was created, it could easily be expanded with new ideas for functionality or features requested from the intended user. Figures 4.8 and 4.9 shows an overview of all the GUI elements with labels explained in table 4.1.

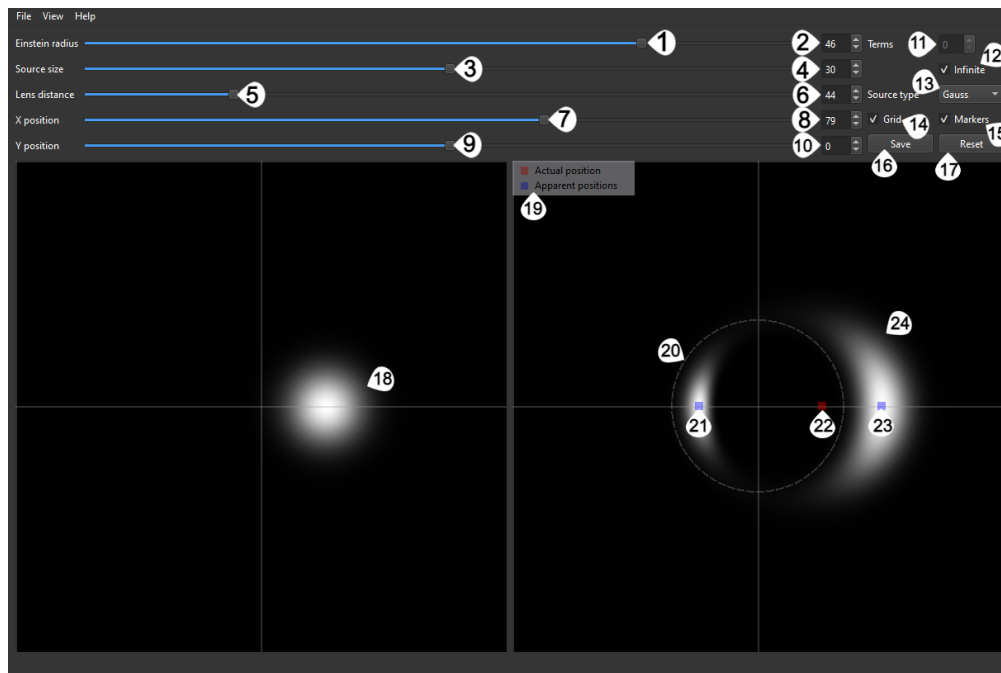


Figure 4.8: Overview of GUI with numbered elements.

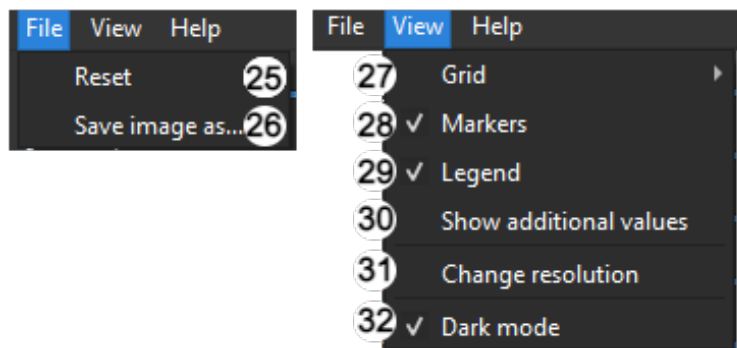


Figure 4.9: Overview of GUI menus with numbered elements.

GUI overview		
Label	What	Description
1	Einstein radius slider	Change einstein radius by moving the slider
2	Einstein radius spin-box	Change einstein radius by typing a value or pressing the arrows
3	Source size slider	Change source size by moving the slider
4	Source size spinbox	Change source size by typing a value or pressing the arrows
5	Lens distance slider	Change lens distance by moving the slider
6	Lens distance spinbox	Change lens distance by typing a value or pressing the arrows
7	X position slider	Change actual X position of source by moving the slider
8	X position spinbox	Change actual X position of source by typing a value or pressing the arrows
9	Y position slider	Change actual Y position of source by moving the slider
10	Y position spinbox	Change actual Y position of source by typing a value or pressing the arrows
11	Terms spinbox	Change number of terms used in equation 2.1
12	Infinite terms check-box	Check to set terms to infinite
13	Source type selector	Change to a different source shape
14	Grid checkbox	Turn on or off the grid/crosshair
15	Markers checkbox	Turn on or off markers on the distorted image
16	Save button	Save images to computer
17	Reset button	Reset to the initial startup values
18	Source in actual image frame	This is the undistorted source in the actual position
19	Legend	Legend explaining what the markers represent
20	Einstein radius indicator	Shows size of the einstein radius
21	Apparent position 1 marker	First calculated apparent position of the source
22	Actual position marker	Actual position of the source
23	Apparent position 2 marker	Second calculated apparent position of the source
24	Distorted source in distorted image frame	This is the distorted source
25	Reset menu option	Same as 17
26	Save image as...	Same as 16
27	Grid view menu	Change size of grid
28	Marker toggle	Same as 15
29	Legend toggle	Turn off legend view while keeping markers
30	Show additional values	Shows apparent position coordinates in new window
31	Change resolution	Set a custom resolution, used for either better quality or better performance
32	Dark mode	Toggles dark mode theme

Table 4.1: Table explaining labels in 4.8 and 4.9

### 4.3.1 Implementation of exact point mass simulator in GUI

Implementing the simulator in Qt and removing any dependency of OpenCV required some changes in the simulator code. First, the image formats needed to be changed. OpenCV's *Mat* data type was originally used to store a matrix of the pixel intensity values. This was exchanged for the *QImage* class in Qt, which is suited for direct pixel access and manipulation.

After the simulator is done distorting an image, the image data type is changed to a *QPixmap*, which is better suited for drawing and displaying an image in Qt. Qt's *QPainter* class could then be used to rotate and scale the image before drawing a legend, a grid, markers for points of interest and a dash-line circle to visualise the einstein radius. Finally, the *QPixmap* can be shown as an image on the GUI itself. Figure 4.10 shows a somewhat simplified process of how the simulator is implemented in the GUI.

The GUI consists of several sliders, spinboxes, buttons, check boxes and menu options. The sliders and spinboxes are used to tweak input values to the simulator. There is a button to reset input values back to default, and a button to save the displayed images. Check boxes are used to turn on or off various visual aids in the images such as the grid, markers and legend. There is also a check box to toggle between the exact and finite lens models. In the top menu bar there are options to change the size of the grid, change the theme, show some additional information as well as setting a custom image resolution. Tooltips were added to some of the GUI elements which gives a short explanation of their functions when the mouse hovers over it.

### 4.3.2 Experimenting with different sources

Using a gaussian distribution as a light source was suggested in the assignment description, but it was a somewhat arbitrary choice. A few different light sources was experimented with. This was mostly for fun but uncovered a few insights not obvious when viewing the circular symmetric gaussian light source. Three new sources were implemented; a ring, a square and an image of a rocket. These images gave more insight into which point in the primary image corresponds to which point in the secondary image as can be seen in figure 4.11. It also gave some insight to the direction of the spurious images as seen in figure 4.12. The figures does not describe this phenomena fully, to get a real feel for it, experimenting with the application is recommended.

When using sources that are not circular symmetric like a square or a rocket, it became apparent that the rotation after the distortion would not give a correct distorted image as the source would be rotated by the angle  $\phi$ . To correct this, the first approach was to rotate the source image around the centre of mass of the source after it was placed at the apparent position (on the x-axis), but before the distortion. This produced a correct distorted image. After a discussion with the supervisors, it became evident that the same result could be achieved in a more intuitive way by placing the source at actual position in the plane (not necessarily on the x-axis), shift it by the ratio in equation 4.4 and then rotate the image around the origin (centre of the lens) by  $-\phi$ . This resulted in both moving it to the x-axis and rotating it. The distortion could then be performed before rotating it back. It must also be mentioned that the centre of mass of the rocket has not been calculated, so the distortion of this image is not completely correct.

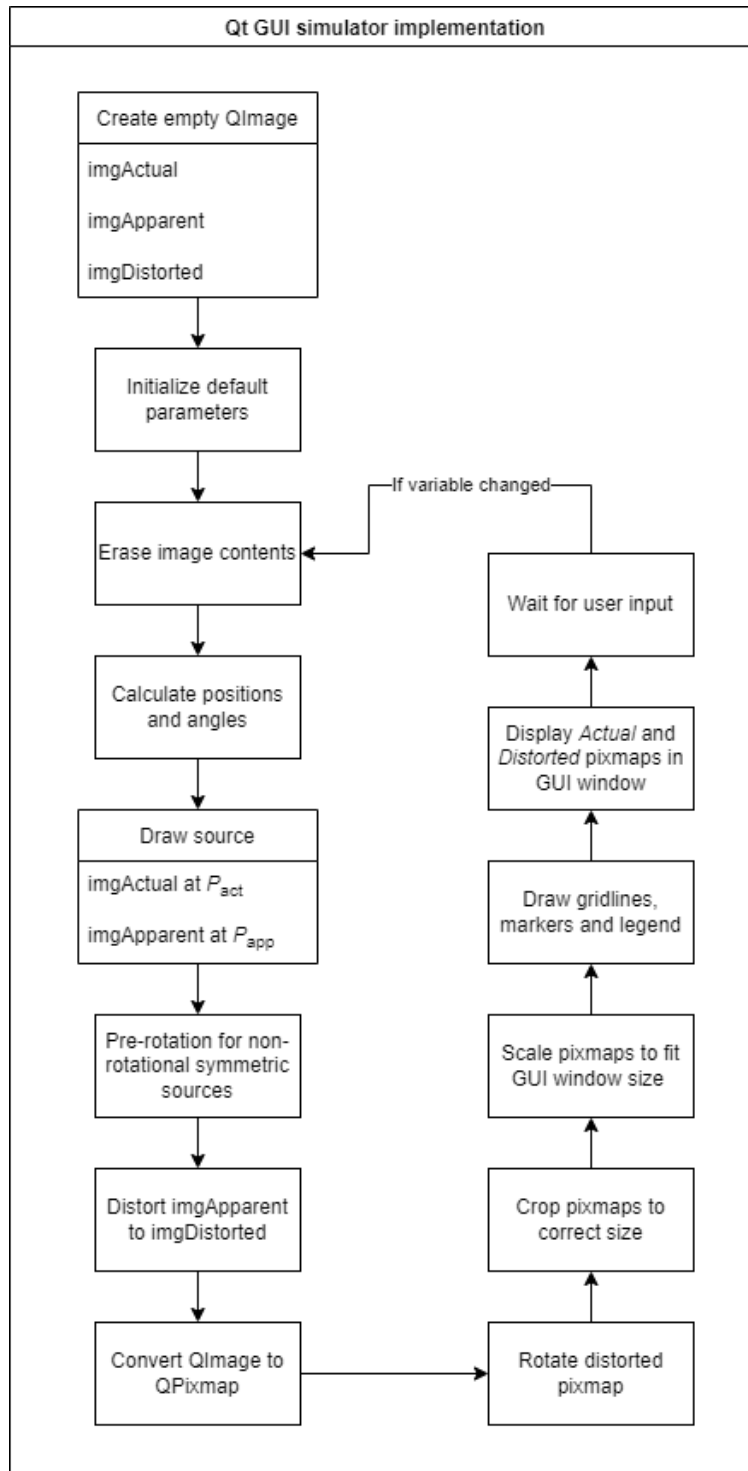


Figure 4.10: Implementation of simulator in Qt.



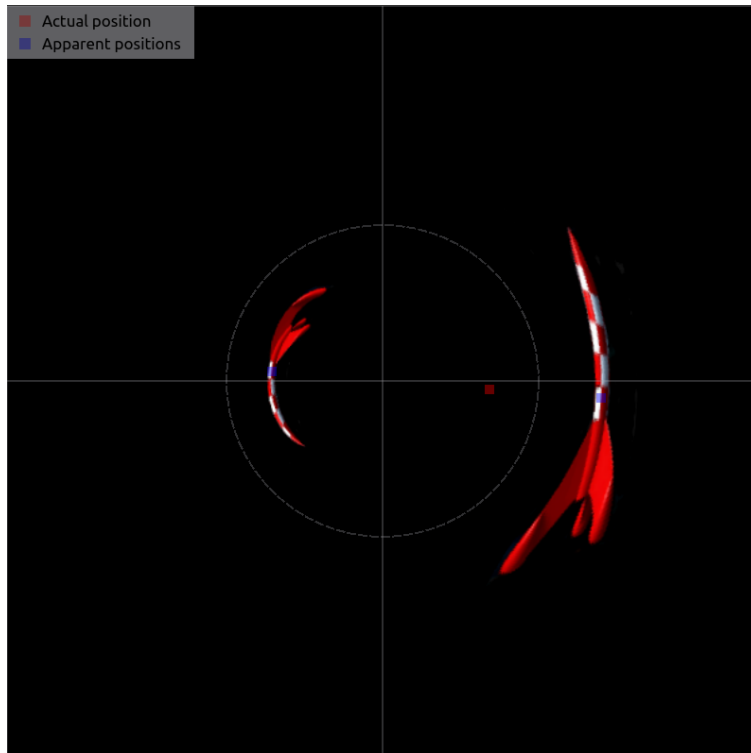


Figure 4.11: Image of the Tintin rocket, distorted by the exact point mass model

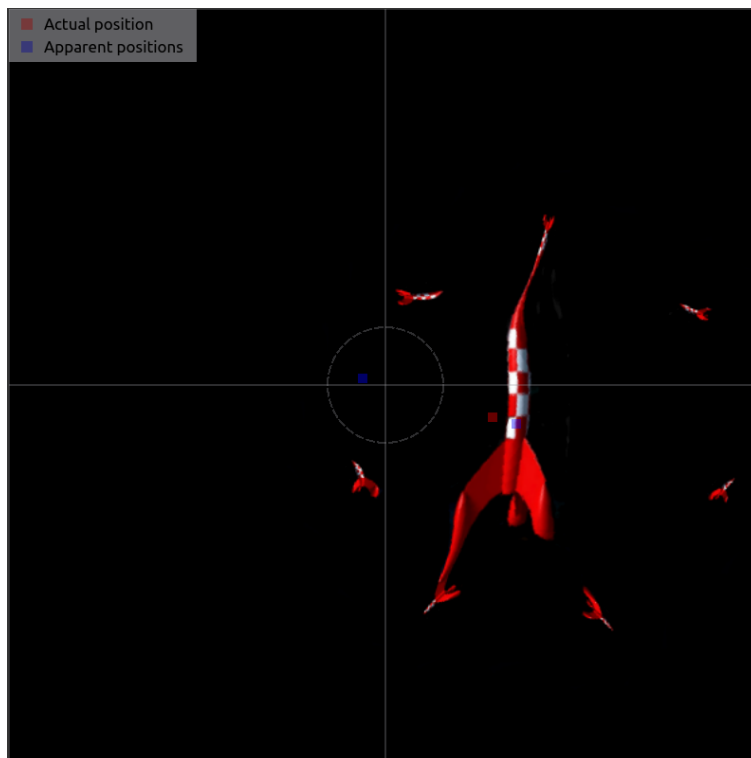


Figure 4.12: Image of the Tintin rocket, distorted by the finite point mass model

## 4.4 Singular isothermal sphere model (SIS)

When implementing the SIS model in the simulator, much of the code developed for the exact point mass model could be reused. The apparent position and position in the lens plane, could be calculated using the same formula as before. This model allows for positions in the whole plane, unlike the point mass case, where it had to be along the positive x-axis. This was not a problem as equation 4.4 could be used to scale both the  $x$  and  $y$  component of  $P_{\text{act}}$  to get  $P_{\text{app}}$ . The rotation before and after the distortion could be omitted. In appendix A, another variable  $2\Gamma$  was used to describe the strength of the lens, rather than the Einstein radius. Equations 2.15 and 2.3 reveals that the Einstein radius simply is equal to  $2\Gamma$ . This is all we need in terms of relative positions. However, the algorithm for doing the actual mapping 2.1 relies on the amplitudes  $\alpha$  and  $\beta$ .

### 4.4.1 Analysing the equations

Looking at equation 2.1, one can observe that each term has a factor  $r^m/m!$ . When plotting this function, as shown in figure 4.13, it is clear that it gets very large at low  $m$ , and then gets negligible at about  $m > 3r$ . This figure only shows  $r = 10$ , but experimenting with different values of  $r$  gives approximately the same result. As a reminder;  $r$  is the distance between the evaluated point and the centre of the image in the lens plane. When summing up to  $m =$  some number  $n$ , we can assume that we get a good approximation, at least in the the area with  $r < n/3$ . This is assuming that the inner sum does not grow very fast.

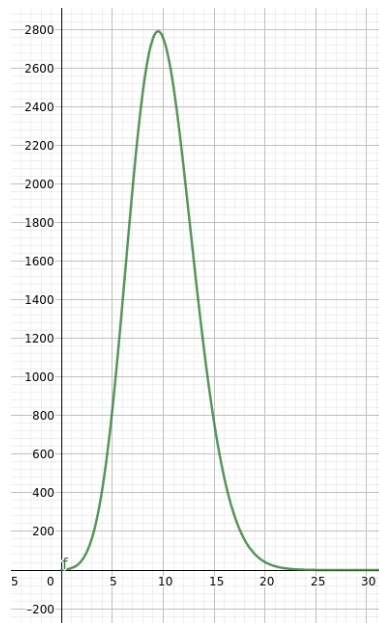


Figure 4.13: Plot of  $\frac{r^m}{m!}$  with  $r = 10$

Table 4.2: Example of alphas for  $n = 5$ 

	$\alpha_1^0$					
$\alpha_0^1$		$\alpha_2^1$				
	$\alpha_1^2$		$\alpha_3^2$			
$\alpha_0^3$		$\alpha_2^3$		$\alpha_4^3$		
	$\alpha_1^4$		$\alpha_3^4$		$\alpha_5^4$	
$\alpha_0^5$		$\alpha_2^5$		$\alpha_4^5$		$\alpha_6^5$

#### 4.4.2 Generating expressions for the amplitudes

The SIS-model does not allow for the same simplifications of the mapping equation 2.1 that the point mass models does. This makes the SIS-model implementation much more computationally expensive. To make the process of calculating the sum as quick as possible, a set of expressions were generated for the amplitudes,  $\alpha_s^m$  and  $\beta_s^m$ , up to  $n$ . These were organised in two dimensional arrays with rows corresponding to  $m$  and columns corresponding to  $s$  (zero based indexing). These would be callable functions of the variables  $X$ ,  $Y$ ,  $\Gamma$  and  $\chi$ . These would be saved to a file, then loaded upon starting the program. When running the program these functions could be evaluated directly without any differentiation.

The first approach used Python and the library *symPy* to make a proof of concept for the generating these expressions. The symbolic functions for  $\alpha_1^0$  and  $\beta_1^0$  was declared using equation 2.7 and 2.8. From there the rest of the expressions in the diagonal going from  $(0,1)$  to  $(n-1, n)$  could be generated by using the recursion relations in equations 2.9 and 2.10, shown as red arrows in table 4.2. The rest of the table could now be filled in using equations 2.11 and 2.12, shown as blue arrows in table 4.2. When the symbolic functions were generated, they could be converted to callable functions. This was done with the *symPy* function *lambdify*. However, this approach caused some problems.

When generating arrays of functions with  $n > 10$ , the functions became very complex. Causing the time to generate and evaluate the functions, as well as the memory usage, to seemingly grow exponentially. *symPy* has a built in function to simplify expressions and this reduced the complexity considerably. However, the time to simplify the expressions also seemed to grow exponentially, making the time to generate up to  $n = 10$  taking hours. For this to be a valid road ahead, we would have to make do with the terms up to  $m = 10$  or optimise the code in some way.

#### 4.4.3 C++ and SymEngine

The next approach was to implement it in C++. This would hopefully increase the performance. In addition, since both the simulator for the exact point mass model and the GUI were written

---

```

generate  $\alpha_1^0$  and  $\beta_1^0$  using 2.7 and  $\psi$ 
for (m, s) along the diagonal do
  Generate  $\alpha_s^m$  and  $\beta_1^0$  using 2.9, 2.10 and the previous  $\alpha$  and  $\beta$ 
  write to file
  Queue a process in the process pool starting at  $(m_{\text{start}}, s_{\text{start}})$ 
end for

```

---

Figure 4.14: Algorithm for generating expressions along the diagonal from (0,1) to (n, n+1)

---

```

for (m, s) starting at  $(m_{\text{start}}, s_{\text{start}})$  to the edge of the table do
  Generate  $\alpha_s^m$  and  $\beta_1^0$  using 2.11, 2.12 and the previous  $\alpha$  and  $\beta$ 
  write to file
end for

```

---

Figure 4.15: Algorithm for generating expressions from  $(m_{\text{start}}, s_{\text{start}})$

in C++, it would simplify the integration process. A lot of time was spent exploring symbolic libraries in C++ including *SymEngine*, *GiNaC*, *viennaMath* and others. They were all quite complicated to use as most of them were designed for being the foundation for more user friendly Python, R and Julia APIs. In addition, none of them had the functionality of simplifying an expression like *symPy* could. It was still possible to generate the expressions, but without the means to simplify the expressions, the complexity, evaluation time and memory usage grew rapidly. As luck would have it, *SymEngine* had the functionality of parsing a string and converting to symbolic function. This meant the functions could be generated and simplified by *symPy* in Python and written in a text file. When starting the simulator/GUI, the text file could be loaded and the strings parsed in C++ and *SymEngine*. The next issue was optimisation of the Python/*symPy* code to be able to generate a sufficient amount of expressions.

A few different methods of parallelization and simplification were used to optimise the *symPy* code. The fastest parallelization method was using Python's multiprocessing library and a pool of processes. One process generated the expressions along the diagonal, and for each expression calculated, a new process was initiated, calculating all the expressions to the lower left that was depending on that expression. Each new term was written to a text file continuously. To avoid multiple processes writing to the file at the same time, the writing to file was handled by a separate process and a queue system. An error was found in the initial steps of the algorithm, making the expressions simpler than they should be. When correcting this error, the number of viable terms were decreased. Using *symPy*'s *simplify* method was slow, but not using it produced extremely large expressions that were time-consuming to evaluate and took up a lot of memory, further limiting the number of viable terms. Using *symPy*'s *factor* function in stead of *simplify* was a lot quicker, and seemed to produce equivalent expressions. With the multiprocessing algorithm and simplifying using *factor* all terms up to  $m = 100$  was generated in a few minutes.

```
m: 0 s: 1 alpha: -2*g*x/(c*sqrt(x**2 + y**2)) beta: -2*g*y/(c*sqrt(x**2 + y**2))
m: 1 s: 2 alpha: 1.0*g*(x**2 - y**2)/(x**2 + y**2)**(3/2) beta: 2.0*g*x*y/(x**2 + y**2)**(3/2)
m: 1 s: 0 alpha: -1.0*g/sqrt(x**2 + y**2) beta: 0 c: 0.5*c
```

Figure 4.16: Expressions generated by *symPy*


---

```
(x', y') ← (0, 0)
for m in [1:n] do
    calculate  $\frac{r^m}{m!}$ 
    for s in [0:m+1] do
        get  $\alpha$  and  $\beta$  values from array
        calculate  $c^+$  and  $c^-$  from 2.2
        accumulate sub terms for  $(\xi_1, \xi_2)$  with the inner sum of equation. 2.5
    end for
    multiply sub terms with fraction and add to  $(x', y')$ 
end for
```

---

Figure 4.17: Algorithm for mapping from undistorted to distorted image

#### 4.4.4 Implementation of SIS-model in simulator

The  $\alpha$  and  $\beta$  arrays has to be loaded from a file at startup. When updating the image, each expression for the amplitudes was called with the current values of  $X$ ,  $Y$ ,  $\chi$  and  $\gamma$ , and the numerical values are stored in separate arrays of same shape as the expression-arrays. When evaluating each pixel in the distorted image, corresponding position  $(x', y')$  in the source plane is calculated using two nested for-loops. For each iteration the corresponding amplitudes are read from the aforementioned arrays. The two sums are accumulated term by term while  $m \leq n$ .

The output from this implementation of the SIS model did not look as expected. As can be seen in figure 4.20, the amount of light in the distorted image (right) is much greater than in the undistorted image (left). The distorted image does not have the expected banana-shape either. It is uncertain if these are caused by an error in the expressions for  $\alpha$  and  $\beta$ , an error in the simulator itself or in the derivation of the formulae. Some of the elements of the output did however make sense. The lens seemed to have circular symmetry, and the expected spurious images were there. Moreover, this implementation suggests that it is possible to make completely working implementation.

#### Verification of expressions

As the output did not look quite as expected, the generating, saving, loading and parsing of the expressions had to be tested to see if the error could be found here. To verify the expressions generated in *symPy*, some of the expressions were calculated manually (with some help from wolfram alpha) and compared to the expressions in the arrays.  $\alpha_1^0$  and  $\beta_1^0$  could easily be calculated by differentiating  $\psi$  with respect to  $X$  and  $Y$  respectively and multiply by  $-\chi$  which gives the same result as the first line in figure 4.16. For  $m = 1$ ,  $s = 0$  the  $C_+^+$  value was calculated to

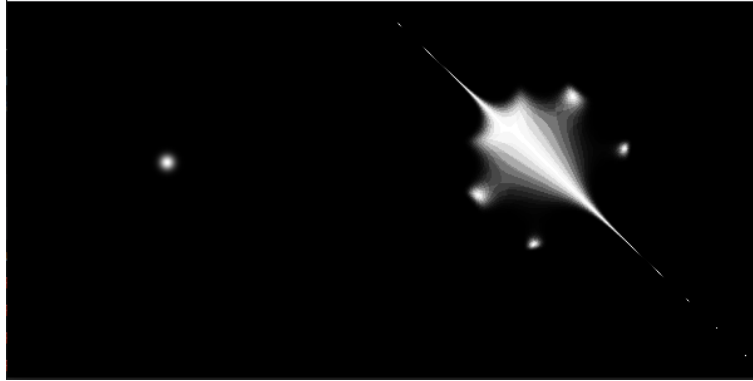


Figure 4.18: SIS simulator  $n = 6$

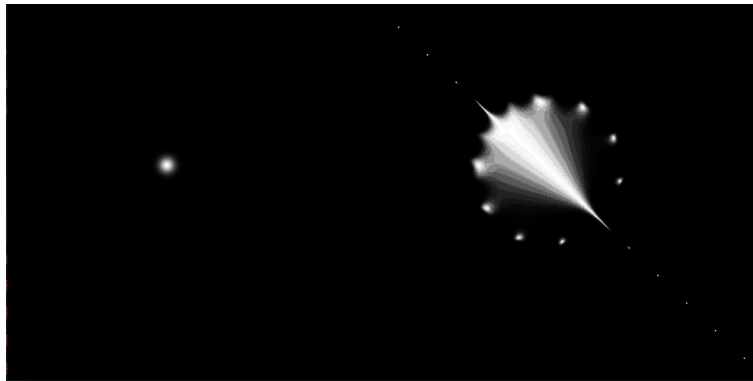


Figure 4.19: SIS simulator  $n = 10$

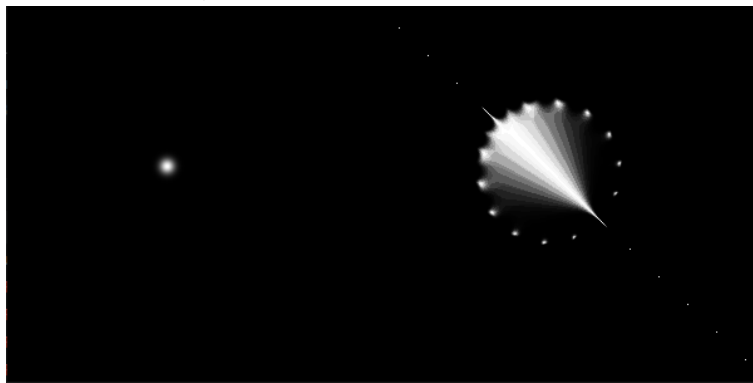


Figure 4.20: SIS simulator  $n = 14$

---

```

(term1, term2) ← (0, 0)
for m in [1:n] do
    Accumulate terms with the sum of equation 2.4.
end for
(x', y') = the first term on the rhs. of 2.4
           minus the fraction multiplied by the accumulated sum.
Return (x', y')
```

---

Figure 4.21: Algorithm for finite sum point mass model

be  $\chi/2$  and using wolfram alpha, the derivative of  $\alpha_1^2$  and  $\beta_1^2$  when simplified and multiplied with  $C_+^+$  gave the same answer as the second line in figure 4.16. Using the same method,  $\alpha_0^1$  and  $\beta_0^1$  was verified as well. This verifies one iteration in each “direction”, and the algorithm can be assumed to be correct.

To verify the process of simplifying, saving, loading and parsing the expressions, two sets of expressions for  $m < 10$  was generated in Python. One using no simplification and one using the *factor* function. They were evaluated with some arbitrary values of  $X$ ,  $Y$ ,  $\chi$  and  $\Gamma$  namely 10, 11, 12, 0.5 and the outputs were printed to the console. Then the expressions were saved to a file, loaded and parsed in C++, then evaluated with the same values. The outputs from both Python and C++, simplified and not simplified, were exported to a spreadsheet and compared. They all gave the same output and the saving, loading and parsing part could be said to be validated.

## 4.5 Finite point mass model

Quite late in the semester, through a discussion with the supervisors, the cosmologist realised that the finite summation version of the point mass model could be useful to implement. In other words, implement equation 2.4 with  $m$  going to some finite number  $n$ . On the surface, this appeared be less time consuming to implement than the spherical model, so it seemed to be a sensible area to prioritise. This model could visualise the typical traits of roulettes with the ring of spurious images with  $r < R$  in which the mapping converges and the improvement of the mapping as the number of terms increases as described in figure 6.2 from [1]. At this point, the development of the Qt graphical user interface had come quite far along, so this model was implemented directly in the GUI. Using equation 2.4, algorithm 4.21 was implemented. Figure 6.1 shows the output of the simulator with  $n = [1, 2, 3, 4, 5, 6, 30, 100]$ .

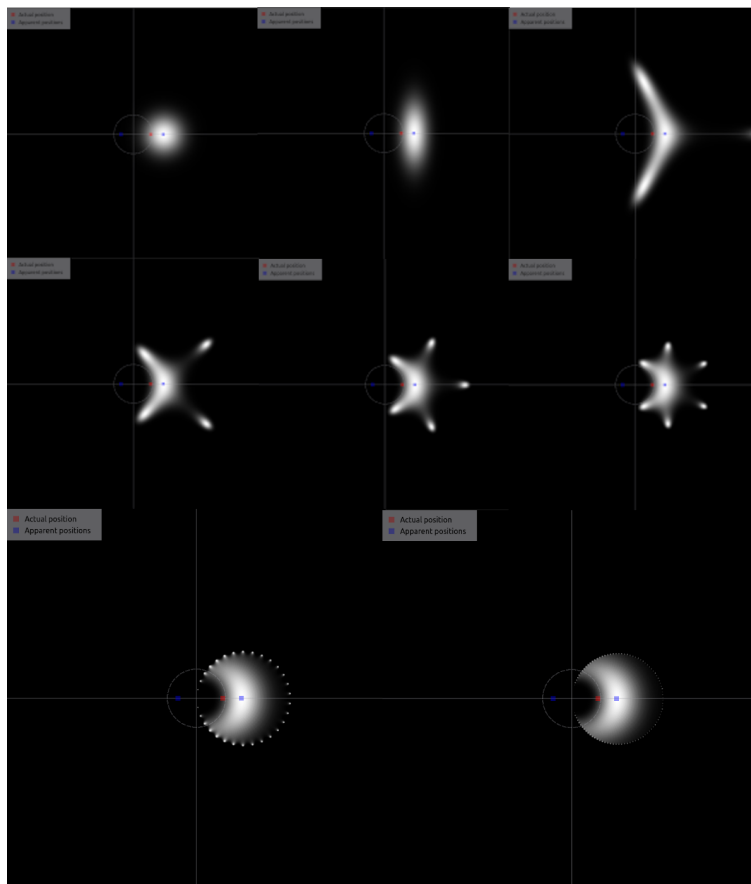


Figure 4.22: Simulator, finite point mass model



# Chapter 5

## Results - Machine Learning

This chapter will highlight the findings related to the second goal presented in 1.2;

*“The second goal is to use machine learning (ML) to analyse images with gravitational lensing. By training a ML-model, the goal is to identify the lens’ mathematical model along with its parameters. As the gravitational lensing effect is a function of the matter’s mass, we can determine the mass based on the lens model. The location can be determined from analysis of red shift. Thus, a map of the (indirectly) observed dark matter can be created.”*

### 5.1 Machine learning tasks

There were several approaches to the machine learning. This thesis has singled out two of them. One where we assume that the lens’ position between source and observer ( $\chi_L/\chi_S$ ) is unknown, and one where it is assumed to be known. Regardless of approach, the main task was to train a network to be able to estimate the parameters of the lens and the position and size of the light source given a distorted image. In the generated images, the lens is always placed in the centre. This could not be assumed in real world images. The position of the lens could be determined by observing visible matter in the lens. A neural network could also be trained to locate the centre of the lens, however this was not pursued at this time. Knowing the position of the lens would mean that the network would only have to estimate the distances to the lens ( $\chi_L$ ) and the light source ( $\chi_S$ ), the lens parameters, and size and position of the light source.

In the alternative approach, one could assume the ratio  $\chi_L/\chi_S$  is known. By measuring the red shift in the light from the observable matter in the lens, and the light from the source, one can deduct  $\chi_L$  and  $\chi_S$ . Assuming that these could be measured, it would also be useful to build a network that estimates only the lens parameters, size and position of the source given the distorted image,  $\chi_L$  and  $\chi_S$ . These two tasks became the main focus of the machine learning part of this thesis. As the exact point mass model was the only model that got implemented in time, this was the model that the machine learning focused on.

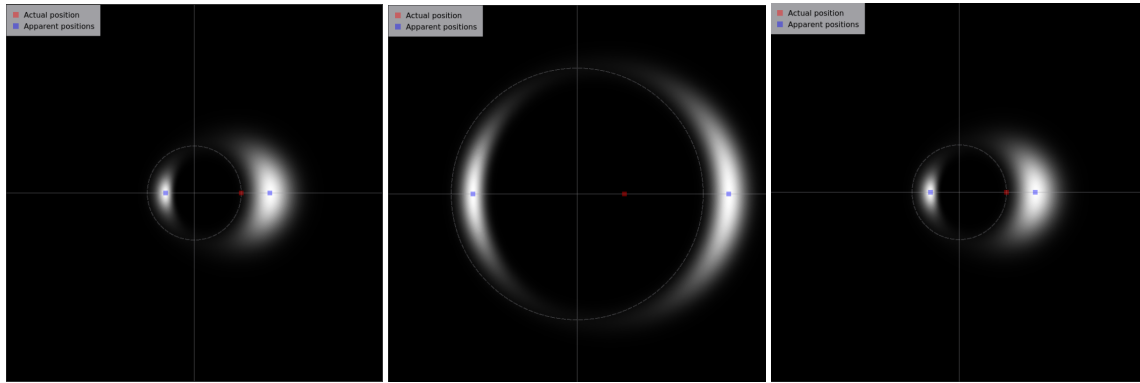


Figure 5.1: Simulator output with  $(R_E, \chi) = (30, 0.3)$ ,  $(80, 0.3)$  and  $(80, 0.8)$

## 5.2 $\chi$ vs. Einstein radius

The output from the simulator with different values of  $\chi$  and Einstein radius uncovers an interesting observation. Looking at figure 5.1, in the leftmost image  $\chi$  is 0.3 and Einstein radius is 30. Increasing the Einstein radius to 80 in the middle image distorts and blows up the image considerably. Increasing  $\chi$  to 0.8 seems to be getting us right back to the first image. Increasing  $\chi$  seems to have the exact opposite effect as increasing the Einstein radius. This begs the question: is it at all possible to determine one of these parameters not knowing the other one? This question could probably be answered mathematically, but this is beyond the mathematical knowledge of the group. It could also be somewhat answered programmatically by analysing the matrices themselves and see if any two different sets of values can produce the same image, or at least very similar images considering rounding errors etc. The third, and preferred way to solve this is to ignore the problem for now and go on and train the network on the images and see if it approaches the correct values.

## 5.3 Generate and import data

In order to train the neural net the group needed to generate large amounts of data. The model parameters was randomly generated for each image. For this purpose a script was created. The script needed to be able to generate hundreds of thousand of images, in other words it needed to be efficient and fast.

The data generator takes in arguments using a parser. Number of images to generate, resolution of the images, and folder name is input when running the script. Using this method of passing arguments to the parser is practical when starting the data generation automatically from a separate Python script used for training the neural net.

A random number generator was used when creating the images. It was important that the numbers generated were not based on the same seed every time, since this would lead to the training set being identical to the testing set. To achieve this randomness, the function `random_device()` [20] from the C++ standard library was used to generate a non-deterministic seed. The seed was then used to power a random number engine [21] based upon the Mersenne Twister algorithm [22].

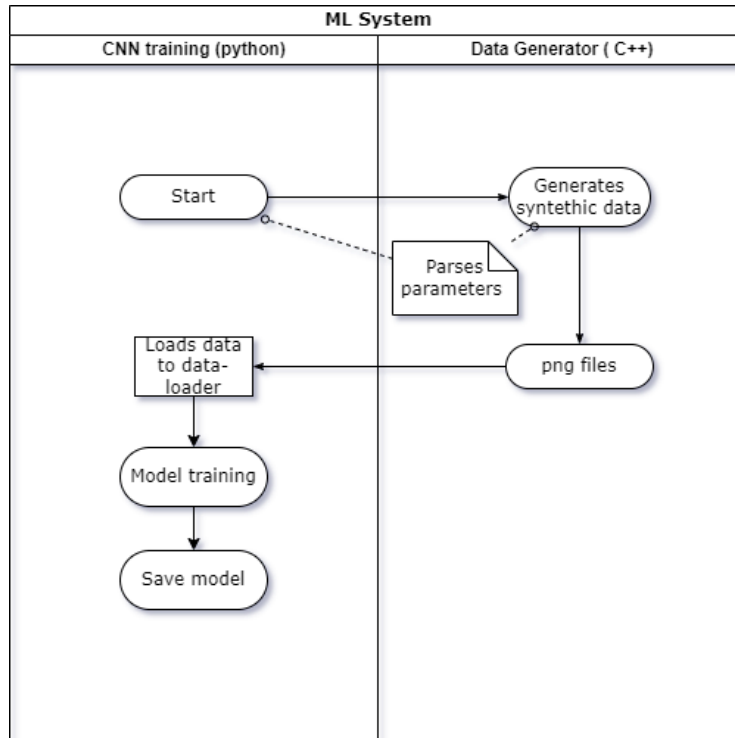


Figure 5.2: Program flow

The data format used for storing the images was the PNG file format. This file format was well suited for this task mainly because it offers highly efficient lossless compression. It is also a popular format that is well supported by various APIs used in this project like PyTorch and openCV.

Each image generated also had information about the model parameters used when generating the image. This information also needs to accompany the image files when used for machine learning. For large data sets this information is often contained in a csv or json file. However the group chose to contain the information in the filename of the images. This allowed for easy and quick manual inspection of the generated images.

To get the variable  $\chi$  to be approximately in the same order of magnitude as the other variables, a percentage value was used changing the range from  $[0, 1]$  to  $[0, 100]$ . For the remainder of this chapter, whenever  $\chi$  is mentioned, it is the percentage value that is being referred to.

## 5.4 Training models

To get started on the machine learning, a training script was written in Python using PyTorch. A basic network consisting of a few convolutional layers with max pooling, and a couple of fully connected layers was created. After experimenting with different loss functions and optimisers on the basic network, unsurprisingly the mean square error loss function and Adam optimiser gave the best results, so these were mostly used from here.

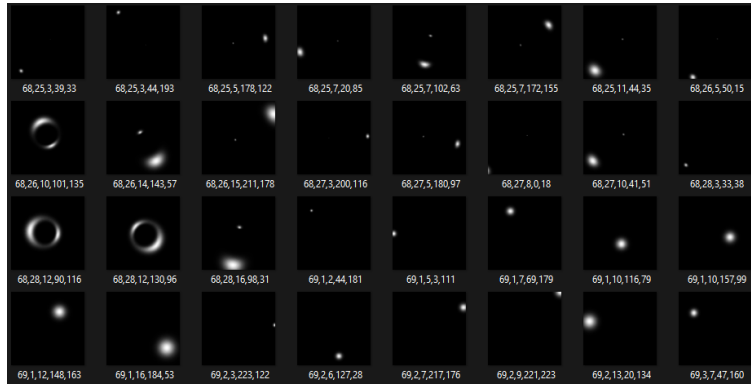


Figure 5.3: Images generated with lensing parameters shown in the filenames

### 5.4.1 First iteration: Three parameters

The first approach was to start with the basic network. The idea was to keep it simple and as computationally inexpensive as possible so that the iterative process of finding the best parameters would be fast, and then add complexity as we went along. To begin with, the training data had a resolution of 256x256 pixels and three parameters were estimated by the network, namely actual position (on the x-axis), Einstein radius and source size, the y coordinate and  $\chi$  was not implemented yet. After a few days of experimenting, and still with a quite simple network, with a somewhat modified yet simple network, a mean squared error as low as one figure was accomplished.

With such low resolution, there were some artefacts showing up on the source in the image as can be seen in figure 5.4. There seems to be slightly darker lines going over the light source and the luminosity changes step-wise. The convolutional network can probably take advantage of this when estimating the parameters, so these results should probably be taken with a grain of salt.

### 5.4.2 Second iteration: Increasing image resolution

To avoid this problem the resolution of the images were increased to another nice power of two; 512. This decreased the visibility of the artefacts quite a bit. When training on images with resolution of 512x512 pixels the results were nowhere near as good. Obviously by doubling the size of the image, you double the range of the parameters as well and probably double the errors and hence quadrupling the squared errors. The results, however were much worse than quadrupled, the loss were in the hundreds. This could imply that the network in fact did take advantage of the artefacts on the image at lower resolutions.

### 5.4.3 Third iteration: AlexNet

After some initial experimentation with simple networks it was time to try something more advanced. AlexNet was implemented. AlexNet is traditionally used as a classification network. In other words it was intended to recognise and classify object like cats and dogs. But for the purpose of estimating the lens parameters, the network needed to perform regression, not clas-

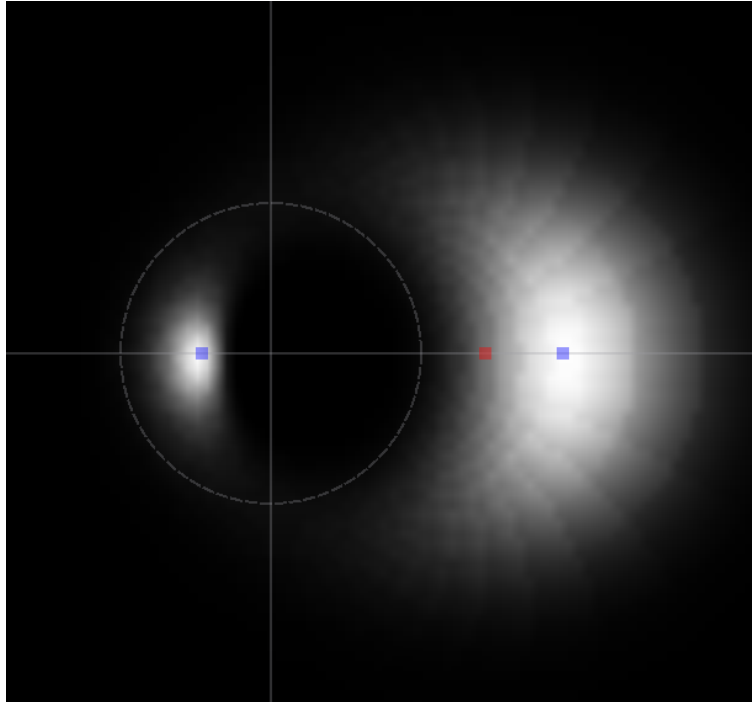


Figure 5.4: Artefacts at low resolutions

sification. AlexNet was easily adapted to this task. The classification step traditionally used in AlexNet was skipped, the input layer was changed from three channels (RGB) to 1 channel (grayscale) and the network was set up with the lensing parameters as direct output nodes. As this project has an “infinite” amount of training data, over-fitting did not seem to be an issue, so to speed up the training time, training a model without the dropout layers was tested. This produced much better results, so the drop out layers were omitted. Implementing AlexNet improved performance considerably compared to the previous network.

#### 5.4.4 Fourth iteration: Five parameters

When the simulator had been developed to be able to use the variable  $\chi$  as well as  $y$ -position. The data set generator was changed to generate data sets with random values for all five parameters:  $x$ - and  $y$ -position, Einstein radius, source size and  $\chi$ . This was then implemented into the machine learning model as well. Predicting five variables rather than predicting three with the two remaining being kept constant seemed to be a much more difficult task for the CNN and the mean square error could not be improved beyond 100. Statistics for this model is discussed in section 5.4.8.

#### 5.4.5 Fifth iteration: Improving training data

On closer inspection of the training data it was noticed that some of the images had the light sources fully or partially outside of the frame. This was caused by the constraints of the five parameters  $\chi$ , Einstein radius, source size,  $x$ -position and  $y$ -position being too loose. This resulted in parts of the source being outside of the frame. This could appear to be part of the reason for

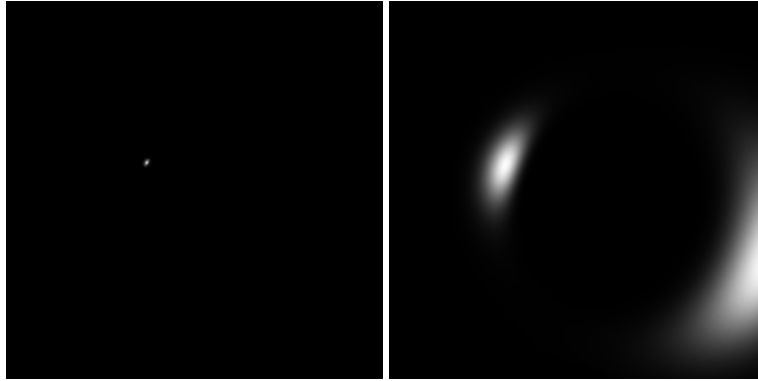


Figure 5.5: Hard to predict

the poor results. To avoid this,  $\chi$  was constrained between 30 and 100, the Einstein Radius and the source size was constrained to a tenth of the image size. This ensured that the source would rarely be touching the edge or appear outside of the image.

Later in the training it was noticed that some particular data points had terrible loss. When investigating the data again it was clear that with extreme values of  $\chi$ , Einstein Radius and x- or y-position, the source would still be barely or not visible at all. To improve this, the constraints for  $\chi$  Einstein Radius and source size was kept, and in addition the x- and y- position was being constrained to be at least 100 pixels from the edge. This resulted in much more consistent images and better training results.

It was decided this was a reasonable thing to do as one must assume that a complete image of the source is available. Figure 5.5 shows two typical images that would be hard for the network to predict. The leftmost one looks simple enough but the source is actually down to the right and the small dot is the secondary image. The rightmost one is not terrible, but some info is lost as some of the source is outside the frame.

To check if there were other values that gave unreasonable images, the network was tested with a thousand images. The loss was saved in a list together with the corresponding image, parameters and output. The list was sorted by loss from high to low, the images was plotted and the data was printed to console 5.6. The images with the highest loss looked quite normal, with the source well inside the frame, the only thing they had in common was a relatively low value for  $\chi$  (the first parameter in the list with range 30 - 100). This suggested that there were no unreasonable images with extreme prediction errors, so the variables did not need further constraints.

#### 5.4.6 Iteration six: Assuming that $\chi$ is known

Assuming that  $\chi$  is known, it would be useful to be able to estimate the four remaining parameters. To be able to do this a multi modal or multi input network would have to be implemented. This was not something any of the group members was familiar with, so this was put off for the moment. A simpler task that would be somewhat similar was to set  $\chi$  to a constant for all data points and predicting the remaining parameters, ignoring  $\chi$ . To implement this, new train-

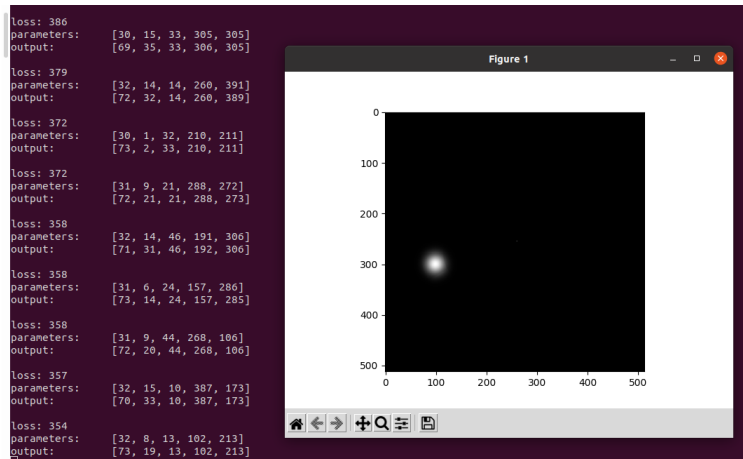


Figure 5.6: Some of the worst losses with corresponding parameters and output

ing data with constant  $\chi$  was generated and the output layer of the network was changed to 4 outputs. This was a much simpler task for the network to predict and it was trained to achieve a mean square error of 4 in a couple of hours. Some statistics for this model is discussed in section 5.4.8.

### 5.4.7 Iteration seven: Multi modal network

The great results of the four parameter network, could imply that if a network somehow had knowledge of  $\chi$ , it would get similar results for all values of  $\chi$ . An attempt of making a multi modal network was made. The implementation was not as difficult as expected. The code had to be changed to take  $\chi$  as a separate variable which would be taken as input to the AlexNet together with the image.  $\chi$  was implemented in the network by concatenating  $\chi$  with the output from the convolutional part of the network before feeding it into the first fully connected layer. This is displayed in figure 5.7. It was not obvious to us that this would work at all, even less that one out of the 9217 inputs to the fully connected layer would be able to influence the network enough to get decent outputs. The results, however, were great. After just a few hours of training, the mean square error of the network had dropped below 2. Section 5.4.8 shows the statistics for this model.

### 5.4.8 Looking at the statistics

Figure 5.8, 5.9 and 5.10 shows the distribution of the errors of the parameter predictions using the latest models. It uses data sets of 10000 samples and is plotted as histograms with 100 bins.  $\chi$  is represented as a percentage value and the units for the rest of the variables are pixels. The y-axis represents number of samples in each bin. To clarify: the range of  $\chi$  is [30, 100], the range of  $R_E$  and source size is [1, 51] and the range of x- and y-position is [100, 411].

In the five parameter case, as expected, the source size and position was estimated quite well. It would be interesting to compare the distributions with the distributions one would get by guessing the middle of the range each time. In this case the error would be a uniform distribution with a mean of 0, and the standard deviation could be calculated using equation 5.1. This would give

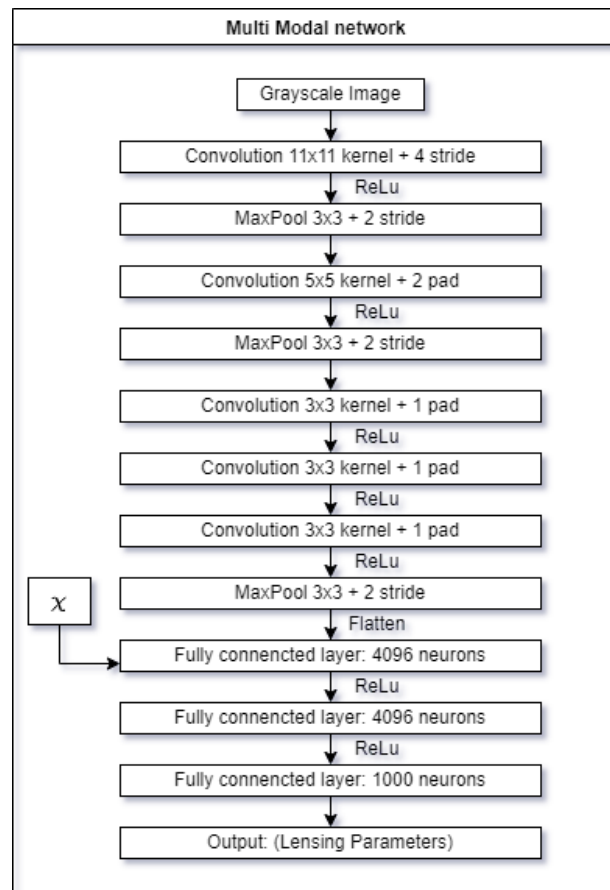


Figure 5.7:  $\chi$  is inserted directly into the network



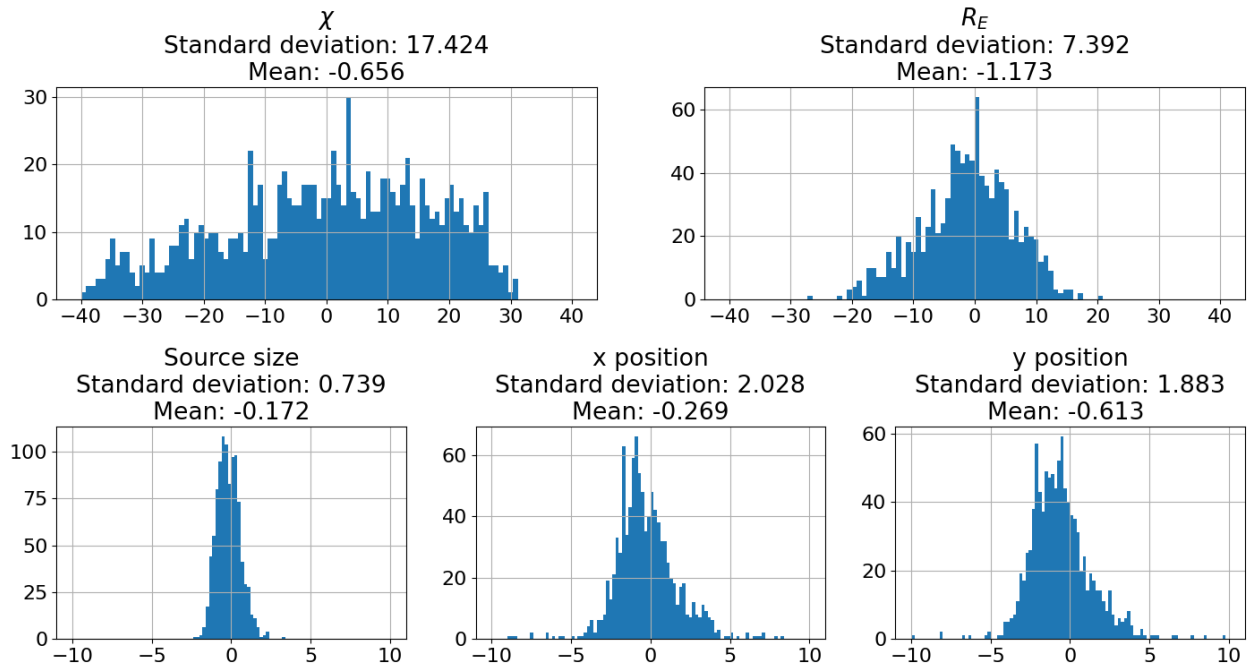


Figure 5.8: Distribution of the errors of the five parameters

a standard deviation of 90.1 for x- and y-position and 14.4 for source size. A standard deviation of 2 and 1.9 for x- and y-position and 0.7 for the source size is a great improvement.

$\chi$  and  $R_E$ , however were not as accurate. The standard deviation in the case of guessing the middle of the range would be 20.2 for  $\chi$  and 14.4 for  $R_E$ . The results of the CNN of 17.4 for  $\chi$  and 7.4 for  $R_E$  is not as impressive, but still an improvement. It is unknown whether this improvement is a result of the network being able to predict the variables independently. This is considered more thoroughly in the discussion chapter.

In the case of the four parameter and the multi modal network, the results is greatly improved as expected from the significantly decreased mean square error.

$$\sigma = \frac{b-a}{2\sqrt{3}} \quad (5.1)$$

## 5.5 Inception3

A version of the Inception3 network was implemented as well. As this is a classification network for RGB images just like AlexNet, the input layer was changed from three channels to one and the output layer was changed to 4 or 5 depending on the number of parameters to estimate. The original Inception3 network has some functionality that the group was not familiar with so the network had to be slightly modified to be able to use it in a way that the group was used to. This network is much larger than the previous networks and therefore used more memory and was slower to train. After some experimenting with this network it didn't seem to get as good results

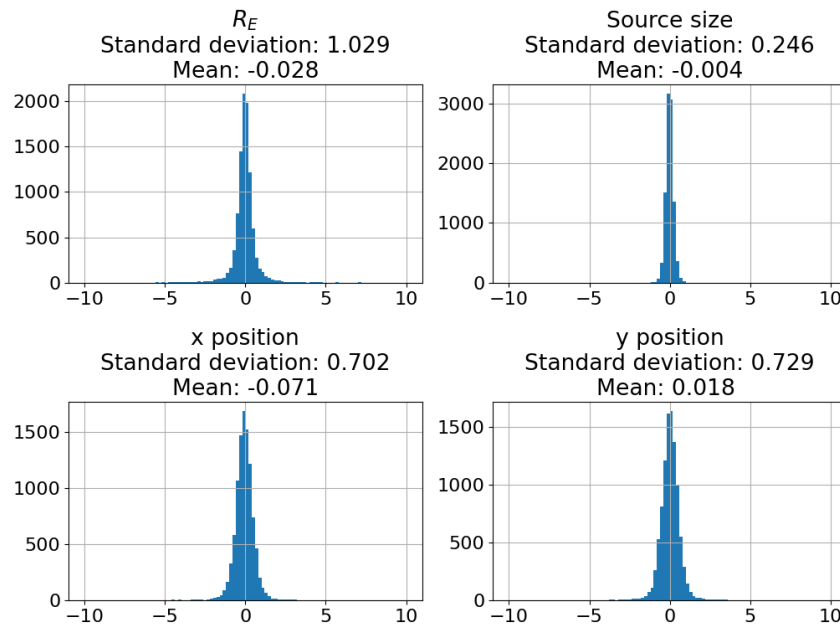


Figure 5.9: Distribution of the errors of the four parameters

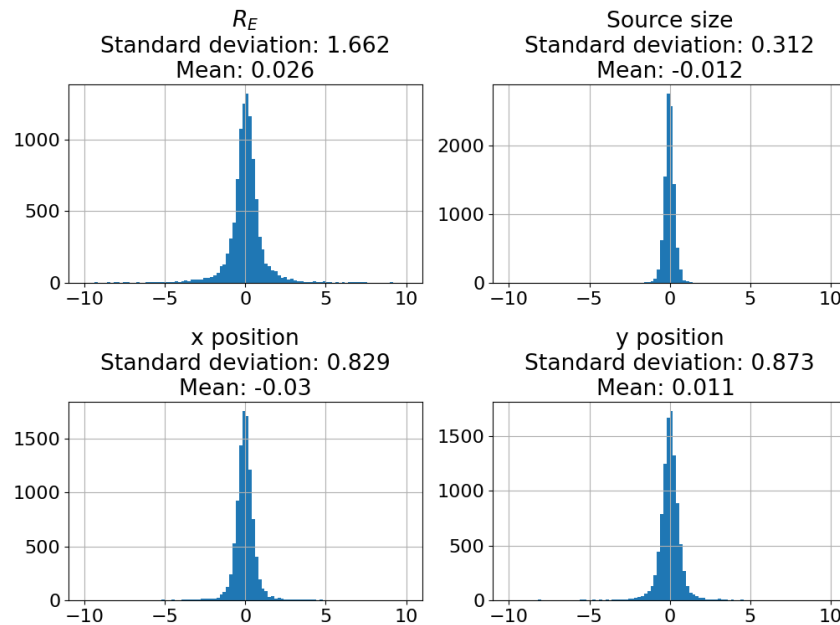


Figure 5.10: Distribution of the errors of the four parameters using multi modal network

as AlexNet, so this architecture was shelved for now.

## 5.6 Scheduler and Learning rate

The learning rate when training the network is controlled by a scheduler. Generally one would start out with a high learning rate so that the network learns quickly. And the loss function quickly approaches the global minimum. After some time of training the progress will reach a plateau. The plateaus can be overcome by reducing the learning rate. This will shift the training more towards the exploitative end of the scale rather than the explorative end.

The scheduler used in this project would detect if there was no improvement for a given amount of epochs, and then decrease the learning rate by a given factor. This given number of epochs is called *patience*.

After some experimentation these parameters was set to the following.

- Initial learning rate: 0.01
- Patience: 5
- Scaling factor: 0.5

Please note that the ideal setting for these numbers can variate depending on things like size of data set and number of epochs run.

# Chapter 6

## Discussion

This chapter will discuss and reflect around the project's organisation, as well as the results presented in chapter 4 and 5. It also covers some reflection on how the group's prerequisites were relevant for the bachelor thesis' learning outcomes.

### 6.1 Project organisation

This being a large group of four members made project organisation extra important. Every Monday morning the group had a meeting to discuss what needed to be done, and who was going to do it. Of course, there was also much informal communication within the group on a daily basis. The agile method of working worked well and allowed for quick planning and changes when needed. Parts of the development of the machine learning and the GUI depended on the implementation of a somewhat working simulator. The job of implementing the simulator was quite difficult to split between several group members. This caused some delays in the beginning of the project. However, when we had a working implementation of the simulator, work on the GUI, simulator and ML part of the project was easily divided between the group members.

### 6.2 Prerequisites and learning goals

In the startup phase of this project, the group members and some of the teachers were slightly concerned that the learning outcomes for automation and robotics engineering would not be met. It was recommended that the group made sure to focus on the engineering parts of the assignment, and not get lost in the advanced physics and maths.

The project covered several different areas deeply connected to what the group had learned during previous semesters. The group was able to utilise much of this knowledge in this project.

- A good understanding of mathematical modelling of physical systems
- Understanding different coordinate systems and projections

- Having a good understanding of general programming in Python and C++.
- Basic knowledge of machine learning
- Knowledge of convolution and image- and signal-processing
- Some knowledge and intuition of statistical methods for analysing the results
- Previous experience with Opencv both in python and C++

The members of the group all learned a great deal working on this project. This includes areas such as simulating physics, creating a GUI, machine learning, calculus, linear algebra, statistics and project organisation. As well as general programming experience in C++ and Python. Learning about astrophysics and the universe was an added bonus.

Key learning experiences from this project:

- Implementation and testing of convolutional neural networks.
  - Python programming
  - Learning to use PyTorch
  - Understanding neural networks
  - Analysing statistics to understand the results
  - Generating and handling large amounts of data. Learned about file-systems and storage.
  - Operating Linux systems(IDUN) remotely using SSH protocol.
- Interpreting and implementing advanced astrophysics theory into running code.
  - C++ programming:
    - \* Builds and compilers
    - \* Multithreading
    - \* OpenCV
    - \* Cross-platform support
  - Understanding scientific papers and mathematical notation
  - Analysing results
- Creating a GUI application
  - Qt framework
  - Implementing user feedback
- Project organisation
  - Using Github

- Agile development

## 6.3 Simulator

### 6.3.1 Point mass model

Looking at the reasonably simple equation 2.5, implementing the simulator for the exact point mass model seemed to be quite a simple task. Learning that there were three different coordinate systems involved, in two different planes, made the task a bit more difficult. The task of translating between these coordinate systems and between matrix indices and coordinates, at the same time as scaling due to the coordinate systems being in different planes became quite complicated and error prone. The first prototype included many of these “translation errors”. Also this version ran quite slow. Despite this it served as a proof of concept, giving an indication that we were on the right track.

Being very careful and splitting the translations up in steps during implementation and debugging, finally resulted in the first correct implementation as described in 4.2.2. This version was an important milestone in the development of a working application. The issues from the first prototype with lagging visualisation and slow calculations were removed, indicating that a switch from Python to C++ was a step in the right direction. The prototype proved that a user friendly application was a feasible goal within reach. It also proved that the algorithms and visualisation could run in real time as the user changed the variables. This version was mathematically correct, but it was far from complete. The biggest shortcoming was that it did not have the option of varying  $\chi_L/\chi_S$ , which places the lens and the source on the same plane, something that is not realistic. In reality the source and lens would be far apart. Another shortcoming was that the simulator did not support source displacement in the y-direction.

Each version improved the simulator by adding features and improving the algorithms. All the shortcomings of the previous versions were addressed and mended. Improving and debugging the code was a great learning experience both in mathematical modelling and visualisation, general programming and in the quirks of the low level programming language C++.

Due to the already established work on the exact point mass model, the implementation of the finite point mass model was just a matter of replacing equation 2.5 with a function implementing equation 2.4 using a for loop. Despite being relatively simple to implement, this model gave useful insights to the properties of Roulettes.

A comparison of the outputs of the finite point mass model with  $n = [1, 2, 3, 4, 5, 6, 30, 100]$  to figure 6.2 from [1] suggest that the implementation is correct. Comparing the output of the finite point mass model summing up to  $m = 100$  with the output of the exact point mass model in figure 6.3, we can see that they are virtually indistinguishable within the region where  $r < R$ . This strengthens the confidence in the Roulette formalism as well the confidence of both implementations.

The task of implementing the point mass model, as implemented in the GUI, both exact and finite, could be said to be complete.

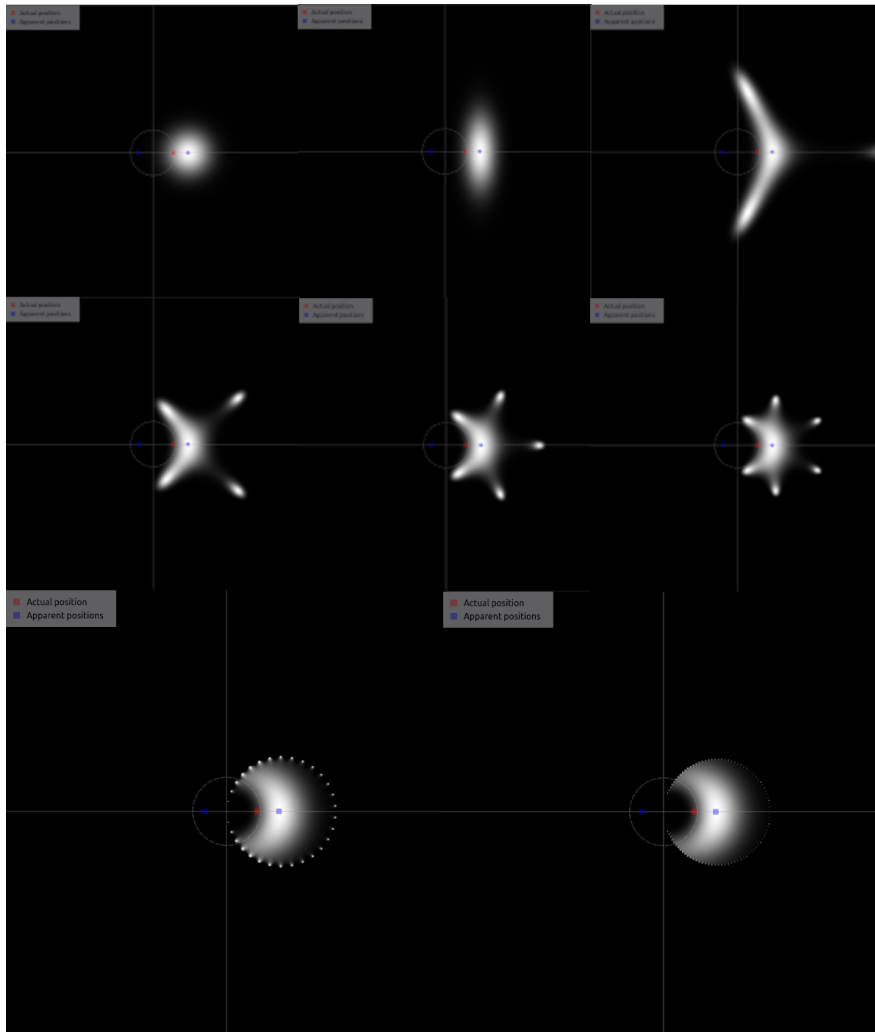


Figure 6.1: Simulator, finite point mass model

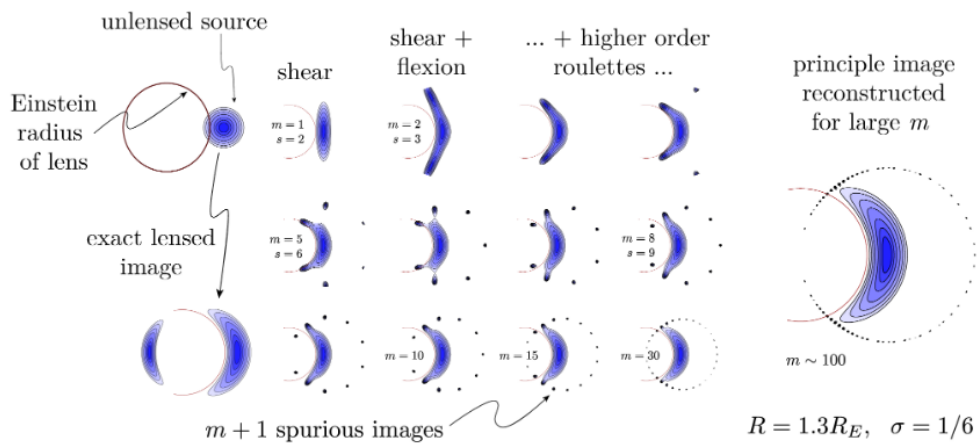


Figure 6.2: Illustrations of Roulette lensing as presented in Clarksons work. Credit: C. Clarkson [1]

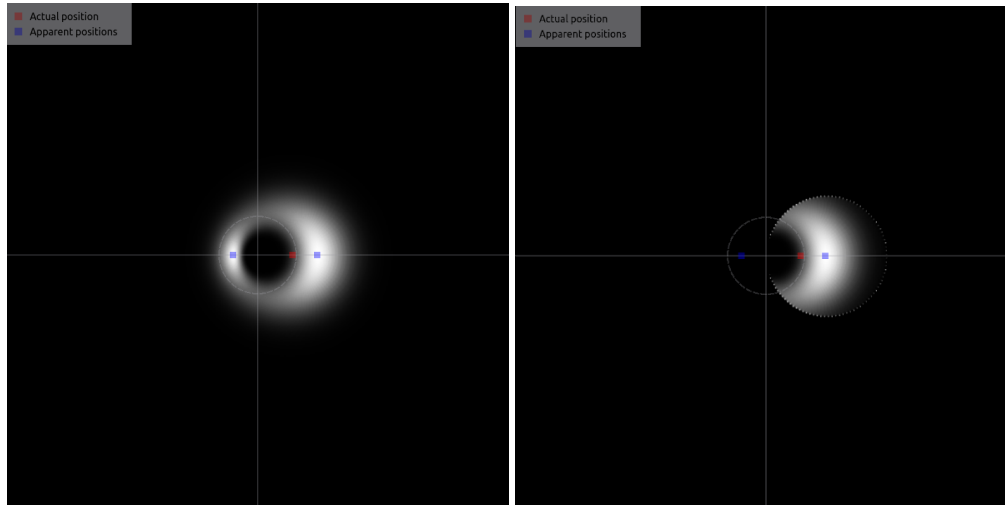


Figure 6.3: Comparison of the output from the exact and the finite point mass model summing up to  $m = 100$

### 6.3.2 Spherical model (SIS)

After some research on the subject, the investigation of convergence within the area where  $r < n/3$ , was deemed somewhat redundant. The literature states that the mapping in fact converges within the area where  $r < R$  [1]. This somewhat aligns with our own findings, which is a reassuring sign that the equations has not been misinterpreted during implementation. Even though the subject was covered in existing literature, the findings are included in 4.4 as it was not immediately obvious and led to some interesting discussions within the group and with the supervisors.

Despite the implementation of the SIS-model showing some unexpected output, a lot is learned from implementing it. It has been shown that it is possible to generate amplitude expressions using symbolic maths libraries up to  $m = 100$  in a reasonable amount of time, however these expressions have not been fully verified. It has also shown that it is possible to transfer these expressions from python to C++ without loss of data. It has shown that an image of reasonable resolution can be distorted using terms up to  $m = 50$  in about a few seconds or less. The group is quite convinced that it is possible to have this implementation working correctly if the reason for the incorrect distortion is found and corrected. When the SIS-model is finished, it would be a natural next step to implement it in the GUI.

For the point mass models, all the variables involved (in equations 2.5 and 2.6) are defined in the lens plane and the source plane except from  $\chi_L$  and  $\chi_S$  which is the distances to the lens plane and the source plane respectively. Luckily we are only interested in the ratio between these, which are dimensionless, so the magnitudes and units are irrelevant. Looking at equation 2.5 knowing that this ratio is dimensionless, it is apparent that both sides are vectors in the plane with arbitrary length units. This enables us to make our own scaled down coordinate systems with pixels as units. For the isothermal sphere however the translation between the real world and the pixel world is not as apparent. Equation 2.7 and 2.8 refers to the variable  $\chi_L$  which is the actual distance to the lens (which can not be defined in pixels). As the rest of



the  $\alpha$  and  $\beta$  terms are combinations of derivatives in different directions, and multiplied with  $\chi_L$  a different number of times, it is very difficult to keep control of the dimensions. In the implementation, all the variables are defined with pixels as units except from the  $\chi_L$  variable which use the ratio between  $\chi_L$  and  $\chi_S$  which is probably wrong. To get the proper input and hopefully a reasonable output, it is highly likely that a proper dimensional analysis would be a key in solving the problem.

## 6.4 Machine learning

### 6.4.1 Possible to predict both $\chi$ and $R_E$ ?

The statistics showing the performance of the network in figure 5.8, shows a clear improvement over guessing the middle of the range. This could imply that it is possible to train a network to predict the parameters to high accuracy. However the results could be misleading. Figure 5.1 shows that it is difficult to predict the exact values for  $\chi$  and  $R_E$  as the effect of changing one could be more or less cancelled by changing the other. Some relation between the two is probably easier to predict. It is possible that the network can predict some relationship between the two parameters quite well, resulting in guessing somewhat arbitrary values for  $\chi$  and  $R_E$ , but with the correct “distance” or ratio between them, is enough to account for the improvement seen in figure 5.8. If this is the case, then this network is no more useful than the four-parameter network. Whether this is the case could probably be proved analytically, but it is beyond the statistics knowledge of the group. However, it can be analysed numerically: let’s assume for example that it is impossible to predict both parameters, but what the network actually does is predicting the *distance* between the two parameters perfectly, than it guesses at the midpoint between the two midpoints, but shifted to get the correct distance.

By generating uniform distributions of a million numbers within the ranges of  $\chi$  and  $R_E$  respectively, and guessing the middle of the range for each of them produces the plots at the top of figure 6.5. Guessing at the midpoint between the midpoints of the two ranges, but offset by a known “distance”, produces a better distribution as can be seen in the plots at the bottom of the same figure. To clarify, these are plots of the distributions of the errors one would get by having no knowledge of  $\chi$  and  $R_E$  and a possible distribution one would get if the network can not predict the exact values but rather can predict the “distance” between the two variables. We can see that this improves the standard deviation quite a bit and get similar results to our results of 17.4 and 7.4 from figure 5.8. This result does not prove whether it is possible or not, but it suggest that the network in fact has a hard time predicting the actual values of  $\chi$  and  $R_E$  and rather makes an educated guess based on some well predicted relation between the two. Also, the network can have other “strategies” to guess the parameters more accurately like for example estimating the *ratio* between them and make a guess based on that. The question remains open.

### 6.4.2 Building networks from scratch

The task of building our own convolutional neural network from scratch was a very interesting and great learning experience, giving insight to the actual workings of the networks. Improving the network by increasing the depth and width as well as tweaking parameters such as kernel

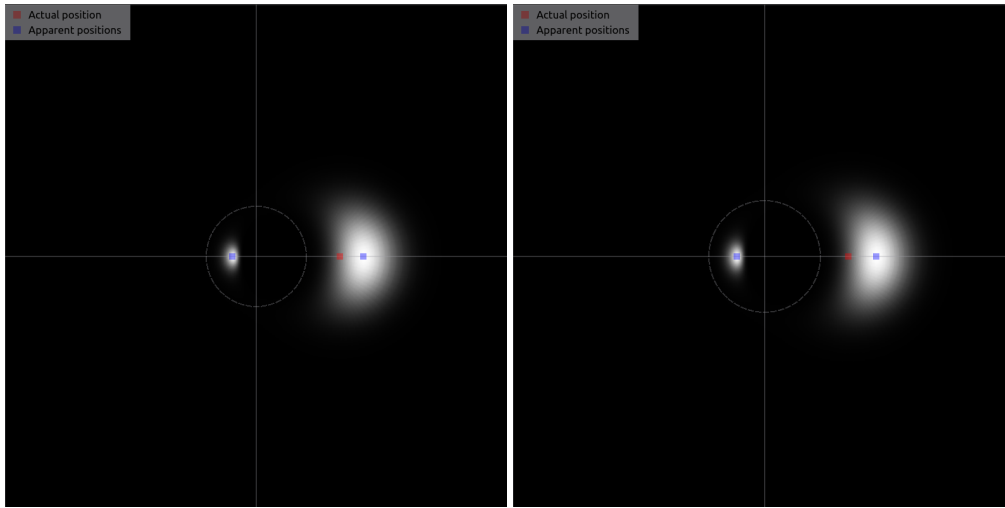


Figure 6.4:  $(\chi, R_E) = (30, 50)$  and  $(40, 60)$

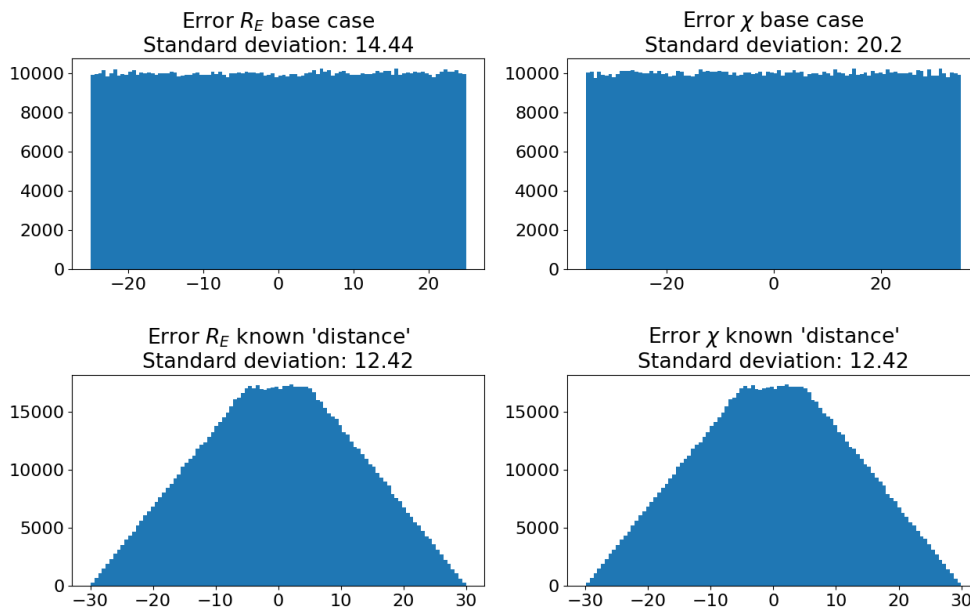


Figure 6.5: Possible "best guess" distribution of  $\chi$  and  $R_E$  if the network is able to predict the 'distance' between variables perfectly

sizes and stride lengths of the convolutions was a very interesting, but time consuming, process.

When AlexNet was implemented and produced better results, this approach was abandoned. However, using this approach to improve or tailor a network like AlexNet to our purpose could be a worthwhile effort.

To speed up this process, and better keep track of which changes improved the network and which did not, a better training framework should be made. The CNN should be rewritten to be able to change parameters such as number of layers, kernel size etc. by taking them as input rather than hard coding them. The training script should automatically export all info about a network and it's performance to a spreadsheet.

### 6.4.3 AlexNet

The results achieved by using AlexNet were very impressive. Especially in the case of estimating  $R_E$ , the source size and position, given  $\chi$ . The group is convinced that this approach could be used in the real world, at least for the exact point mass model. Omitting the drop out functionality is acceptable when using ideal synthetic data generated in a controlled environment. However, if this model would ever be used on real world images, the drop out layers should probably be used to make the network more robust.

### 6.4.4 Inception3

Some functionalities were removed to simplify the implementation. It is not certain whether this simplification caused a performance decrease in the network, or even introduced some sort of unknown bug to the system. This could be an interesting topic for further research. For our purposes, a network as big and advanced as Inception3 was considered excessive. However, for more complex images from the real world it could perform well as many stones were left unturned.

### 6.4.5 Other architectures

Both AlexNet and Inception3 are specialised object detection networks. It is beyond the knowledge of the group, but it can be imagined that there are other architectures that would fit regression tasks better.

### 6.4.6 Image resolution

512x512 pixel images were used when training and testing the neural networks. This certainly provided images with high amounts of details. As mentioned in the results chapter, using lower resolution produced some artefacts in the image that the network possibly takes advantage of. To get lower resolution images without getting these artefacts, a solution could be to generate images with high resolution and then down sample them before using them for training. Using lower resolution would probably not give any better results in terms of accuracy. However, it

would certainly increase training speed and lower memory requirements. This is especially true for the Inception3 model, which was experienced as very memory consuming.

### 6.4.7 Extending to Roulette images and real world applications

To have this approach working for the models based on the Roulette formalism, a few modifications would have to be done. The spurious images would need to be removed from the image and the missing secondary image would have to be addressed. A solution for both of these problems could be to only evaluate the area where the series converge (where  $r < R$ ) and make sure to constrain the parameters so that all or most of the light source is contained within this region.

There is still much work to be done before this machine learning concept can be applied to recognise real gravitation lenses in images taken with real telescopes. The images generated in this project are synthetic and ideal, containing no noise or disturbances. Real world images are usually far from ideal.

## 6.5 GUI

The goal was to produce a user interface with emphasis on user friendliness. The supervising cosmologist, given the daunting task to speak on behalf of all potential users, outlined some essential functionalities during the meetings throughout the project period. All of the requested functionalities are easily available in the main window. Some of the more niche functions are placed in the drop-down menus, giving the application a sleek, user friendly layout. The final product was approved by the supervisors and no further functions have been requested.

Some additional features had to be left out due to time constraints or low priority. The first suggestion is to save user settings between sessions. It would be a nice-to-have feature for the user to be able to go back to where they left off, but it is not strictly necessary as of the current state of the application. Another suggested feature was the ability to upload a custom source image which would be a challenge to implement as it would need to be scaled appropriately and centred by centre of mass.

The performance of the application is good on most systems at image resolutions around 600x600, which is the default. Performance noticeably worsens at higher resolutions. This is mostly due to the actual size of the image being doubled to avoid the simulator trying to map pixels from outside the image, and then being cropped back to the chosen resolution. This means the pixel count the simulator is working with is four times higher than what it is set to. However, the default resolution of 600x600 is more than enough for most use cases.

Some other possibilities for optimisations in the application have been explored, but not implemented. This is due to the expected performance improvements being too low to prioritise. It could however have some significance if the software were to be expanded in the future.

At the moment, the grid and legend are re-calculated and re-drawn every time the image updates in the simulator. This could instead only be done once and then saved to a *QPixmap*

which could be drawn on top of the images on every loop. They would then only need to be re-drawn if the window size is changed.

## 6.6 Client feedback

Our supervising cosmologist is the author of the assignment description, in appendix A. As of this, it is natural to request some feedback from him to serve as a gauge on how successful the finished product actually is in the eyes of a cosmologist. He was asked to give a brief quote regarding this thesis and replied as follows;

*“The work reported on in this thesis far exceeds what I imagined possible for a group of bachelor students to do over the course of a single semester, whether it be due to my pessimism or their skill, or both. Surely, we were only a week away from cracking the SIS-profile, with machine learning and all. . . ! It has been a pleasure to work with the authors (with annoyingly similar names!), their attitudes all along having revealed an interest in the problems at hand and in finding good solution strategies. Moreover, the way the problems have been solved, through proper implementation with apt tools, in itself constitute an enrichment of my perspective on how to go about further problem solving in this field. The simulator is truly outstanding (as Clarkson put it: “I love it!”), and the machine learning bit is interesting. In ending, I find their work impressive.”*

— Ben David Normann, cosmologist at NTNU

From his feedback it is clear that the application was a success. Both Normann and Clarkson has been able to use the application without specific instructions from the group. This gives a good indication that the project’s first goal, where emphasis on a user friendly application was a definite goal, has been accomplished. In addition, Normann could not identify any flaws with the simulated visualisations, indicating that the simulations are in line of what to be expected from both point mass closed form and Roulette formalism.

# Chapter 7

## Conclusions

This chapter will serve as an answer to the problem formulation for both simulator and machine learning. As the problems given were presented as open ended goals, as a part of the bigger project's main goals, a list of remaining open problems are presented at the end of this chapter. These would be the natural next steps going forward.

### 7.1 Simulator

By creating an application capable of simulating gravitational lensing, the required data sets for machine learning could be produced. The application could also serve as a tool for researchers, used for visualising gravitational lensing during research. By showing the lens' effect in real time researchers would acquire a far more intuitive understanding of the models. The tool can also be used for education and outreach work. In this thesis, such an application is developed. The application is capable of simulating both the exact point mass lens model and the finite roulette version of the point mass lens. By utilising the agility of Python, and efficiency of C++, even complicated models can be simulated in real time. Due to the applications user interface, the user does not have to have any prior knowledge or experience with programming to use the application. Simply download the application for your operating system and run it. By using the application, the user gets a far better understanding of the physical principles behind the lensing models.

The work of implementing the singular isothermal sphere model is also well under way. However, some work is left for future work. Finishing this model would open up the path to extending this model to a singular isothermal ellipsoid. The output from the exact point mass model from this simulator has been used for training a neural network to be able to recognise the lensing parameters.

### 7.2 Machine learning

A convolutional neural network allows for fast estimation of parameters from input data. In this thesis two cases are investigated. One where the lens distance ratio  $\chi_L/\chi_S$  is unknown, and one

where it is known. In the case where the ratio is unknown, it remains uncertain if the network could accurately differentiate between the effects of  $\chi_L/\chi_S$  and Einstein radius. In the other case, where the ratio is known, it is shown that the network can produce accurate predictions of the lensing parameters. This is a more realistic approach for most lenses, as red shift analysis can be used to determine these distances.

In this thesis, only data generated from the general point mass model has been used for the machine learning. The hope is that a similar approach could be used for data generated by the Roulette formalism, both point mass, singular isothermal sphere and singular isothermal ellipsoid.

### 7.3 Open problems

Throughout the course of this project, several problems were discovered. Alas, not all of them could be solved within this project's timeline. In the following bullet points, the open problems are listed and we encourage others to take these on.

- Finish the implementation of singular isothermal sphere model in the simulator application.
- Expand singular isothermal sphere model to singular isothermal ellipsoid.
- Modify the simulator to make datasets using the Roulette formalism.
- Modify the neural network to be able to train on images generated using the Roulette formalism.
- Ultimately train neural network to locate and classify lenses and predict lensing-parameters, just through analysis of a real world images.

# Bibliography

- [1] C. Clarkson, “Roulettes: A weak lensing formalism for strong lensing - II. derivation and analysis,” *Classical and Quantum Gravity*, vol. 33, 2016. [Online]. Available: <https://doi.org/10.48550/arXiv.1603.04652>
- [2] N. Science, “Dark energy, dark matter.” [Online]. Available: <https://science.nasa.gov/astrophysics/focus-areas/what-is-dark-energy>
- [3] P. Schneider, C. Kochanek, and J. Wambsganss, *Gravitational Lensing: Strong, Weak and Micro*. Springer, 2006.
- [4] P. Schneider, J. Ehlers, and E. E. Falco, *Gravitational Lenses*. Springer-Verlag, 2011.
- [5] C. Kochanek and J. N. Hewitt, *Astrophysical Applications of Gravitational Lensing*. Springer, 1996.
- [6] Ø. Grøn and J. R. Kristiansen, *Rødforskyvning (Norwegian)*. Store Norske Leksikon, 2021. [Online]. Available: <https://snl.no/r%C3%B8dforskyvning>
- [7] B. D. Normann and C. Clarkson, “Recursion relations for gravitational lensing,” 2019. [Online]. Available: <https://arxiv.org/abs/1904.04471>
- [8] H. Dvergsdal, *Nevralt nettverk (Norwegian)*. Store Norske Leksikon, 2019. [Online]. Available: [https://snl.no/nevralt\\_netverk](https://snl.no/nevralt_netverk)
- [9] G. Sanderson. (2022) But what is a neural network? [Online]. Available: <https://www.3blue1brown.com/lessons/neural-networks>
- [10] K. O’Shea and R. Nash, “An introduction to convolutional neural networks,” *arXiv preprint arXiv:1511.08458*, 2015.
- [11] Y. D. Hezaveh, L. P. Levasseur, and P. J. Marshall, “Fast automated analysis of strong gravitational lenses with convolutional neural networks,” *Nature*, vol. 548, no. 7669, pp. 555–557, aug 2017. [Online]. Available: <https://doi.org/10.1038%2Fnature23463>
- [12] R. Morgan, B. Nord, S. Birrer, J. Lin, and J. Poh, “deeplensstronomy: A dataset simulation package for strong gravitational lensing,” *Journal of Open Source Software*, vol. 6, no. 58, p. 2854, feb 2021. [Online]. Available: <https://doi.org/10.21105%2Fjoss.02854>



- [13] C. Jacobs, K. Glazebrook, T. Collett, A. More, and C. McCarthy, “Finding strong lenses in CFHTLS using convolutional neural networks,” *Monthly Notices of the Royal Astronomical Society*, vol. 471, no. 1, pp. 167–181, jun 2017. [Online]. Available: <https://doi.org/10.1093%2Fmnras%2Fstx1492>
- [14] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” *Advances in neural information processing systems*, vol. 25, 2012.
- [15] ImageNet, “Large scale visual recognition challenge 2012.” [Online]. Available: <https://image-net.org/challenges/LSVRC/2012/results.html#abstract>
- [16] J. Sun, X. Cai, F. Sun, and J. Zhang, “Scene image classification method based on alex-net model,” in *2016 3rd International Conference on Informative and Cybernetics for Computational Social Systems (ICCSS)*, 2016, pp. 363–367.
- [17] J. Brownlee, “A gentle introduction to dropout for regularizing deep neural networks,” 2019. [Online]. Available: <https://machinelearningmastery.com/dropout-for-regularizing-deep-neural-networks/>
- [18] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, “Rethinking the inception architecture for computer vision,” *CoRR*, vol. abs/1512.00567, 2015. [Online]. Available: <http://arxiv.org/abs/1512.00567>
- [19] P. M. Radiuk, “Impact of training set batch size on the performance of convolutional neural networks for diverse datasets,” 2017.
- [20] “Std::random\_device.” [Online]. Available: [https://en.cppreference.com/w/cpp/numeric/random/random\\_device](https://en.cppreference.com/w/cpp/numeric/random/random_device)
- [21] “Std::mersenne\_twister\_engine.” [Online]. Available: [https://en.cppreference.com/w/cpp/numeric/random/mersenne\\_twister\\_engine](https://en.cppreference.com/w/cpp/numeric/random/mersenne_twister_engine)
- [22] M. Matsumoto and T. Nishimura, “Mersenne twister: A 623-dimensionally equidistributed uniform pseudo-random number generator,” *ACM Trans. Model. Comput. Simul.*, vol. 8, no. 1, p. 3–30, jan 1998. [Online]. Available: <https://doi.org/10.1145/272991.272995>

# **Appendix A**

## **Gravitational lensing notes by B.D. Normann**

# Gravitational lensing

A note by Ben David Normann<sup>1</sup>

<sup>1</sup>Department of ICT and Natural Sciences, Norwegian University of Science and Technology, Postboks 1517 NO-6025, 6025 Ålesund, Norway

E-mail: `ben.d.normann@ntnu.no`

September 2021

**Abstract.** This note is meant as an overview of gravitational lensing and Clarkson’s roulette formalism in particular.

*Keywords:* cosmology, gravitational lensing, dark matter, astronomy

## 1. Introduction

Today we collect information from the night sky in many different ways. The electromagnetic spectrum is read from the radio-wave band to x-ray frequencies, and gravitational-wave signals are caught on tape and analysed. However, none of these observations would be of any particular value without proper theoretical framework in which to interpret them. Theoretical development is therefore important. One such theoretical development came with the understanding that even light is deflected by gravity’s ‘pull’<sup>†</sup>. This effect is nowadays known as *gravitational lensing (GL)*.

## 2. A brief history of GL

Isaac Newton himself added a couple of lines at the end of his 1704 work on optics, about light also being influenced by gravity in the same way as matter [1]. A century later, scientists such as John Michell <sup>‡</sup> [3] and Pierre-Simon Laplace [4] would build on Newton’s theory of gravity and conclude that light must indeed be affected by gravity. Actually, Newton’s theory misses the correct lensing equation for a point mass by a factor of two, only. This discrepancy between general relativity (**GR**) and Newton’s theory of gravitation was however sufficient to discriminate between the two theories

<sup>†</sup> Actually, gravity does not exert forces on the objects on which it acts. Rather, the notion of straight lines—geodesic motion—is affected. Hence ‘pull’ and not pull.

<sup>‡</sup> This creative chap also speculated that there might be objects so massive that even light cannot escape their gravitational attraction. Today we call such objects *black holes*! An overview is provided in [2].

under a solar eclipse in 1919 [5], favouring Einstein’s theory over Newton’s. A solid framework in which to study GL-effects was hence provided: GR. The contest between the two theories also displayed the power of the tool, leading to optimism concerning its capacities. Since then, the field of GL has been vibrant with contributions.

Einstein’s calculations, however, led to pessimism about observing stars lensed by our sun [6]. It is fortunate, therefore, that Zwicky [7,8] studied lensing effects by galaxies outside our own, successfully showing how such observation could determine the mass of distant galaxies. Later, further theoretical advancement came with the Norwegian astrophysicist Sjur Refsdal, who in the 1960s [9,10] did pioneering work in using GL as a probe on cosmic parameters. By comparing the time delay between two light rays taking different routs around a lens (galaxy), the Hubble constant was successfully calculated †.

The modern field of GL is typically divided into two regimes [11, p. 21]: **weak lensing** ( $\kappa < 1$ ) and **strong lensing** ( $\kappa > 1$ ). Here  $\kappa$  is the dimensionless surface-mass density (or convergence) of the lens. This means that lenses will surface-mass density above a certain threshold will be designated as strong lenses. Observation of strong lensing is rare. Observation of weak lensing is not rare, but subtle as it is a weak effect and requires statistical treatment. A typical application is to measure the distortion of background galaxies lensed by a (foreground) cluster of galaxies. This effect was first detected by Tyson et al. in 1990 [12] in a pioneering work on **cluster lens-mass reconstruction**. Since there is a map between surface-mass density and lensing, one early recognised GL as a good probe on the distribution of dark matter. Already in 1989, microlensing‡ was used to explore the nature of dark matter [13]. Since then the effect has been used to detect black holes, exoplanets and much more. As late as in 2018, the study of microlensing effects in images magnified through galaxy-cluster lensing effects, led to the discovery of Icarus, the most distant star observed so far [14,15]. The field of strong lensing was recently reviewed by Anna Barnacka, who concludes that it provides unique physical information about the central structure of active galaxies [16]. Since GL-effects such as magnification, arcs and rings [17, 18] allow for deep-galaxy observation, one may conclude that improved accuracy of GL-measurements means an improved probe on the Universe of the distant past.

**Gravitational lensing (GL)** is a phenomena rich with information. The theoretical frameworks that we use to understand this information, is somewhat artificially divided into two regimes: weak and strong gravitational lensing.

### 2.1. Number of images in strong lenses

For strong lenses, one will typically observe more than one image. According to [19], H.Witt showed by a direct calculation that for a lens consisting of one lensing body, the

† The Hubble constant measures the current rate of expansion of the Universe

‡ Microlensing typically refers to the lensing of galactic or extra-galactic sources by lenses inside our galaxy.

number of lensed images is  $< n^2 + 1$ . The maximum possible number of lensed images is however shown to be  $\geq 3n + 1$ . Also, one has the following theorem [20]:

**Theorem 2.1.** *The number of lensed images by an  $n$ -mass,  $n > 1$ , planar lens cannot exceed  $5n - 5$  and this bound is sharp. Moreover, the number of images is an even number when  $n$  is odd, and an odd number when  $n$  is even.*

Consider also [21].

### 3. GL as a probe on dark matter

The first use of GL was, as mentioned, to pin down the correct theory of gravity. The dispute concerning whether or not GR really is the correct theory of gravity has however not remained silent. Today we know that if the standard  $\Lambda$ CDM-cosmology † is correct, then galaxies contain a whole lot more matter than that which is luminous. The Planck survey suggests a ratio of luminous to non-luminous of about 1 to 5 [22, 23]. Furthermore, estimates suggest that in a galaxy cluster, about 80% of the mass is due to non-luminous sources [24, p. 300]. According to the same source (p. 327) observations of colliding galaxy clusters cannot properly be accounted for without the inclusion of collision-less dark matter—even if GR is modified on large scales. Within the paradigm of  $\Lambda$ CDM-cosmology, one has therefore set out to search for the nature of this so-called dark matter.

To this end it is important to map the gravitational potential of the Universe. A large share of today’s cosmology is devoted to finding out, and this is where lensing comes in as an excellent tool [25, 26].

### 4. Lens-mass reconstruction

One of the most prospering outputs of studying GL, has been the ability to do lens-mass reconstruction. Understanding how light-rays are being deflected by a lens, is equivalent to understanding the gravitational potential of the lens; its magnitude and distribution. This, in turn, translates into a map of dark matter in the lens. Early in the 1990s, **Kaiser and Squires** did pioneering work to this end; finding expressions for the mass in terms of the shear field measured in galaxy clusters [27, 28]. A key improvement from previous attempts with weak lensing (e.g. [29, 30]) was that this approach was model-independent. Later Schneider and Seitz [31–33] developed the method further, using only local data to reconstruct the lens mass from the shear field. These early works have paved the way for an array of papers describing the implementation of Kaiser and Squires’ work to obtain real data.

† Standard cosmology is named after its two main constituents: The cosmological constant ( $\Lambda$ ) and cold dark matter (CDM).

**Goal:** We want to be able to take the image of a distorted galaxy and find out what it actually looked like. Not because we are so interested in the un-distorted galaxy, but because we through this process we may harvest information about the lens.

## 5. The equation describing the problem

In order to meet our goal, we shall in our project use the so-called Roulette-formalism for lens-mass reconstruction. This approach was developed by Chris Clarkson at QMUL in 2015-2016 in a series of papers [34–36] where he showed how to take higher-order effects into account when dealing with weak lensing. By such, he showed that his theory is capable of treating both weak and strong lensing within the same formalism ‡. The starting point is the equation

$$\ddot{\xi} - \mathcal{R}\xi = \mathcal{F}, \quad (1)$$

We shall skip the mathematical details of this equation, and suffice it to say that  $\xi$  is here a vector that describes the deviation between two light rays. In other words, this is a differential equation showing the dynamical development of this deviation vector. Thus one may for instance consider two light rays originally closely separated, and see how this separation changes in various ways due to a lens. The quantity  $\mathcal{R}$  is called the optical tidal matrix, and entails information about the geometry of the space-time in which the light is propagating. Finally,  $\mathcal{F}$  is a matrix containing leading order screen-space † derivatives of the optical tidal matrix  $\mathcal{R}$ . The more terms we include in  $\mathcal{F}$ , the nastier it becomes and the more accurate the results.

### 5.1. Situation

Let us look at a galaxy in the night sky. This galaxy we shall refer to as *the source* (S). Now, imagine that somewhere between the observer and the source there is a very heavy object – perhaps a galaxy cluster (GC). This galaxy cluster will distort the path of the light rays emitted by S. The effect of this is that the images of the source S will look distorted. Now: This distortion is due to the lens. A lens is fully characterised by its so-called *potential*‡, for which we use the Greek letter  $\psi$  (psi). This means that different lenses will cause different distortions. We may thus say that the quantities  $\mathcal{F}$  and  $\mathcal{R}$  in Equation (1) are functions of the lense, or more particularely of  $\psi$ :

$$\ddot{\xi} - \mathcal{R}(\psi)\xi = \mathcal{F}(\psi),$$

‡ consult ?? for further motivation

† Screen-space is the observers sphere; id est the two-dimensional celestial sphere that he observes as the night sky above him.

‡ Much like we talk about potential in electrical engineering, or potential energy in mechanics, we define also here a quantity that we refer to as the potential  $\psi$  of the lens.

**Idea:** The geodesic deviation equation describes how two light rays of separation  $\xi$  in the source-plane translates into a separation  $\zeta$  in the image.

## 6. The general solution to the equation

In his works Clarkson finds the solution to Equation (1) by a perturbative approach. In fact, he calculates the  $m$ -th order distortion of an image by a lens  $\psi$ . The more orders one includes in this expansion, the better does the solution correspond to reality. Consider soundwaves as an analogy. Playing the same tune on a piano and a violin, does not result in equivalent sounds. The difference is encoded in all the overtones. These are the modes of vibration with wavelengths shorter than the tone played. The more overtones our ears or recorders can detect, the better may we represent the true sound. In much the same way, the more terms we include in the series expansion, the better does our mapping represent the real situation.

*The general solution* Take  $\xi = (\xi_1, \xi_2)$  to be the deviation vector giving the separation of two light rays at the source. Generally then, in terms of a position vector  $\zeta = (\zeta_1, \zeta_2)$  in the image plane, the solution may be expressed as (yes, this will be ugly)

$$\xi_A = \sum_{m=1}^{\infty} \frac{r^m}{m!} \hat{\xi}_A^{(m)}. \quad (2)$$

where

$$\begin{aligned} \hat{\xi}_A^{(m)} = \mathcal{M}_{AB_1 \dots B_m} \hat{\zeta}^{B_1} \dots \hat{\zeta}^{B_m} &= \sum_{s=0}^{m+1} \frac{[1 - (-1)^{m+s}]}{4} \times \\ & \left( [[C^+ \alpha_s^m + \bar{\beta}_s^m] \mathbf{R}_- + [C^+ \beta_s^m - \bar{\alpha}_s^m] \mathbf{R}_/ ] \mathbf{p}_{s-1} \right. \\ & \left. + [[C^- \alpha_s^m - \bar{\beta}_s^m] \mathbf{I} + [C^- \beta_s^m + \bar{\alpha}_s^m] \boldsymbol{\varepsilon}] \mathbf{p}_{s+1} \right). \end{aligned} \quad (3)$$

Moreover, we have defined  $\hat{\zeta} = \zeta/r$  to be the radial unit vector in the lens plane, and we have defined the constants

$$C^\pm = 1 \pm \frac{s}{m+1}. \quad (4)$$

Furthermore,  $\mathbf{I}$  is the  $2 \times 2$  identity matrix, and the rest are the Pauli spin matrices:

$$\boldsymbol{\varepsilon}_- = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}, \quad \mathbf{R}_- = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}, \quad \mathbf{R}_/ = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}. \quad (5)$$

Also

$$\mathbf{p}_s = (\cos(s\theta), \sin(s\theta)). \quad (6)$$

Finally, the **coefficients**  $\alpha_s^m$  and  $\beta_s^m$  (and  $\hat{\alpha}_s^m$  and  $\hat{\beta}_s^m$ ) are the so-called even (and odd) **Roulette-amplitudes**. These are the coefficients of the 2D-Fourier series that

for a given order  $m$  give the amplitudes of the spin- $s$  contributions to the distortion. By such, one has for any order  $m$ , decomposed the  $s$  distortions of an image into independent contributions. This formalism therefore entails not only the distortions included in typical approaches to lensing (these distortions are called convergence and shear), but also higher order effects † to arbitrary order in leading screen-space derivatives. Consequently, one may view this approach as a *weak-lensing approach to strong lensing*.

*But wait...* where is the information about what type of lens we are dealing with? After all, the GDE (5.1) had terms  $\mathcal{R}$  and  $\mathcal{F}$  depending on the lens  $\psi$ , so the solution should also depend on  $\psi$ . Well, this information is now found in the amplitudes  $\alpha_s^m(\psi)$  and  $\beta_s^m(\psi)$  (and  $\hat{\alpha}_s^m(\psi)$  and  $\hat{\beta}_s^m(\psi)$ ), which may be shown to become functions of the lens potential  $\psi$ .

**Tool:** The  $m$ -th order solution (3) to the GDE (1) provides a map showing how a source is distorted into an image by a given lens  $\psi$ . This map works both in the weak-lensing and strong-lensing regimes.

## 7. A Thin lens in the weak-field and flat-sky approximations

In the rest of this note, we concern ourselves with a general lens fulfilling the following criteria.

- The lens is optically thin.
- The lens is so far away that we may consider the observer sphere as a flat sky.
- The lens is weak in the sense that it is described by the *weak-field* approximation.

In this case, the roulette amplitudes may be written down. In a previous work ([37], Eq. 75 and 76 therein) I showed that in the thin-lens and weak-field approximation the amplitudes may be expressed as follows.

$$\begin{aligned}
 \alpha_s^m &= \Gamma_s^m \square^{a^-} \sum_{k=0}^{2k \leq s} (-1)^k \binom{s}{2k} \partial_X^{s-2k} \partial_Y^{2k} \psi(X, Y), \\
 \beta_s^m &= \Gamma_s^m \square^{a^-} \sum_{k=0}^{2k+1 \leq s} (-1)^k \binom{s}{2k+1} \partial_X^{s-2k-1} \partial_Y^{2k+1} \psi(X, Y), \\
 \bar{\alpha}_s^m &= 0, \\
 \bar{\beta}_s^m &= 0.
 \end{aligned} \tag{7}$$

Inserting Eqs. (7) into (3), and using (5) and (6), we find a matrix relation between the source and lens plane:

† The two first of which are typically named flexion and second flexion.



**In the thin-lens, weak-field and flat-sky approximations**, the map from coordinates  $(\xi_1, \xi_2)^T$  in the source plane to (polar) coordinates  $[r, \theta]^T$  in the lens plane is given by

$$\begin{pmatrix} \xi_1 \\ \xi_2 \end{pmatrix} = \sum_{m=1}^{\infty} \frac{r^m}{m!} \sum_{s=0}^{m+1} \frac{[1 - (-1)^{m+s}]}{4} \times \left( \alpha_s^m \begin{pmatrix} \cos(s-1)\theta & \cos(s+1)\theta \\ -\sin(s-1)\theta & \sin(s+1)\theta \end{pmatrix} + \beta_s^m \begin{pmatrix} \sin(s-1)\theta & \sin(s+1)\theta \\ \cos(s-1)\theta & -\cos(s+1)\theta \end{pmatrix} \right) \begin{pmatrix} C^+ \\ C^- \end{pmatrix} \quad (8)$$

The information about the lens comes in through the amplitudes  $\alpha_s^m(\psi)$  and  $\beta_s^m(\psi)$ , as given by Eqs. (7).

Let it be clear that Equations (8) is a general map  $\mathcal{L} : U \mapsto D$ , where the potential  $\psi$  has not been specified. In principle, this is all we need. In the following, however, we shall introduce recursion relations that may help in actual calculations, especially when the potential  $\psi$  proves complicated.

*Recursion relations* The amplitudes  $\alpha_s^m(\psi)$  and  $\beta_s^m(\psi)$  may be shown to be related through recursion relations, as proven in [37]. These relations are

$$\alpha_{s+1}^{m+1} = (C_+^+)^{m+1} (\partial_X \alpha_s^m - \partial_Y \beta_s^m) \quad (9)$$

$$\beta_{s+1}^{m+1} = (C_+^+)^{m+1} (\partial_X \beta_s^m + \partial_Y \alpha_s^m) \quad (10)$$

and

$$\alpha_{s-1}^{m+1} = (C_-^+)^{m+1} (\partial_X \alpha_s^m + \partial_Y \beta_s^m) \quad (11)$$

$$\beta_{s-1}^{m+1} = (C_-^+)^{m+1} (\partial_X \beta_s^m - \partial_Y \alpha_s^m). \quad (12)$$

Here  $(C_+^+)$  and  $(C_-^+)$  are algebraic coefficients given by

$$(C_+^+)^m = 2^{\delta_{0(s-1)}} \frac{m+1}{m+1+s} \chi \quad \text{and} \quad (C_-^+)^m = 2^{-\delta_{0s}} \frac{m+1}{m+1-s} \chi. \quad (13)$$

The signs in the name of the coefficients  $(C_+^+)$  and  $(C_-^+)$  are there to indicate the direction of change in the value  $m$  (superscript) and  $s$  (subscript) †. These recursion relations permit for simply calculating for instance  $\alpha_1^0$  and  $\beta_1^0$  from the potential. From Eq. (7) one may show that these are

$$\alpha_1^0 = -\chi \partial_X \psi \quad \text{and} \quad \beta_1^0 = -\chi \partial_Y \psi. \quad (14)$$

From these two ‘lowest modes’ of expansion, the remaining amplitudes may be calculated from the recursion relations. This circumvents the need for invoking the intimidating formulae (7) for the calculation of every new amplitude.

† We provide a more compact notation for these relations (and the inverses) in the paper.

**The recursion relations** provide an alternative route (circumventing eq. (7)) for calculating the amplitudes  $\alpha_s^m(\psi)$  and  $\beta_s^m(\psi)$ . If this is a beneficial route to go, must be judged on a case-to-case basis.

### 7.1. Notation used in the thin-lens approximation

In creating a map between the source and the image, we shall have to deal with several coordinate systems, as shown in Figure 1. These are:

- *Coordinates centered at the image.* We take the coordinates  $(x, y)$  to be centered at the image in the lens-plane. In polar coordinates we write  $(x, y) = r(\cos \theta, \sin \theta)$ .
- *Coordinates centered at the source.* We take the coordinates  $(x', y')$  to be centered at the apparent source in the source plane.
- *Coordinates centered at the lens.* We shall use capital letters  $(X, Y)$  to refer to coordinates centered at the lens, in the lens-plane.

When observing an image, however, all we can do, is to observe angular distances. The (observed) angular position of a point in the image-coordinate system  $(x, y)$  may be seen as a two-dimensional vector of angles  $\hat{\zeta} = (\cos \theta, \sin \theta)$ . Knowing that the distance  $\chi_L$  to the lens plane is much larger than the distances between image points, and going to polar coordinates, one may show that  $\zeta = (x, y)/\chi_L = r/\chi_L$  and that the unit vector is  $\hat{\zeta} = (\cos \theta, \sin \theta)$ .

### 7.2. Projected surface-mass density in the thin-lens approximation

The Poisson equation in three dimensions † describe the gravitational potential  $\Phi$  resulting from a mass-density  $\rho$  distributed in space. We are interested in what happens in the two-dimensional screen-space. It is therefore customary to integrate along the direction orthogonal to screen-space. One may show (refer to Appendix B for some detail) that in the thin-lens approximation, the projected ‡ Poisson equation reads

$$\kappa = \frac{\chi_L^2}{2} \nabla^2 \psi. \quad (15)$$

Here  $\psi(X, Y)$  is an effective two-dimensional potential defined, and  $\kappa(X, Y)$  is the dimensionless, projected surface-mass density

$$\kappa = \frac{\Sigma}{\Sigma_{\text{CR}}}. \quad (16)$$

In the latter equation  $\Sigma$  is the projected surface mass density, and  $\Sigma_{\text{CR}}$  is defined as

$$\Sigma_{\text{CR}} = \frac{4\chi_S}{\chi_L(\chi_S - \chi_L)\lambda}. \quad (17)$$

† Refer to Appendix B for details

‡ What we mean by projected in this context is something like “integrated along the direction orthogonal to screen-space.”

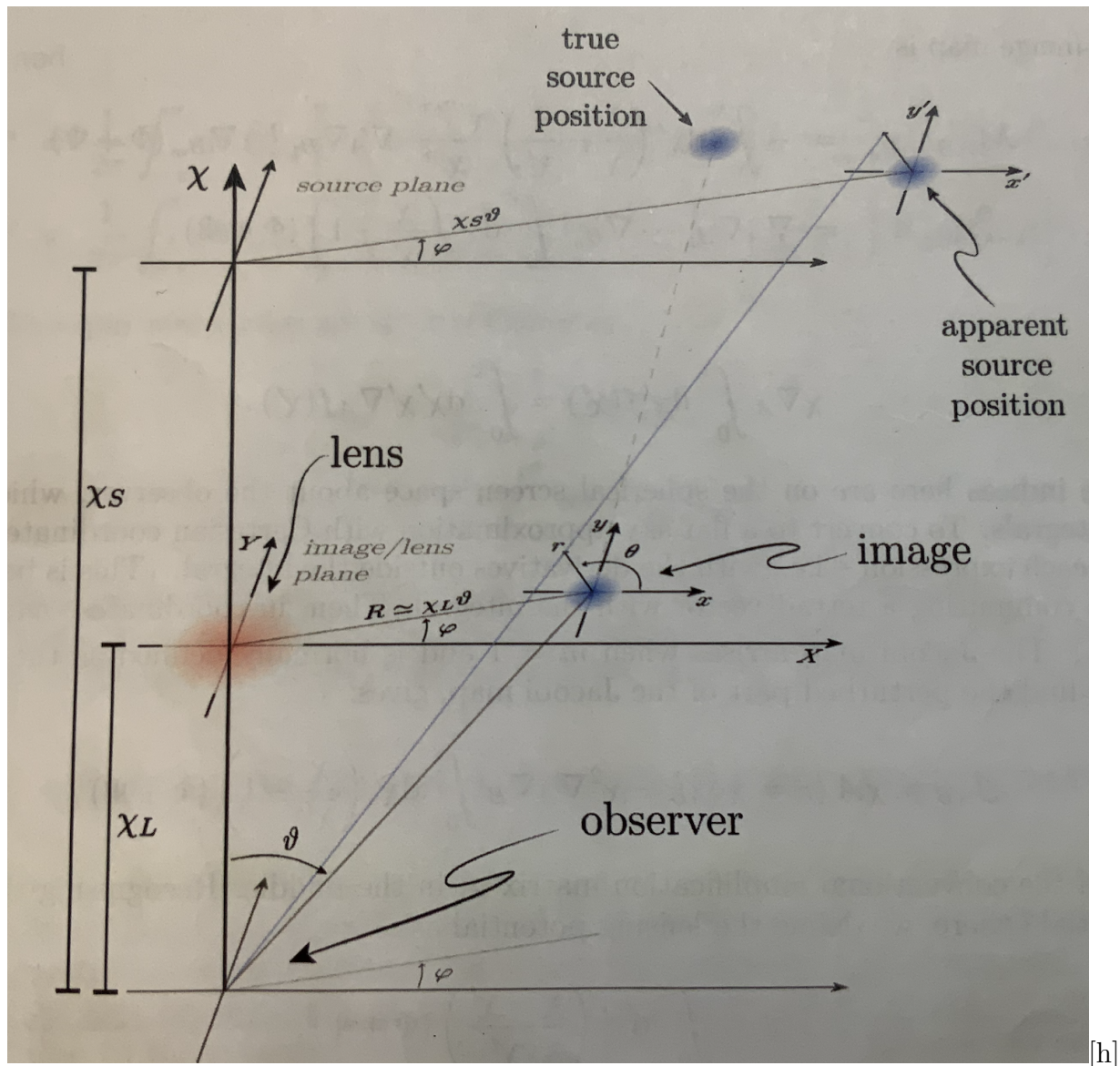


Figure 1: The figure shows the notation used for the Roulette formalism in the thin-lens approximation.

Note that  $\kappa < 1$  corresponds to weak lensing, whereas  $\kappa > 1$  corresponds to strong lensing.

### 8. The study of a particular lens ( $\mathcal{L}_\psi : U \mapsto D$ )

In order to actually use the general map (3), or the slightly more restricted map (8), we must specify a potential. Id est, we must create a map  $\mathcal{L}_\psi : U \mapsto D$ . The procedure will be as follows:

In order to use Eq. (3) (or Eq. (8)) to create a distortion of a source  $S$ , we must generally

- (i) Specify the lens by giving the lens potential  $\psi(X, Y)$ .
- (ii) Calculate the amplitudes  $\alpha_s^m(\psi)$ ,  $\beta_s^m(\psi)$  etc...
- (iii) Insert these amplitudes into Eq. (3) (or Eq. (8)).

By following the above procedure we have a map from the source to the image for a particular lens  $\psi$ . The information about the lens, comes in via the amplitudes. To actually construct a map  $\mathcal{L}_\psi$ , we must thus specify  $\psi$ . In the following subsections, we look at the mathematical description of a few particular lensing potentials  $\psi$ , and give tasks for implementation of these.

## 9. Implementing a point-mass lens

The simplest lens model one may study, is that of a point-mass. Id est; a lens where all the mass is concentrated at one point. Denote this point in the screen-space for  $R$ . Then  $\Sigma = \Sigma_0 \delta(R)$  in this case, for some constant  $\Sigma_0$ . One may also compute that in this case

$$\Sigma_{\text{CR}} = \frac{M}{\pi R_E^2}, \quad (18)$$

where  $R_E$  is the Einstein radius. More importantly, the amplitudes take a particularly simple form. Placing a source on the  $X$ -axis (see figure 1) in the lens plane, the non-vanishing amplitudes may be shown to become

$$\alpha_{m+1}^m = (-1)^s m! \left( \frac{R_E}{R} \right)^2 \left( \frac{\chi_L}{R} \right)^{m-1}. \quad (19)$$

Inserting Equation (19) into Equation (8) (id est: finishing off step (iii) in the procedure described previously) one finds that the relationship between the coordinates in the source plane and lens plane may actually be written down in closed form [36]! They become

$$\begin{pmatrix} \frac{\chi_l}{\chi_s} \\ \frac{y'}{x'} \end{pmatrix} = r \begin{pmatrix} \cos \theta \\ \sin \theta \end{pmatrix} + \frac{R_E^2 r}{r^2 + R^2 + 2rR \cos \theta} \begin{pmatrix} \frac{r}{R} + \cos \theta \\ -\sin \theta \end{pmatrix} \quad (20)$$

where  $R_E$  is a constant (of the lens) known as the Einstein radius. In coordinates

✓ **Task 1:** Use equation (20) to create a visualisation of how a circular source in the source plane is distorted into an image for a point-mass lens.

To do this project it might be useful to simply create a circularly symmetric source of for instance a Gaussian intensity profile

$$I(x', y') = \frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{1}{2\sigma^2}((x'-x'_0)^2 + (y'-y'_0)^2)} \quad (21)$$

where  $P = (x_0, y_0)$  is the position of the source. Consider for instance

$$\sigma = \frac{1}{6} \frac{\chi_L}{\chi_S} R_E \quad \text{and} \quad R = 1.3 R_E. \quad (22)$$

This will then be the source that should be distorted by the mapping given by Equation (20)

## 10. Implementing the isothermal sphere

If one considers a galaxy as a spherically-symmetric self-gravitating concentration of stars behaving like an ideal gas in hydrostatic equilibrium, one obtains a differential equation for its radial mass distribution (consider [38] and references therein, such as Binney and Tremain 1987). One of the solutions to this equation is known as defining the so-called *singular isothermal sphere*. It is very simple, since it has spherical symmetry. This symmetry is therefore also present when we look at the 2dimensional projected mass-distribution. One may show the following.

**SIS** profile: The dimensionless projected surface-mass density  $\kappa$  is

$$\kappa(R) = \frac{\Gamma}{R} \quad \text{where} \quad \Gamma = \frac{4\pi v^2}{\lambda \Sigma_{\text{CR}}}. \quad (23)$$

Here  $R = \sqrt{X^2 + Y^2}$  and  $v^2$  is the velocity dispersion along the line-of-sight.

Inserting this into (15) and solving, one finds the general solution

$$\psi(R) = \frac{2\Gamma}{\chi_L^2} R + k_1 \ln R + k_2. \quad (24)$$

Requiring  $\psi(0) = 0$  we are forced to set  $k_1 = k_2 = 0$ . The potential of the SIS profile is therefore

$$\psi(R) = \frac{2\Gamma}{\chi_L^2} R. \quad (25)$$

The problem with this profile, is the diverging of the surface-mass density at the center. Still in the spirit of SIS, but to avoid this singularity, one constructs a similar profile, known as the *nonsingular isothermal sphere*, by throwing in a finite core with radius  $R_c$ . One defines

**NIS** profile: The dimensionless projected surface-mass density  $\kappa$  is

$$\kappa(R) = \frac{\Gamma}{\sqrt{R^2 + R_c^2}}. \quad (26)$$

Using this in Eq. (15) one finds the general solution

$$\psi(R) = \frac{2\Gamma}{\chi_L^2} \left( \sqrt{R^2 + R_c^2} - R_c \tanh^{-1} \left( \frac{\sqrt{R^2 + R_c^2}}{R_c} \right) \right) + k_1 \ln R + k_2, \quad (27)$$

Since both the tanh-function and the ln-function blow up for  $R = 0$  we find that it is possible to make these terms balance each other out, to produce  $\psi(0) = k_2 \S$ . Requiring  $\psi(0) = 0$  we set  $k_2 = 0$ .

\S Scetchy mathematics... I haven't looked at the details. Also: Why do we have to require  $\psi$  finite for  $R = 0$ ? It is the surface-mass density that is the physical field here...?

*10.0.1. Elliptical profiles* A more complicated model, would be one obtained by now relieving the SIS model of its circular symmetry. Id est, we let instead

Consider also [\[39\]](#).

**Task  $\infty$ :** Use equation (8) to create a visualisation of how the source (in the source-plane coordinates  $(\xi_1, \xi_2)$ ) is distorted into an image (in the image-plane coordinates  $[r, \theta]$ ). This should be done for

- The SIS-profile (25) (circular symmetry).
- The An elliptic lens profile



## Appendix A. Roulette coefficients

For a general *thin lens* in the *weak-field* and *flat-sky* approximation, the (non-vanishing) roulette coefficients may be shown to be given by

$$\alpha_s^m = -2^{-\delta_{0s}} \chi^{m+1} \sum_{k=0}^m \binom{m}{k} (\mathcal{C}_s^{m(k)} \partial_X + \mathcal{C}_s^{m(k+1)} \partial_Y) \partial_X^{m-k} \partial_Y^k \psi, \quad (\text{A.1})$$

$$\beta_s^m = -\chi^{m+1} \sum_{k=0}^m \binom{m}{k} (\mathcal{S}_s^{m(k)} \partial_X + \mathcal{S}_s^{m(k+1)} \partial_Y) \partial_X^{m-k} \partial_Y^k \psi, \quad (\text{A.2})$$

where  $\chi$  is the distance from the observer to the lens.  $X, Y$  are coordinates in the lens-plane and  $\psi = \psi(X, Y)$  is the lensing potential, which must be specified †. Also the spin  $s$  is restricted such that  $0 \leq s \leq m + 1$  and the roulette amplitudes  $\alpha_s^m, \beta_s^m$  may be non-zero only if  $m + s$  is odd. Finally;

$$\mathcal{C}_s^{m(k)} = \frac{1}{\pi} \int_{-\pi}^{\pi} d\theta \sin^k \theta \cos^{m-k+1} \theta \cos s\theta, \quad (\text{A.4})$$

$$\mathcal{S}_s^{m(k)} = \frac{1}{\pi} \int_{-\pi}^{\pi} d\theta \sin^k \theta \cos^{m-k+1} \theta \sin s\theta. \quad (\text{A.5})$$

The above expressions for the roulette-amplitudes  $\alpha_s^m, \beta_s^m$  may also be written down in a much simpler form by going to the complex plane, as shown in [37]. Defining  $\gamma_s^m = \alpha_s^m + i\beta_s^m$  we find to the neat little inch

$$\gamma_s^m = \Gamma_s^m \square^{a^-} \partial_c^s \psi. \quad (\text{A.6})$$

Here  $a^- = (m + 1 - s)/2$  and  $\Gamma_s^m$  are numerical coefficients given by

$$\Gamma_s^m = \begin{cases} -(2^{-\delta_{0s}}) \frac{\chi^{m+1}}{2^m} \binom{m+1}{a^-} & m + s \text{ odd,} \\ 0 & \text{else.} \end{cases} \quad (\text{A.7})$$

Sums and integrals are thus dismissed from the expressions. This latter part makes it much easier to actually compute the roulette coefficients.

† Actually it must be calculated from the defining relation

$$\psi = \int_0^\chi d\chi' \left( \frac{\chi - \chi'}{\chi \chi'} \right) (\Phi + \Psi), \quad (\text{A.3})$$

where  $\Phi$  and  $\Psi$  are functions defining the potential. These functions are obtained from something known as the metric of space-time. Recall that in general relativity, a gravitational potential is a curved space-time.

## Appendix B. The Poisson equation for gravity

Gauss' law for gravity on differential form, reads

$$\tilde{\nabla} \mathbf{g} = -4\pi G \rho, \quad (\text{B.1})$$

where  $G$  is the gravitational constant and  $\rho$  is the matter density in the Euclidian 3-space. Also,  $\tilde{\nabla}$  is the 3-dimensional Laplacian<sup>†</sup> Knowing that the gravitational field  $\mathbf{g}$  is conservative and irrotational, it may be expressed as the gradient of a scalar  $\psi$ . We choose a negative sign and find

$$\mathbf{g} = -\tilde{\nabla} \Phi, \quad (\text{B.2})$$

and thus find what is know as Poisson's equation:

$$\tilde{\nabla}^2 \Phi = \frac{1}{2} \lambda \rho \quad \text{where} \quad \lambda = 8\pi G. \quad (\text{B.3})$$

In perturbation theory, what is known as *the Poisson gauge* starts from a perturbed metric where the line-element is

$$ds^2 = - \left( 1 + 2 \frac{\Phi}{c^2} \right) c^2 dt^2 + \left( 1 - 2 \frac{\Psi}{c^2} \right) \gamma_{ij} dx^i dx^j, \quad (\text{B.4})$$

where  $\Phi$  and  $\Psi$  are 3-scalar perturbations,  $\gamma_{ij}$  is the 3-dimensional metric tensor and  $c$  is the speed of light. Furthermore, in [35], the so-called *lensing potential* in the weak-field approximation is **defined** (Eq. 102 in [35]) as

$$\psi = \int_0^x d\chi' \left( \frac{\chi - \chi'}{\chi \chi'} \right) (\Phi + \Psi). \quad (\text{B.5})$$

Ignoring peculiar velocity terms (setting  $\Psi = 0$  (?)) and using the (B.3) equation, one may show that *in the thin lens approximation* we find a 2-dimensional Poisson equation given by

$$\kappa(X, Y) = \frac{\chi_L^2}{2} \nabla^2 \psi(X, Y). \quad (\text{B.6})$$

The dimensionless surface-mass density  $\kappa$  is defined as  $\kappa = \Sigma / \Sigma_{\text{CR}}$ , where

$$\Sigma(X, Y) = \int_0^{\chi_S} d\chi' \rho(\chi', X, Y) \quad (\text{B.7})$$

is the surface-mass density obtained by integrating the mass density along the optical axis and

$$\Sigma_{\text{CR}} = \frac{4\chi_S}{\chi_L(\chi_S - \chi_L)\lambda}. \quad (\text{B.8})$$

<sup>†</sup> A  $\tilde{\nabla}$  is used to distinguish it from the 2-dimensional Laplacian, which we shall be using in the main text.

## Appendix C. References

- [1] I. Newton, *Opticks: or, A treatise of the reflections, refractions, inflexions and colours of light. Also two treatises of the species and magnitude of curvilinear figures.* London : Printed for Sam Smith, and Benj. Walford, printers to the Royal Society, at the prince's Arms in St. paul's Church-yard, 1804.
- [2] C. Montgomery, W. Orchiston and I. Whittingham, *Michell, Laplace and the origin of the black hole concept, Journal of Astronomical History and Heritage* **12** (July, 2009) 90–96.
- [3] J. Mitchell, *On the means of discovering the distance, magnitude, &c. of the fixed stars, in consequence of the diminution of the velocity of their light, in case such a diminution should be found to take place in any of them, and such other data should be procured from observations, as would be farther necessary for that purpose. by the rev. john michell, b. d. f. r. s. in a letter to henry cavendish, esq. f. r. s. and a. s., Philosophical Transactions of the Royal Society of London (1776-1886)* **74** (1784) .
- [4] S. Laplace, *Exposition du système du monde.* .
- [5] W. Dyson Frank, S. Eddington Arthur and C. Davidson, *Ix. a determination of the deflection of light by the sun's gravitational field, from observations made at the total eclipse of may 29, 1919, Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character* **220** (1920) 291–333.
- [6] A. Einstein, *Lens-like action of a star by the deviation of light in the gravitational field, Science* **84** (1936) 2.
- [7] F. Zwicky, *Nebulae as gravitational lenses, Physical Review* **51** (1937) 290–290.
- [8] F. Zwicky, *On the probability of detecting nebulae which act as gravitational lenses, Physical Review* **51** (1937) 679–679.
- [9] H. Bondi and S. Refsdal, *The gravitational lens effect, Monthly Notices of the Royal Astronomical Society* **128** (1964) 295–306.
- [10] S. Refsdal, *On the possibility of determining hubble's parameter and the masses of galaxies from the gravitational lens effect, Monthly Notices of the Royal Astronomical Society* **128** (1964) 307–310.
- [11] P. Schneider, C. Kochanek and J. Wambsganss, *Gravitational Lensing: Strong, Weak and Micro*, vol. 33 of *Saas-Fee Advanced Course.* Springer-Verlag Berlin Heidelberg, 1 ed., 2006, [10.1007/978-3-540-30310-7](https://doi.org/10.1007/978-3-540-30310-7).
- [12] J. A. Tyson, F. Valdes and R. A. Wenk, *Detection of systematic gravitational lens galaxy image alignments - mapping dark matter in galaxy clusters, Astrophysical Journal* **349** (1990) L1–&.
- [13] M. J. Irwin, R. L. Webster, P. C. Hewett, R. T. Corrigan and R. I. Jedrzejewski, *Photometric variations in the q2237 + 0305 system - first detection of a microlensing event, Astronomical Journal* (1989) .
- [14] J. M. Diego, N. Kaiser, T. Broadhurst, P. L. Kelly, S. Rodney, T. Morishita et al., *Dark matter under the microscope: Constraining compact dark matter with caustic crossing events, The Astrophysical Journal* **857** (2018) 25.
- [15] P. L. Kelly, J. M. Diego, S. Rodney, N. Kaiser, T. Broadhurst, A. Zitrin et al., *Extreme magnification of an individual star at redshift 1.5 by a galaxy-cluster lens, Nature Astronomy* **2** (2018) 334–342.
- [16] A. Barnacka, *Gravitational lenses as high-resolution telescopes, Physics Reports-Review Section of Physics Letters* **778** (2018) 1–46.
- [17] J. P. Kneib, P. P. van der Werf, K. K. Knudsen, I. Smail, A. Blain, D. Frayer et al., *A multiply imaged, submillimetre-selected ultraluminous infrared galaxy in a galaxy group at z similar to 2.5, Monthly Notices of the Royal Astronomical Society* **349** (2004) 1211–1217.
- [18] J. Richard, T. Jones, R. Ellis, D. P. Stark, R. Livermore and M. Swinbank, *The emission line properties of gravitationally lensed  $1.5 < z < 5$  galaxies, Monthly Notices of the Royal Astronomical Society* **413** (2011) 643–658.

- [19] C. D. Fassnacht, C. R. Keeton and D. Khavinson, *Gravitational lensing by elliptical galaxies, and the schwarzschild function*, 2007.
- [20] S. H. Rhie, *n-point gravitational lenses with  $5(n-1)$  images*, 2003.
- [21] N. Straumann, *Complex formulation of lensing theory and applications*, 1997.
- [22] P. A. R. Ade, N. Aghanim, M. Arnaud, M. Ashdown, J. Aumont, C. Baccigalupi et al., *Planck 2015 results xiii. cosmological parameters*, *Astronomy & Astrophysics* **594** (2016) .
- [23] N. Aghanim, Y. Akrami, M. Ashdown, J. Aumont, C. Baccigalupi, M. Ballardini et al., *Planck 2018 results*, *Astronomy & Astrophysics* **641** (Sep, 2020) A6.
- [24] P. Schneider, *Extragalactic Astronomy and Cosmology*. Springer-Verlag Berlin Heidelberg, 2 ed., 2015, [10.1007/978-3-642-54083-7](https://doi.org/10.1007/978-3-642-54083-7).
- [25] R. B. Metcalf and P. Madau, *Compound gravitational lensing as a probe of dark matter substructure within galaxy halos*, *The Astrophysical Journal* **563** (Dec, 2001) 9–20.
- [26] S. Birrer and A. Amara, *lenstronomy: Multi-purpose gravitational lens modelling software package*, *Physics of the Dark Universe* **22** (2018) 189–201.
- [27] N. Kaiser, *Weak gravitational lensing of distant galaxies*, *Astrophysical Journal* **388** (1992) 272–286.
- [28] N. Kaiser and G. Squires, *Mapping the dark matter with weak gravitational lensing*, *Astrophysical Journal* **404** (1993) 441–450.
- [29] C. S. Kochanek, *Inverting cluster gravitational lenses*, *Monthly Notices of the Royal Astronomical Society* **247** (1990) 135–151.
- [30] J. Miraldaescude, *The correlation-function of galaxy ellipticities produced by gravitational lensing*, *Astrophysical Journal* **380** (1991) 1–8.
- [31] P. Schneider, *Cluster lens reconstruction using only observed local data*, *Astronomy & Astrophysics* **302** (1995) 639–648.
- [32] P. Schneider and C. Seitz, *Steps towards nonlinear cluster inversion through gravitational distortions .1. basic considerations and circular clusters*, *Astronomy & Astrophysics* **294** (1995) 411–431.
- [33] S. Seitz and P. Schneider, *Cluster lens reconstruction using only observed local data: An improved finite-field inversion technique*, *Astronomy & Astrophysics* **305** (1996) 383–401.
- [34] C. Clarkson, *The general theory of secondary weak gravitational lensing*, *Journal of Cosmology and Astroparticle Physics* **2015** (2015) 033–033.
- [35] C. Clarkson, *Roulettes: a weak lensing formalism for strong lensing: I. overview*, *Classical and Quantum Gravity* **33** (2016) .
- [36] C. Clarkson, *Roulettes: a weak lensing formalism for strong lensing: Ii. derivation and analysis*, *Classical and Quantum Gravity* **33** (2016) .
- [37] B. D. Normann and C. Clarkson, *Recursion relations for gravitational lensing*, *General Relativity and Gravitation* **52** (Mar, 2020) .
- [38] R. Kormann, P. Schneider and M. Bartelmann, *Isothermal elliptic gravitational lens models*, *Astronomy & Astrophysics* **284** (1994) 285–299.
- [39] J. Hjorth and J.-P. Kneib, “Elliptical galaxies as gravitational lenses.”
- [1] I. Newton, *Opticks: or, A treatise of the reflections, refractions, inflexions and colours of light. Also two treatises of the species and magnitude of curvilinear figures*. London : Printed for Sam Smith, and Benj. Walford, printers to the Royal Society, at the prince’s Arms in St. paul’s Church-yard, 1804.
- [2] C. Montgomery, W. Orchiston and I. Whittingham, *Mitchell, Laplace and the origin of the black hole concept*, *Journal of Astronomical History and Heritage* **12** (July, 2009) 90–96.
- [3] J. Mitchell, *On the means of discovering the distance, magnitude, &c. of the fixed stars, in consequence of the diminution of the velocity of their light, in case such a diminution should be found to take place in any of them, and such other data should be procured from observations, as would be farther necessary for that purpose. by the rev. john michell, b. d. f. r. s. in a letter to henry cavendish, esq. f. r. s. and a. s.*, *Philosophical Transactions of the Royal Society of*

- London (1776-1886)* **74** (1784) .
- [4] S. Laplace, *Exposition du système du monde*, .
- [5] W. Dyson Frank, S. Eddington Arthur and C. Davidson, *Ix. a determination of the deflection of light by the sun's gravitational field, from observations made at the total eclipse of may 29, 1919*, *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character* **220** (1920) 291–333.
- [6] A. Einstein, *Lens-like action of a star by the deviation of light in the gravitational field*, *Science* **84** (1936) 2.
- [7] F. Zwicky, *Nebulae as gravitational lenses*, *Physical Review* **51** (1937) 290–290.
- [8] F. Zwicky, *On the probability of detecting nebulae which act as gravitational lenses*, *Physical Review* **51** (1937) 679–679.
- [9] H. Bondi and S. Refsdal, *The gravitational lens effect*, *Monthly Notices of the Royal Astronomical Society* **128** (1964) 295–306.
- [10] S. Refsdal, *On the possibility of determining hubble's parameter and the masses of galaxies from the gravitational lens effect*, *Monthly Notices of the Royal Astronomical Society* **128** (1964) 307–310.
- [11] P. Schneider, C. Kochanek and J. Wambsganss, *Gravitational Lensing: Strong, Weak and Micro*, vol. 33 of *Saas-Fee Advanced Course*. Springer-Verlag Berlin Heidelberg, 1 ed., 2006, [10.1007/978-3-540-30310-7](https://doi.org/10.1007/978-3-540-30310-7).
- [12] J. A. Tyson, F. Valdes and R. A. Wenk, *Detection of systematic gravitational lens galaxy image alignments - mapping dark matter in galaxy clusters*, *Astrophysical Journal* **349** (1990) L1–&.
- [13] M. J. Irwin, R. L. Webster, P. C. Hewett, R. T. Corrigan and R. I. Jedrzejewski, *Photometric variations in the q2237 + 0305 system - first detection of a microlensing event*, *Astronomical Journal* (1989) .
- [14] J. M. Diego, N. Kaiser, T. Broadhurst, P. L. Kelly, S. Rodney, T. Morishita et al., *Dark matter under the microscope: Constraining compact dark matter with caustic crossing events*, *The Astrophysical Journal* **857** (2018) 25.
- [15] P. L. Kelly, J. M. Diego, S. Rodney, N. Kaiser, T. Broadhurst, A. Zitrin et al., *Extreme magnification of an individual star at redshift 1.5 by a galaxy-cluster lens*, *Nature Astronomy* **2** (2018) 334–342.
- [16] A. Barnacka, *Gravitational lenses as high-resolution telescopes*, *Physics Reports-Review Section of Physics Letters* **778** (2018) 1–46.
- [17] J. P. Kneib, P. P. van der Werf, K. K. Knudsen, I. Smail, A. Blain, D. Frayer et al., *A multiply imaged, submillimetre-selected ultraluminous infrared galaxy in a galaxy group at z similar to 2.5*, *Monthly Notices of the Royal Astronomical Society* **349** (2004) 1211–1217.
- [18] J. Richard, T. Jones, R. Ellis, D. P. Stark, R. Livermore and M. Swinbank, *The emission line properties of gravitationally lensed  $1.5 < z < 5$  galaxies*, *Monthly Notices of the Royal Astronomical Society* **413** (2011) 643–658.
- [19] C. D. Fassnacht, C. R. Keeton and D. Khavinson, *Gravitational lensing by elliptical galaxies, and the schwarzschild function*, 2007.
- [20] S. H. Rhie, *n-point gravitational lenses with  $5(n-1)$  images*, 2003.
- [21] N. Straumann, *Complex formulation of lensing theory and applications*, 1997.
- [22] P. A. R. Ade, N. Aghanim, M. Arnaud, M. Ashdown, J. Aumont, C. Baccigalupi et al., *Planck 2015 results xiii. cosmological parameters*, *Astronomy & Astrophysics* **594** (2016) .
- [23] N. Aghanim, Y. Akrami, M. Ashdown, J. Aumont, C. Baccigalupi, M. Ballardini et al., *Planck 2018 results*, *Astronomy & Astrophysics* **641** (Sep, 2020) A6.
- [24] P. Schneider, *Extragalactic Astronomy and Cosmology*. Springer-Verlag Berlin Heidelberg, 2 ed., 2015, [10.1007/978-3-642-54083-7](https://doi.org/10.1007/978-3-642-54083-7).
- [25] R. B. Metcalf and P. Madau, *Compound gravitational lensing as a probe of dark matter substructure within galaxy halos*, *The Astrophysical Journal* **563** (Dec, 2001) 9–20.
- [26] S. Birrer and A. Amara, *lenstronomy: Multi-purpose gravitational lens modelling software*

- package, *Physics of the Dark Universe* **22** (2018) 189–201.
- [27] N. Kaiser, *Weak gravitational lensing of distant galaxies*, *Astrophysical Journal* **388** (1992) 272–286.
- [28] N. Kaiser and G. Squires, *Mapping the dark matter with weak gravitational lensing*, *Astrophysical Journal* **404** (1993) 441–450.
- [29] C. S. Kochanek, *Inverting cluster gravitational lenses*, *Monthly Notices of the Royal Astronomical Society* **247** (1990) 135–151.
- [30] J. Miraldaescude, *The correlation-function of galaxy ellipticities produced by gravitational lensing*, *Astrophysical Journal* **380** (1991) 1–8.
- [31] P. Schneider, *Cluster lens reconstruction using only observed local data*, *Astronomy & Astrophysics* **302** (1995) 639–648.
- [32] P. Schneider and C. Seitz, *Steps towards nonlinear cluster inversion through gravitational distortions .1. basic considerations and circular clusters*, *Astronomy & Astrophysics* **294** (1995) 411–431.
- [33] S. Seitz and P. Schneider, *Cluster lens reconstruction using only observed local data: An improved finite-field inversion technique*, *Astronomy & Astrophysics* **305** (1996) 383–401.
- [34] C. Clarkson, *The general theory of secondary weak gravitational lensing*, *Journal of Cosmology and Astroparticle Physics* **2015** (2015) 033–033.
- [35] C. Clarkson, *Roulettes: a weak lensing formalism for strong lensing: I. overview*, *Classical and Quantum Gravity* **33** (2016) .
- [36] C. Clarkson, *Roulettes: a weak lensing formalism for strong lensing: Ii. derivation and analysis*, *Classical and Quantum Gravity* **33** (2016) .
- [37] B. D. Normann and C. Clarkson, *Recursion relations for gravitational lensing*, *General Relativity and Gravitation* **52** (Mar, 2020) .
- [38] R. Kormann, P. Schneider and M. Bartelmann, *Isothermal elliptic gravitational lens models*, *Astronomy & Astrophysics* **284** (1994) 285–299.
- [39] J. Hjorth and J.-P. Kneib, “Elliptical galaxies as gravitational lenses.”

# **Appendix B**

## **Pre-project report**

# FORPROSJEKT - RAPPORT

## FOR BACHELOROPPGAVE

TITTEL:

Intergalaktisk maskinsyn

KANDIDATNUMMER(E):

**ID 510316**

**ID 517305**

**ID 510350**

**ID 60381**

DATO:	EMNEKODE:	EMNE:	DOKUMENT TILGANG:
<b>23.01.2022</b>	<b>IELEA2920</b>	<b>Bacheloroppgave</b>	- Åpen
STUDIUM:	ANT SIDER/VEDLEGG:		BIBL. NR:
<b>AUTOMATISERING OG ROBOTIKK</b>	/		- Ikke i bruk -

OPPDRAGSGIVER(E)/VEILEDER(E):

Ben David Normann

Hans Georg Schaathun

OPPGAVE/SAMMENDRAG:

Dette er en forprosjektsrapport for en planlagt bacheloroppgave hvor målet er å utføre grunnarbeid for videre forskning på mørk materie. Mørk materie kan ikke observeres direkte, men materies gravitasjon vil påvirke lys og endre lysets bane på samme måte som ordinært materie. Ved observasjon av lysbøyningen kan dermed massen til materiet beregnes. Lyset vil påvirkes som ved en linse og man kan dermed sette opp linsemodeller som beskriver lysbøyningen. Ved å undersøke lys fra fjerne galakser skal man i teorien kunne lage et kart over konstellasjoner av mørk materie.

Målet med denne bacheloroppgaven er todelt. Det første målet er å produsere et simuleringsverktøy hvor ulike linsemodeller kan simuleres. Verktøyet skal være til bruk av forskere fra hele verden som ønsker å undersøke linsemodeller. Brukergrensesnittet må da utvikles med brukervennlighet som fokus. Det andre målet er å benytte dette verktøyet til å produsere treningsdata for en maskinlæringsalgoritme. Gjennom maskinlæring vil det være mulig å utvide verktøyets funksjonalitet. Man skal da kunne mate inn et vilkårlig forvrengt bilde av en fjern galakse og få ut et uforvrengt bilde. I tillegg vil utfordringer og begrensninger for videre forskingsarbeid drøftes.

**Postadresse**

NTNU  
Høgskolen i Ålesund  
N-6025 Ålesund  
Norway

**Besøksadresse**

Larsgårdsvegen 2  
**Internett**  
www.ntnu.no

**Telefon**

73 59 50 00

**Epostadresse**

[postmottak@ntnu.no](mailto:postmottak@ntnu.no)

**Foretaksregisteret**

NO 971 572 140





## INNHOOLD

<b>INNHOOLD</b> .....	<b>3</b>
<b>1 INNLEDNING</b> .....	<b>4</b>
<b>2 BEGREPER</b> .....	<b>3</b>
<b>3 PROSJEKTORGANISASJON</b> .....	<b>4</b>
3.1 PROSJEKTGRUPPE .....	4
3.1.1 Oppgaver for prosjektgruppen – organisering .....	4
3.1.2 Oppgaver for prosjektleder .....	4
3.1.3 Oppgaver for sekretær .....	5
3.1.4 Maskinlæringsansvarlig.....	5
3.1.5 Simulatoransvarlig.....	5
3.1.6 Felles ansvar.....	5
3.2 STYRINGSGRUPPE (VEILEDER OG KONTAKTPERSON OPPDRAGSGIVER) .....	5
<b>4 AVTALER</b> .....	<b>6</b>
4.1 AVTALE MED OPPDRAGSGIVER .....	6
4.2 ARBEIDSSTED OG RESSURSER .....	6
4.3 GRUPPENORMER – SAMARBEIDSREGLER – HOLDNINGER .....	6
<b>5 PROSJEKTBESKRIVELSE</b> .....	<b>6</b>
5.1 PROBLEMSTILLING - MÅLSETTING - HENSIKT .....	6
5.2 KRAV TIL LØSNING ELLER PROSJEKTRESULTAT – SPESIFIKASJON .....	6
5.3 PLANLAGT FRAMGANGSMÅTE(R) FOR UTVIKLINGSARBEIDET – METODE(R) .....	6
5.4 INFORMASJONSINNSAMLING – UTFØRT OG PLANLAGT .....	7
5.5 VURDERING – ANALYSE AV RISIKO .....	7
5.5.1 Nedstenging av samfunnet .....	7
5.5.2 Sykdom hos én person.....	8
5.5.3 Sykdom hos flere personer samtidig .....	8
5.5.4 Sykmelding pga. dårlig ergonomi .....	8
5.6 HOVEDAKTIVITETER I VIDERE ARBEID .....	8
5.7 FRAMDRIFTSPLAN – STYRING AV PROSJEKTET.....	9
5.7.1 Hovedplan.....	9
5.7.2 Styringshjelpemidler .....	10
5.7.3 Utviklingshjelpemidler .....	10
5.7.4 Intern kontroll – evaluering .....	10
5.8 BESLUTNINGER – BESLUTNINGSPROSESS .....	10
<b>6 DOKUMENTASJON</b> .....	<b>10</b>
6.1 RAPPORTER OG TEKNISKE DOKUMENTER.....	10
<b>7 PLANLAGTE MØTER OG RAPPORTER</b> .....	<b>11</b>
7.1 MØTER .....	11
7.1.1 Møter med styringsgruppen .....	11
7.1.2 Prosjektmøter.....	11
7.2 PERIODISKE RAPPORTER .....	11
7.2.1 Framdriftsrapporter (inkl. milepæl) .....	11
<b>8 PLANLAGT AVVIKSBEHANDLING</b> .....	<b>11</b>
<b>9 UTSTYRSBEHOV/FORUTSETNINGER FOR GJENNOMFØRING</b> .....	<b>11</b>

## 1 INNLEDNING

Mesteparten av universets materie er mørk materie. Mørk materie er ikke mulig å observere direkte. Det mørke materie kan observeres ved å undersøke hvordan gravitasjonsfeltet til annet materie forvrenger lys. Forvrengningen skjer som i en tradisjonell optisk linse. Konseptet kalles dermed *gravitasjonslinse*. Dersom man tar bilde av fjerne galakser, vil man kunne observere denne forvrengningen. Ut fra det forvrengte bilde burde man kunne finne egenskapene til gravitasjonslinsen og videre kunne tegne et kart over den mørke materien som forårsaker forvrengningen.

Oppdragsgiver er astrofysiker Ben David Normann og professor Hans Georg Schaathun ved NTNU Ålesund. De har som mål å starte et forskningsprosjekt hvor denne teorien skal settes på prøve i et forsøk på å kunne tegne et kart over mørk materie i universet. Denne bacheloroppgaven er en oppstart av dette forskningsarbeidet. Målet er å kunne produsere et simuleringsverktøy hvor ulike linsemodeller kan simuleres. Videre er målet å benytte dette verktøyet til å produsere treningsdata for en maskinlæringsalgoritme. Gjennom maskinlæring er målet å kunne mate inn et vilkårlig forvrengt bilde av en fjern galakse og få ut linsemodellen som skapte forvrengningen.

## 2 BEGREPER

- Mørk materie  
Materie i universet som ikke sender ut stråling, og som vi dermed ikke kan observere direkte. Bevis for eksistensen av mørk materie har kommet frem gjennom observasjoner av gravitasjonsvirkningen som den mørke materien har på synlig materie ([SNL](#)).
- Gravitasjonslinse  
Gravitasjonslinse, objekt i verdensrommet (for eksempel en planet, stjerne, sort hull, eller en galakse) som ligger mellom Jorden og en strålingskilde (for eksempel en stjerne, galakse, kvasar eller annet) på en slik måte at det gir opphav til optiske fenomener som fokusering, forsterkning og oppsplitting i flere bilder. Disse linseeffektene kommer av at elektromagnetisk stråling blir avbøyd i objektets gravitasjonsfelt, noe som er en konsekvens av Einsteins generelle relativitetsteori ([SNL](#)).
- Maskinlæring  
Maskinlæring er en spesialisering innen kunstig intelligens hvor man bruker statistiske metoder for å la datamaskiner finne mønstre i store datamengder. Vi sier at maskinen «lærer» i stedet for å bli programmert ([SNL](#)).

## 3 PROSJEKTORGANISASJON

### 3.1 Prosjektgruppe

Studentnummer(e)
ID 510316
ID 517305
ID 510350
ID 60381

Tabell: Studentnummer(e) for alle i gruppen som leverer oppgaven for bedømmelse i faget IELEA2920

### 3.1.1 Oppgaver for prosjektgruppen – organisering

Prosjektleder:	Sondre Westbø Remøy
Sekretær:	Simon Nedreberg Runde
Maskinlæringsansvarlig:	Simon Ingebrigtsen
Simulatoransvarlig:	Einar Leite Austnes

### 3.1.2 Oppgaver for prosjektleder

- Ansvarsområde
  - Prosjektorganisering
  - Ansvarsfordeling
  - Konflikthåndtering
  - Møter
- Arbeidsoppgaver
  - Dialog med veiledere
  - Møteinnkalling
  - Oppdatere oversikt over oppgaveansvarlig
  - Ferdigstille dokumentasjon

### 3.1.3 Oppgaver for sekretær

- Ansvarsområde
  - Prosjektplan
- Arbeidsoppgaver
  - Oppdatere fremgangsskjema ved 14. dags milepæl
  - Skrive møtereferat
  - Assistere prosjektleder i ferdigstillelse av dokumentasjon

### 3.1.4 Maskinlæringsansvarlig

- Ansvarsområder
  - Algoritmer
  - Spesialisering i maskinlæring
- Arbeidsoppgaver
  - Implementasjon av maskinlæringsalgoritmer
  - Oppdatere Github

### 3.1.5 Simulatoransvarlig

- Ansvarsområder
  - Simulatorapplikasjon
  - Brukergrensesnitt
  - Spesialisering i implementasjon av brukergrensesnitt
- Arbeidsoppgaver
  - Utvikling av simulator
  - Oppdatere Github

### **3.1.6 Felles ansvar**

Alle gruppemedlemmer har et felles ansvar å bidra til alle arbeidsoppgaver. Dette for å sikre overkommelig arbeidsmengde uavhengig av ansvarsområde, samt læring og utvikling hos samtlige. Hvert gruppemedlem skal gi beskjed ved behov for assistanse, noe som skal imøtekommes av gruppen.

## **3.2 Styringsgruppe (veileder og kontaktperson oppdragsgiver)**

## **4 AVTALER**

### **4.1 Avtale med oppdragsgiver**

Det er avtalt møter med oppdragsgivere hver 14. dag fra og med onsdag 19.01.2022. Oppdragsgiver oppfordrer til å også ta kontakt utenom disse møtene dersom det er behov.

Hans Georg skal undersøke mulighetene for å få benytte Idun for oppgaver som krever høyere prosessorkraft.

### **4.2 Arbeidssted og ressurser**

Oppdragsgiver:

Ben David Normann og Hans Georg Schaathun ved NTNU - Department of ICT and Natural Sciences

Arbeidssted:

NTNU Ålesund

Ressurser:

- Idun High Performance Computing Group
- Veiledere
  - Hans Georg Schaathun og Ben David Normann

### **4.3 Gruppenormer – samarbeidsregler – holdninger**

Samtlige gruppemedlem skal:

- Overholde avtaler om møtetidspunkt og melde fra dersom man er forhindret i å møte.
- Oppdatere resten av gruppen om utvikling i eget arbeid.
- Føre timeliste og loggføre eget arbeid.
- Avdekke avvik.
- Bistå i kontinuerlig oppdatering av prosjektplan.
- Være objektive og holde personlige meninger adskilt fra arbeidet.

En god ingeniør arbeider metodisk og strukturert, dokumenterer sitt arbeid og setter bærekraft i fokus. Den vitenskapelige metoden er malen for utførelse av forskningsarbeidet i dette prosjektet og egeninteresser og personlige meninger skal ikke komme i veien for fakta og objektivitet. Prosjektet skal være et forarbeid for videre forskning. Gruppen har derfor som mål å gjøre et grundig arbeid slik at videre forskning kan baseres på et solid og grundig forarbeid.

## **5 PROSJEKTBEKRIVELSE**

### **5.1 Problemstilling - målsetting - hensikt**

Denne bacheloroppgaven har som mål å utføre grunnarbeid for videre forskning på mørk materie. Mørk materie kan ikke observeres direkte, men materies gravitasjon vil påvirke lys og endre lysets bane på samme måte som lyst materie. Observasjon av lysbøyingen kan gi en indikasjon på massen til materie. Lyset vil påvirkes som ved en linse og man kan dermed sette opp linsemodeller som beskriver

lysbøyingen. Ved å undersøke lys fra fjerne galakser skal man i teorien kunne lage et kart over konstellasjoner av mørk materie.

Målet med denne bacheloroppgaven er todelt. Det første målet er å produsere et simuleringsverktøy hvor ulike linsemodeller kan simuleres. Verktøyet skal være til bruk av forskere fra hele verden som ønsker å undersøke linsemodeller. Brukergrensesnittet må da utvikles med brukervennlighet som fokus. Det andre målet er å benytte dette verktøyet til å produsere treningsdata for en maskinlæringsalgoritme. Gjennom maskinlæring vil det være mulig å utvide verktøyets funksjonalitet. Man skal da kunne mate inn et vilkårlig forvrengt bilde av en fjern galakse og få ut et uforvrengt bilde. I tillegg vil utfordringer og begrensninger for videre forskingsarbeid drøftes.

## 5.2 Krav til løsning eller prosjektresultat – spesifikasjon

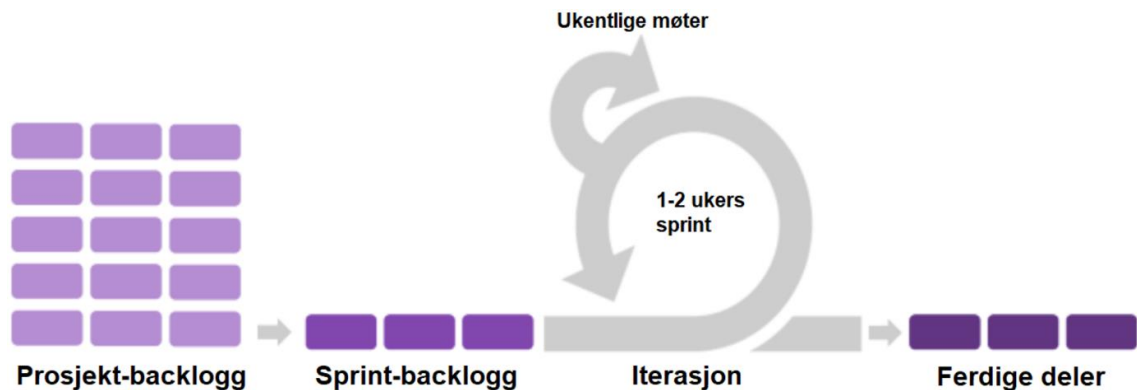
- Simulatoren skal:
  - Kunne ta inn flere ulike linsemodeller.
  - Ha støtte for kommandolinjeinput slik at simulatoren kan skape datasett til bruk for maskinlæring.
  - Ha et brukervennlig brukergrensesnitt for simulatoren.
- Det nevrale nettet skal kunne ta inn et vilkårlig forvrengt bilde og gi brukeren parameterne for linsemodellen.

### *Merk:*

Sluttkravene til den ferdige løsningen kan endres underveis da dette er et forskningsarbeid og verken gruppen eller veilederne vet hvilke problemstillinger eller begrensninger som vil dukke opp.

## 5.3 Planlagt framgangsmåte(r) for utviklingsarbeidet – metode(r)

Prosjektarbeidet er planlagt å utføres etter en *smidig utviklingsmetodikk*.



Figur 5.1 Illustrasjon av smidig utviklingsmetodikk

Først skal det produseres en simulator. Dette fordi det er simulatoren som skal generere datasettene som maskinlæringsalgoritmen skal trenes med. Så snart en tilfredsstillende simulator er laget, vil arbeidet med maskinlæring starte. Parallelt med dette vil simulatorens funksjoner og brukergrensesnitt utvikles og forbedres.

For utviklingen av brukergrensesnitt skal brukerhistorier legges til grunn.

## 5.4 Informasjonsinnsamling – utført og planlagt

Gruppen har satt seg inn i arbeidet til Chris Clarkson med gravitasjonslinser og Roulette-linsemodellen han har utviklet. Ben David Normann har også produsert et introduksjonspaper med hensikt å familiarisere gruppen med nødvendige konsept. Veilederne har opprettet en samling med litteratur som potensielt kan bli nyttig for prosjektet i en wiki på tjenesten Confluence.

## 5.5 Vurdering – analyse av risiko

Figur 5.1 viser en risikomatrix hvor grønt er optimalt, gult er akseptabelt og rødt nivå uakseptabelt. Da må tiltak iverksettes.

Sannsynlighet/ konsekvens	1 Svært lite sannsynlig	2 Lite sannsynlig	3 Sannsynlig	4 Ganske sannsynlig	5 Svært sannsynlig
5 Katastrofal	Yellow	Yellow	Red	Red	Red
4 Svært stor	Green	Yellow	Yellow	Red	Red
3 Stor	Green	Green	Yellow	Yellow	Red
2 Middels	Green	Green	Green	Yellow	Yellow
1 Liten	Green	Green	Green	Green	Yellow

Figur 5.2 Risikomatrix (kilde: <https://www.bfobrann.no/eksempler/risikovurdering>)

### 5.5.1 Nedstenging av samfunnet

#### Risiko uten tiltak

Sannsynlighet: Ganske sannsynlig

Konsekvens: Svært stor

Rød

#### Tiltak

*Konsekvens:*

Lagre alt arbeid i skyløsninger slik at alle gruppelemmer har tilgang til dette uansett hvor de befinner seg. Alle er også oppmerksomme på at alle fremtidige møter kan bli digitale.

#### Risiko med tiltak

Sannsynlighet: Ganske sannsynlig

Konsekvens: Liten

Grønn

### 5.5.2 Sykdom hos én person

#### Risiko uten tiltak

Sannsynlighet: Svært sannsynlig

Konsekvens: Stor

Rød

#### Tiltak

*Sannsynlighet:*

God håndhygiene og minimering av kontakt med andre mennesker, noe som allerede ligger til grunn som en følge av COVID-19.

*Konsekvens:*

God kommunikasjon slik at refordeling av arbeidsoppgaver kan skje ved en evt. sykdomssituasjon.

#### Risiko med tiltak

Sannsynlighet: Sannsynlig

Konsekvens: Middels

Grønn

### 5.5.3 Sykdom hos flere personer samtidig

#### Risiko uten tiltak

Sannsynlighet: Ganske sannsynlig

Konsekvens: Svært stor

Rød

**Tiltak**

*Sannsynlighet:*

God håndhygiene og minimering av kontakt med andre mennesker, noe som allerede ligger til grunn som en følge av COVID-19.

*Konsekvens:*

Holde seg til prosjektplanen da denne tar høyde for forsinkelser.

**Risiko med tiltak**

Sannsynlighet: Lite sannsynlig

Konsekvens: Stor

Grønn

### 5.5.4 Sykmelding pga. dårlig ergonomi

**Risiko uten tiltak**

Sannsynlighet: Svært sannsynlig

Konsekvens: Stor

Rød

**Tiltak**

*Sannsynlighet:*

Gruppemedlem gjøres oppmerksomme på de potensielle konsekvensene av dårlig ergonomi under arbeidet, samt den forebyggende effekten til mosjon og bevegelse..

*Konsekvens:*

God kommunikasjon slik at tidlig refordeling av arbeidsoppgaver kan skje for å redusere belastning.

**Risiko med tiltak**

Sannsynlighet: Lite sannsynlig

Konsekvens: Middels

Grønn

### 5.6 Hovedaktiviteter i videre arbeid

Nr.	Hovedaktivitet	Ansvar	Kostnad	Tid/omfang
A1	Forprosjektrapport	SWR / SNR	--	8 virkedager
A11	Ansvarsfordeling	SWR / SNR	--	2 timer
A12	SJA	SWR / SNR	--	2 timer
A14	Samarbeidsavtale	SWR / SNR	--	1 time
A15	Veiledermøte	SWR / SNR	--	5 timer
A16	Skrive forprosjektrapport	SWR / SNR	--	6 virkedager
A17	Planlegge møter	SWR / SNR	--	1 time
B1	<b>Simulator</b>	ELA / SI	--	Ferdig før mai – kontinuerlig smidig utvikling
D1	<b>Utvikle GUI</b>	SNR / ELA	--	1 mnd.
B12	Punktmasse modell	SI / SNR	--	1 uke
B13	Spherical modell	ELA / SI	--	1 uke
B14	Ellipsoidal modell	ELA / SI	--	1 uke
B15	Generering av syntetisk data	ELA / SI	--	2 uker
B16	Reversering av modell(er)	ELA / SI	--	1 uke
C1	<b>Maskinlæringsmodell</b>	SI / SWR	--	2 mnd.
C11	Research av ML bibliotek til python	SI / SWR	--	2 uker
C12	Implementere klassifiseringsnettverk	SNR / SI	--	2 uker
C13	Implementere rekonstruksjonsnettverk	SI / SNR	--	2 uker
D1	Rapport		--	Ferdig til 20. mai 2022



## 5.7 Framdriftsplan – styring av prosjektet

### 5.7.1 Hovedplan

Først skal det produseres en simulator. Dette fordi det er simulatoren som skal generere datasettene som maskinlæringsalgoritmen skal trenes med. Så snart en tilfredsstillende simulator er laget, vil arbeidet med maskinlæring starte. Parallelt med dette vil simulatorens funksjoner og brukergrensesnitt utvikles og forbedres.

For utviklingen av brukergrensesnitt skal brukerhistorier legges til grunn for funksjoner og utvikling av brukergrensesnitt.

Gruppen planlegger å jobbe ut ifra en smidig utviklingsmetodikk. Dette betyr at det unngås å planlegge mer enn en til to uker frem i tid. Oppgaven er ganske løst definert i utgangspunktet og den er basert på pågående forskning, så oppgaven kommer til å utvikle seg underveis. Listen i pkt. 5.6 er derfor kun et utgangspunkt for fremdriften.

Milepæler:

- Fullføre simulator for punktmasse-linse
- Utvide simulator til sfæriske og ellipsoide linsemodeller.
- Generering av syntetisk data.
- Implementere nevral nett for klassifisering av linser (punktmasse vs. ingen forvrengning)
- Utvide klassifisering til sfærisk og ellipsoidisk linse, evt hop og sterk/svak linse
- Implementere nevral nett for rekonstruksjon av punktmasse-linse.
- Utvide til rekonstruksjon av sfærisk og ellipsoidisk modell.
- Utvikle et brukervennlig grensesnitt for simulering.

Pga. den smidige utviklingsmodellen blir tidsfrister på milepæler og beslutningspunkter ikke så relevant for dette prosjektet.

### 5.7.2 Styringshjelpemidler

- Confluence
- Forprosjektsrapport
- Github (issues-modul)
- Trello

### 5.7.3 Utviklingshjelpemidler

Fysiske hjelpemidler:

- PC
- Evt. Idun high performance computing group for trening av nevral nett

Programvare:

- CLion/ MSVS for utvikling av C++ program
- Qt Creator for utvikling av brukergrensesnitt
- PyCharm/ MSVS for utvikling av Python program
- Github (versjonskontroll)

### 5.7.4 Intern kontroll – evaluering

Prosjektets fremdrift blir gjennomgått på ukentlige gruppemøter i tillegg til møter med veiledere hver 14. dag.

## 5.8 Beslutninger – beslutningsprosess

Viktige/større beslutninger tas gjennom diskusjoner i gruppen. Ved vedvarende uenigheter rundt arbeidet vil veiledere også involveres i diskusjonen.

## 6 DOKUMENTASJON

### 6.1 *Rapporter og tekniske dokumenter*

- Forprosjektrapport
- Møtereferat
- Fremdriftsrapport hver 14. dag
- Endelig bachelorrappport leveres 20.05.2022.

## 7 PLANLAGTE MØTER OG RAPPORTER

### 7.1 *Møter*

#### 7.1.1 Møter med styringsgruppen

- Hver 14. dag fra og med 19.01.2022.
  - o Møtene skal brukes til å diskutere videre arbeid med veiledere.
  - o Det skal også gis oppdatering fra veiledere om forskningens fremdrift og veiledernes evt. nye ønsker til sluttkrav.

#### 7.1.2 Prosjektmøter

- Gruppen planlegger å møtes hver mandag for å gå gjennom foregående ukes fremdrift, planlegge arbeid for kommende uke, samt utføre arbeid som krever samarbeid.
- Standup-møter blir gjennomført når det er behov for rask briefing.

### 7.2 *Periodiske rapporter*

#### 7.2.1 Framdriftsrapporter (inkl. milepæl)

- Framdriftsrapport vil produseres hver 14. dag i forkant av veiledningsmøtene.
  - Sendes til veiledere på morgenen slik at den kan gjennomgås på veiledningsmøte
- Møtereferat vil produseres etter alle offisielle møter.

## 8 PLANLAGT AVVIKSBEHANDLING

- Ansvar for fremgang i prosjektet ligger hos gruppeleder.
- Alle gruppemedlemmer er ansvarlige for å avdekke og melde avvik. Avvik skal meldes til hele gruppen.
- Ved samarbeidskonflikter skal gruppeleder varsles, gruppeleder har ansvaret for å løse konflikten(e).
- Alle endringer skal komme frem ved enighet av majoriteten i gruppa, samt godkjennes av oppdragsgiver. Sekretær har da ansvar for å oppdatere prosjektplan med vedtatte endringer.

## 9 UTSTYRSBEHOV/FORUTSETNINGER FOR GJENNOMFØRING

- Idun High Performance Computing Group

# **Appendix C**

## **Progress reports**

### **C.1 Progress report 02.02.2022**

<b>IELEA2920</b> <b>Bachelor Thesis</b> <b>Automation</b>	Project Inter-galactic machine vision	Number of meeting this period 1). 1 meeting with counsellors 1 official group meeting	Client NTNU Ålesund	Page 1 of 2
<b>Progress report</b>	Period/week(s) 2	Number of hours this period. (from log) Approx. 200	Group members Sondre Remøy, Simon Runde, Simon Ingebrigtsen, Einar Austnes	Date 02.02.22

<p>Main goal/purpose for this periods work</p> <ul style="list-style-type: none"> <li>- Fix/improve simulator</li> <li>- Document simulator</li> </ul>
<p>Planned activities this period</p> <ul style="list-style-type: none"> <li>- Make plots from simulator realistic compared to lens-model.</li> <li>- Remove plotting bugs</li> <li>- Produce documentation of simulators working principle for the ECMS-article</li> </ul>
<p>Actually conducted activites this period</p> <ul style="list-style-type: none"> <li>- Simulator mathematics were fixed</li> <li>- Added visualization of actual and apparent source position</li> <li>- Added visualization of Einstein radius</li> <li>- Creation of datasets with randomized parameters</li> <li>- First implementation of CNN</li> <li>- Research towards GUI-implementation</li> </ul>
<p>Description of/ justification for potential deviation between planned and real activities</p> <ul style="list-style-type: none"> <li>- No major deviations from planned activities. Activities performed exceeded number of planned activities. For next period more activities may be planned but might not be possible due to the parallel subject that started during current period.</li> </ul>
<p>Description of/ justification for changes that is desired in the projects content or in the further plan of action – or progress report</p> <ul style="list-style-type: none"> <li>- No desired changes yet</li> </ul>
<p>Main experience from this period</p> <ul style="list-style-type: none"> <li>- Implementation of point mass model was more complicated than expected. Coordinate systems was not initially interpreted correctly and the formula for conversion between apparent and actual position was somewhat difficult to understand. Both problems got solved during a meeting with Ben David where the math was discussed.</li> </ul>
<p>Main purpose/focus next period</p> <ul style="list-style-type: none"> <li>- Further development of point mass model</li> <li>- Further development of neural network</li> <li>- Start Qt GUI</li> <li>- Start writing on the final report</li> </ul>
<p>Planned activities next period</p> <ul style="list-style-type: none"> <li>- Enable y-axis for model, currently only working on x-axis</li> <li>- Research for improving neural network</li> <li>- Create a simple GUI using Qt creator</li> <li>- Start writing chapter for theory</li> </ul>
<p>Other</p>

<b>IELEA2920</b> <b>Bachelor Thesis</b> <b>Automation</b>	Project Inter-galactic machine vision	Number of meeting this period 1). 1 meeting with counsellors 1 official group meeting	Client NTNU Ålesund	Page 2 of 2
<b>Progress report</b>	Period/week(s) 2	Number of hours this period. (from log) Approx. 200	Group members Sondre Remøy, Simon Runde, Simon Ingebrigtsen, Einar Austnes	Date 02.02.22

Wish/need for counselling

- Nothing in particular

Approval/signature group leader

Sondre Remøy

Sondre Remøy

Signature other group participants

Einar Austnes

Einar Austnes

Simon Runde

Simon Runde

Simon Ingebrigtsen

Simon Ingebrigtsen

**C.2 Progress report 02.03.2022**

<b>IELEA2920 Bachelor Thesis Automation</b>	Project Inter-galactic machine vision	Number of meeting this period 1). 1 meeting with counsellors 2 lectures 2 official group meeting	Client NTNU Ålesund	Page 1 of 2
<b>Progress report</b>	Period/week(s) <b>Week 9 4 week period</b>	Number of hours this period. (from log) Approx. 200	Group members Sondre Remøy, Simon Runde, Simon Ingebrigtsen, Einar Austnes	Date 02.03.22

<p>Main goal/purpose for this periods work</p> <ul style="list-style-type: none"> <li>- Further development of point mass model</li> <li>- Further development of neural network</li> <li>- Start Qt GUI</li> <li>- Start writing on the final report</li> </ul>
<p>Planned activities this period</p> <ul style="list-style-type: none"> <li>- Enable y-axis for model, currently only working on x-axis</li> <li>- Research for improving neural network</li> <li>- Create a simple GUI using Qt creator</li> <li>- Start writing chapter for theory</li> </ul>
<p>Actually conducted activites this period</p> <ul style="list-style-type: none"> <li>- Created QT GUI template</li> <li>- Y-axis implemented for model</li> <li>- Changed simulator so it keeps full resolution throughout the process</li> <li>- Gained access to Idun. Managed to run some training on cnn.</li> <li>- Documented some of the mathematics and algorithms used in the simulator in the thesis.</li> </ul>
<p>Description of/ justification for potential deviation between planned and real activities</p> <p>No major deviations from plan, but the work period had to be doubled due to illness. Given the longer work period, more progress in planned activities might be expected, but this was not the case due to COVID-19 infection in the group and higher workload in parallel subject than initially expected.</p> <p>For next period less activities are planned due to deadline of project and final exam in the parallel subject.</p>
<p>Description of/ justification for changes that is desired in the projects content or in the further plan of action – or progress report</p> <ul style="list-style-type: none"> <li>- No desired changes yet</li> </ul>
<p>Main experience from this period</p> <ul style="list-style-type: none"> <li>- Gained deeper understanding of gravitational lensing.</li> <li>- Some experience with basic Qt Gui development</li> <li>- A little experience with using Idun</li> <li>- General programming experience from improving the simulator</li> </ul>
<p>Main purpose/focus next period</p> <ul style="list-style-type: none"> <li>- Start implementation of spherical model</li> <li>- Implement simulator in Qt Creator</li> </ul>
<p>Planned activities next period</p> <ul style="list-style-type: none"> <li>- Research and test libraries for differentiation in C++</li> <li>- Refactor simulator code and implement it in Qt Creator (GUI development tool)</li> </ul>
<p>Other</p>




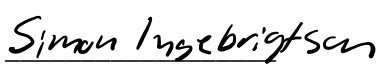
<b>IELEA2920 Bachelor Thesis Automation</b>	Project Inter-galactic machine vision	Number of meeting this period 1). 1 meeting with counsellors 2 lectures 2 official group meeting	Client NTNU Ålesund	Page 2 of 2
<b>Progress report</b>	Period/week(s) <b>Week 9 4 week period</b>	Number of hours this period. (from log) Approx. 200	Group members Sondre Remøy, Simon Runde, Simon Ingebrigtsen, Einar Austnes	Date 02.03.22

Wish/need for counselling - Nothing in particular	
Approval/signature group leader Sondre Remøy <u>Sondre Remøy</u>	Signature other group participants Einar Austnes <u>Einar Austnes</u> Simon Runde <u>Simon Runde</u> Simon Ingebrigtsen <u>Simon Ingebrigtsen</u>



### **C.3 Progress report 25.03.2022**

<b>IELEA2920 Bachelor Thesis Automation</b>	Project Inter-galactic machine vision	Number of meeting this period 1). 1 meeting with counsellors	Client NTNU Ålesund	Page 1 of 1
<b>Progress report</b>	Period/week(s) 3-week period	Number of hours this period. (from log) Approx. 20	Group members Sondre Remøy, Simon Runde, Simon Ingebrigtsen, Einar Austnes	Date 25.03.22

Main goal/purpose for this periods work	
<ul style="list-style-type: none"> <li>- Start implementation of spherical model</li> <li>- Implement simulator in Qt Creator</li> </ul>	
Planned activities this period	
<ul style="list-style-type: none"> <li>- Research and test libraries for differentiation in C++</li> <li>- Refactor simulator code and implement it in Qt Creator (GUI development tool)</li> </ul>	
Actually conducted activities this period	
<ul style="list-style-type: none"> <li>- Implemented simulator code in QT Creator. Several features in the GUI still missing.</li> <li>- More documentation of the simulator and mathematics in the report.</li> </ul>	
Description of/ justification for potential deviation between planned and real activities	
No major deviations from plan, but the work period was extended with one week due to deadline of project and final exam in the parallel subject.	
Description of/ justification for changes that is desired in the projects content or in the further plan of action – or progress report	
<ul style="list-style-type: none"> <li>- No desired changes yet</li> </ul>	
Main experience from this period	
<ul style="list-style-type: none"> <li>- Further experience with programming GUI</li> </ul>	
Main purpose/focus next period	
<ul style="list-style-type: none"> <li>- Further develop GUI</li> <li>- Implementation of spherical model</li> <li>- Continue writing the report</li> </ul>	
Planned activities next period	
<ul style="list-style-type: none"> <li>- Add more features to GUI, some buttons currently do nothing</li> <li>- Working GUI executable for windows, linux and mac</li> <li>- More research on libraries for differentiation in C++</li> </ul>	
Other	
Wish/need for counselling	
<ul style="list-style-type: none"> <li>- Nothing in particular</li> </ul>	
Approval/signature group leader	Signature other group participants
Sondre Remøy 	Einar Austnes  Simon Runde 
	Simon Ingebrigtsen 

**C.4 Progress report 06.04.2022**

<b>IELEA2920 Bachelor Thesis Automation</b>	Project Inter-galactic machine vision	Number of meeting this period 1). 1 meeting with counsellors	Client NTNU Ålesund	Page 1 of 2
<b>Progress report</b>	Period/week(s)  < 2-week period	Number of hours this period. (from log) Approx. 300	Group members Sondre Remøy, Simon Runde, Simon Ingebrigtsen, Einar Austnes	Date 06.04.22

<p>Main goal/purpose for this periods work</p> <ul style="list-style-type: none"> <li>- Further develop GUI in Qt Creator (GUI development tool)</li> <li>- Implementation of spherical model</li> <li>- Continue writing the report</li> </ul>
<p>Planned activities this period</p> <ul style="list-style-type: none"> <li>- Add more features to GUI, some buttons currently do nothing.</li> <li>- Working GUI executable for windows, linux and mac.</li> <li>- More research on libraries for differentiation in C++.</li> </ul>
<p>Actually conducted activities this period</p> <ul style="list-style-type: none"> <li>- Refactored GUI-code as pure Qt-project. Now using Qt's built-in visualization-tools instead of OpenCV.</li> <li>- Added more source types to GUI tool.</li> <li>- GUI now compatible with Windows, Linux and MacOS (hopefully).</li> <li>- Replaced the previously implemented convolution network with a modified AlexNet. Loss for trained models have now been significantly reduced.</li> <li>- ML is now able to determine the relationship of lens and source distance; <math>X_l/X_s</math></li> <li>- Read up on spherical model. Made a preliminary plan on how to implement the simulator. <ul style="list-style-type: none"> <li>- Prepared questions for Ben David regarding implementation.</li> </ul> </li> </ul>
<p>Description of/ justification for potential deviation between planned and real activities</p> <ul style="list-style-type: none"> <li>- Nothing new has been written in the report. Focus has been on programming.</li> </ul>
<p>Description of/ justification for changes that is desired in the projects content or in the further plan of action – or progress report</p> <ul style="list-style-type: none"> <li>- No desired changes yet</li> </ul>
<p>Main experience from this period</p> <ul style="list-style-type: none"> <li>- Experience with visualization tools in Qt Creator</li> <li>- Limitations of the neural networks previously implemented</li> <li>- Experience with implemented AlexNet convolution network which seems to give better results.</li> <li>- More experience with Idun.</li> </ul>
<p>Main purpose/focus next period</p> <ul style="list-style-type: none"> <li>- More training of AlexNet on Idun</li> <li>- Find the networks limitations</li> <li>- Finish the implementation of elliptical model.</li> <li>- Collect user stories and expand GUI functionality</li> </ul>
<p>Planned activities next period</p> <ul style="list-style-type: none"> <li>- Train network on Idun with dataset of <math>\geq 500\ 000</math> generated images.</li> <li>- Map the limitations of AlexNet through training on large datasets. <ul style="list-style-type: none"> <li>- When limitations of the network are known (i.e., which parameters is it able/unable to accurately predict), a hypothesis for the cause must be produced in order to potentially improve the trained model.</li> </ul> </li> <li>- Add or remove buttons and functionalities in accordance with wishes from user stories.</li> </ul>

<b>IELEA2920 Bachelor Thesis Automation</b>	Project Inter-galactic machine vision	Number of meeting this period 1). 1 meeting with counsellors	Client NTNU Ålesund	Page 2 of 2
<b>Progress report</b>	Period/week(s)  < 2-week period	Number of hours this period. (from log) Approx. 300	Group members Sondre Remøy, Simon Runde, Simon Ingebrigtsen, Einar Austnes	Date 06.04.22

<ul style="list-style-type: none"> <li>- Clarify mathematics for the spherical model to give an accurate implementation <ul style="list-style-type: none"> <li>- Try to implement a spherical model in the simulator</li> <li>- Conduct meeting with Ben David for discussion, more info in “Wish/need for counselling”.</li> </ul> </li> </ul>	
Other N/A	
Wish/need for counselling <ul style="list-style-type: none"> <li>- Some of the calculations for implementing elliptical model needs clarification with Ben David. Planned meeting: 06.04. 13:30-14:30</li> </ul>	
Approval/signature group leader Sondre Remøy <u>Sondre Remøy</u>	Signature other group participants Einar Austnes <u>Einar Austnes</u> Simon Runde <u>Simon Runde</u>  Simon Ingebrigtsen <u>Simon Ingebrigtsen</u>

**C.5 Progress report 27.04.2022**

<b>IELEA2920 Bachelor Thesis Automation</b>	Project Inter-galactic machine vision	Number of meeting this period 1). 1 meeting with counsellors	Client NTNU Ålesund	Page 1 of 2
<b>Progress report</b>	Period/week(s) 3-week period	Number of hours this period. (from log) Approx. 300	Group members Sondre Remøy, Simon Runde, Simon Ingebrigtsen, Einar Austnes	Date 27.04.22

<p>Main goal/purpose for this periods work</p> <ul style="list-style-type: none"> <li>- More training of AlexNet on Idun</li> <li>- Find the networks limitations</li> <li>- Finish the implementation of spherical model.</li> <li>- Collect user stories and expand GUI functionality</li> </ul>
<p>Planned activities this period</p> <ul style="list-style-type: none"> <li>- Train network on Idun with dataset of &gt; 500 000 generated images.</li> <li>- Map the limitations of AlexNet through training on large datasets. <ul style="list-style-type: none"> <li>- When limitations of the network are known (i.e., which parameters is it able/unable to accurately predict), a hypothesis for the cause must be produced in order to potentially improve the trained model.</li> </ul> </li> <li>- Add or remove buttons and functionalities in accordance with wishes from user stories.</li> <li>- Clarify mathematics for the spherical model to give an accurate implementation <ul style="list-style-type: none"> <li>- Try to implement a spherical model in the simulator</li> <li>- Conduct meeting with Ben David for discussion, more info in “Wish/need for counselling”.</li> </ul> </li> </ul>
<p>Actually conducted activities this period</p> <ul style="list-style-type: none"> <li>- Written sample text for feedback from counsellors. <ul style="list-style-type: none"> <li>- Received feedback and made changes in the final report.</li> </ul> </li> <li>- Implemented a somewhat working simulator for the spherical (SIS) model as planned. Managed to generate the expressions for the amplitudes by using symbolic math in python and implement these in a prototype-simulator in c++. This approach looks promising, although the output does not look quite correct. This could be due to insufficient terms in the summation or some other mistake in the code.</li> <li>- GUI-application: <ul style="list-style-type: none"> <li>- Now fully functional on Linux, Mac and Windows.</li> <li>- Conferred with main user, Ben David, about functionalities. The main functions are in place, but SIS-model is not yet implemented.</li> <li>- Image display will resize together with main window</li> <li>- Added ability to change resolution during runtime</li> <li>- Added different grid sizes which can be changed from the top View menu</li> <li>- Button in GUI and File menu to save images</li> </ul> </li> <li>- Conducted meeting with Ben David. This clarified the mathematical problems that had come up during the last period.</li> <li>- AlexNet and Inception3 has been training on Idun with 500k pictures dataset. Implemented multiple GPU (but single node) training on IDUN. The large dataset training has not shown any improvement in model accuracy.</li> </ul>
<p>Description of/ justification for potential deviation between planned and real activities</p> <p>-</p>
<p>Description of/ justification for changes that is desired in the projects content or in the further plan of action – or progress report</p> <ul style="list-style-type: none"> <li>- No desired changes yet</li> </ul>
<p>Main experience from this period</p>

<b>IELEA2920 Bachelor Thesis Automation</b>	Project Inter-galactic machine vision	Number of meeting this period 1). 1 meeting with counsellors	Client NTNU Ålesund	Page 2 of 2
<b>Progress report</b>	Period/week(s) 3-week period	Number of hours this period. (from log) Approx. 300	Group members Sondre Remøy, Simon Runde, Simon Ingebrigtsen, Einar Austnes	Date 27.04.22

<ul style="list-style-type: none"> <li>- Experience with deployment on multiple platforms</li> <li>- Symbolic math, multi-threading/-processing and saving, loading and parsing data in python and c++</li> </ul>	
<p>Main purpose/focus next period</p> <ul style="list-style-type: none"> <li>- Write the final report.</li> <li>- Clarify the extent of the SIS-model.</li> </ul>	
<p>Planned activities next period</p> <ul style="list-style-type: none"> <li>- Create finished text from all the information in our notes.</li> <li>- Make more drafts that counsellors can proofread.</li> <li>- Conduct testing and seek counselling for the implementation of SIS-model. If successful, implement it in the finished application.</li> <li>- As the estimation of all five parameteres of the point mass lens seems to be too difficult to get accurate, the next natural step would be to build a net that estimates the x- and y-position, Einstein radius and source size assuming <math>\chi_l/\chi_s</math> is known.</li> </ul>	
<p>Other</p> <p>N/A</p>	
<p>Wish/need for counselling</p> <ul style="list-style-type: none"> <li>- Some of the calculations for implementing SIS-model needs clarification with counsellors. Planned meeting: 27.04. 12:00-13:00</li> </ul>	
<p>Approval/signature group leader</p> <p>Sondre Remøy</p> <p><u>Sondre Remøy</u></p>	<p>Signature other group participants</p> <p>Einar Austnes</p> <p><u>Einar Austnes</u></p> <p>Simon Runde</p> <p><u>Simon Runde</u></p> <p>Simon Ingebrigtsen</p> <p><u>Simon Ingebrigtsen</u></p>



## **C.6 Progress report 11.05.2022**

<b>IELEA2920 Bachelor Thesis Automation</b>	Project Inter-galactic machine vision	Number of meeting this period 1). 1 meeting with counsellors	Client NTNU Ålesund	Page 1 of 2
<b>Progress report</b>	Period/week(s) 2-week period	Number of hours this period. (from log) Approx. 300	Group members Sondre Remøy, Simon Runde, Simon Ingebrigtsen, Einar Austnes	Date 11.05.22

<p>Main goal/purpose for this periods work</p> <ul style="list-style-type: none"> <li>- Write the final report.</li> <li>- Clarify the extent of the SIS-model.</li> </ul>
<p>Planned activities this period</p> <ul style="list-style-type: none"> <li>- Create finished text from all the information in our notes.</li> <li>- Make more drafts that counsellors can proofread.</li> <li>- Conduct testing and seek counselling for the implementation of SIS-model. If successful, implement it in the finished application.</li> <li>- As the estimation of all five paramterers of the point mass lens seems to be too difficult to get accurate, the next natural step would be to build a net that estimates the x- and y-position, Einstein radius and source size assuming <math>\chi_1/\chi_s</math> is known.</li> </ul>
<p>Actually conducted activities this period</p> <ul style="list-style-type: none"> <li>- Written a draft of the final report for feedback from supervisors. <ul style="list-style-type: none"> <li>- Received feedback and started making changes in the final report.</li> </ul> </li> <li>- Implemented a multi modal variation of the AlexNet, taking chi as an input as well as the image.</li> <li>- Discussed SIS- model and potential sources of error with Ben David. Further development on SIS-model put on ice for now.</li> <li>- Implemented the finite terms version of the point mass lens in the GUI.</li> <li>- Small tweaks and minor improvements on GUI.</li> </ul>
<p>Description of/ justification for potential deviation between planned and real activities</p> <ul style="list-style-type: none"> <li>- The extent of remaining work to implement a fully functional SIS-model seems too big to be finished on time.</li> </ul>
<p>Description of/ justification for changes that is desired in the projects content or in the further plan of action – or progress report</p> <ul style="list-style-type: none"> <li>- No desired changes yet</li> </ul>
<p>Main experience from this period</p> <ul style="list-style-type: none"> <li>- A better insight in proper writing style and language for the report.</li> <li>-</li> </ul>
<p>Main purpose/focus next period</p> <ul style="list-style-type: none"> <li>- Finish final report</li> <li>- Make video</li> <li>- Design poster</li> </ul>
<p>Planned activities next period</p> <ul style="list-style-type: none"> <li>- Make appropriate changes to the final report based on the feedback from supervisors.</li> <li>- Add text to the unfinished sections of the report.</li> <li>- Document simulator and ML on video.</li> <li>- Record voice-over for each part of the video.</li> <li>- Create a design for the poster and send to printing.</li> </ul>
<p>Other</p> <p>N/A</p>
<p>Wish/need for counselling</p>

<b>IELEA2920 Bachelor Thesis Automation</b>	Project Inter-galactic machine vision	Number of meeting this period 1). 1 meeting with counsellors	Client NTNU Ålesund	Page 2 of 2
<b>Progress report</b>	Period/week(s) 2-week period	Number of hours this period. (from log) Approx. 300	Group members Sondre Remøy, Simon Runde, Simon Ingebrigtsen, Einar Austnes	Date 11.05.22

- Guidance on writing report some time in week 20, if possible.

Approval/signature group leader Sondre Remøy <u>Sondre Remøy</u>	Signature other group participants Einar Austnes <u>Einar Austnes</u> Simon Runde <u>Simon Runde</u> Simon Ingebrigtsen <u>Simon Ingebrigtsen</u>
--	---

# **Appendix D**

## **Meeting minutes**

# CosmoAI – Møte med klienter/veiledere

## Møtereferat

02.03.2022, 12:30 – 14:00, A436

Til stede:	Prosjektgruppe:	Klienter/veiledere:
	Sondre Westbø Remøy	Hans Georg Schaathun
	Simon Nedreberg Runde	Ben David Normann
	Simon Ingebrigtsen	
	Einar Leite Austnes	

Neste møte: 16.03.2022, sted/tid TBD

## Oppsummering

Viste frem ny versjon av simulator med  $y$ -akse. Diskuterte noe av matematikken bak gaussian som gruppen bør se på. Veiledere nevnte at det lureste er å arbeide med utvikling og rapport parallelt, og også dokumentere simulatoren i rapporten slik at veiledere kan lese og se om matematikken stemmer.

Diskuterte om problemer med det nevrale nettet der det ikke ser forskjell på linseposisjon og einstein radius. Må utforskes om matrisene faktisk ser like ut eller om der er en forskjell.

Veileder tipset om forskjellig som gruppen bør lese om og teste; adversarial training, inception v3, U-net.

Diskutert mer rundt metoder og løsninger til problemet med maskinlæring.

# CosmoAI – Møte med klienter/veiledere

## Møtereferat

06.04.2022, 12:30 – 14:20, Digitalt

Til stede:	Prosjektgruppe:	Klienter/veiledere:
	Sondre Westbø Remøy	Hans Georg Schaathun
	Simon Nedreberg Runde	Ben David Normann
	Simon Ingebrigtsen	
	Einar Leite Austnes	

Neste møte: 20.03.2022

## Oppsummering

Kjørbart program ble sendt til veiledere i forkant. HG fikk ikke kjørt den.

Fikk noen tilbakemeldinger om hva som kan forbedres i GUI.

Gruppen burde skrive om «roteringstrikset» i rapporten og vise det matematisk.

Forslag til arbeidsmetoder for maskinlæring. Lablogg. Visualisere original og predicted.

Fikk også oppklaringer rundt spørsmål om matematikken bak sfærisk modell.

CosmoAI – Møte med klienter/veiledere

## Møtereferat

11.05.2022, 12:00 – 14:00, F420

Til stede:	Prosjektgruppe:	Klienter/veiledere:
	Sondre Westbø Remøy	Hans Georg Schaathun
	Simon Nedreberg Runde	Ben David Normann
	Simon Ingebrigtsen	
	Einar Leite Austnes	

Neste møte: Møter med veiledere neste uke

## Oppsummering

Gruppen sendte over et utkast av hele rapporten noen dager før møtet. Veiledere ga mye tilbakemelding og forslag til forbedringer gjennom hele møtet. Alle forslag og tips kan finnes på møtenotatene.

# CosmoAI – Møte med klienter/veiledere

## Møtereferat

12.01.2022, 9:00 – 10:30, Ankeret B434

Til stede:	Prosjektgruppe:	Klienter/veiledere:
	Sondre Westbø Remøy	Hans Georg Schaathun
	Simon Nedreberg Runde	Ben David Normann
	Simon Ingebrigtsen	
	Einar Leite Austnes	

Neste møte: 19.01.2021, 15:00, Rundskue

## Oppsummering

Hans Georg og Ben David gikk gjennom oppgaven stegvis slik gruppen fikk en oversikt over arbeidsoppgaver og kan planlegge prosjektet deretter.

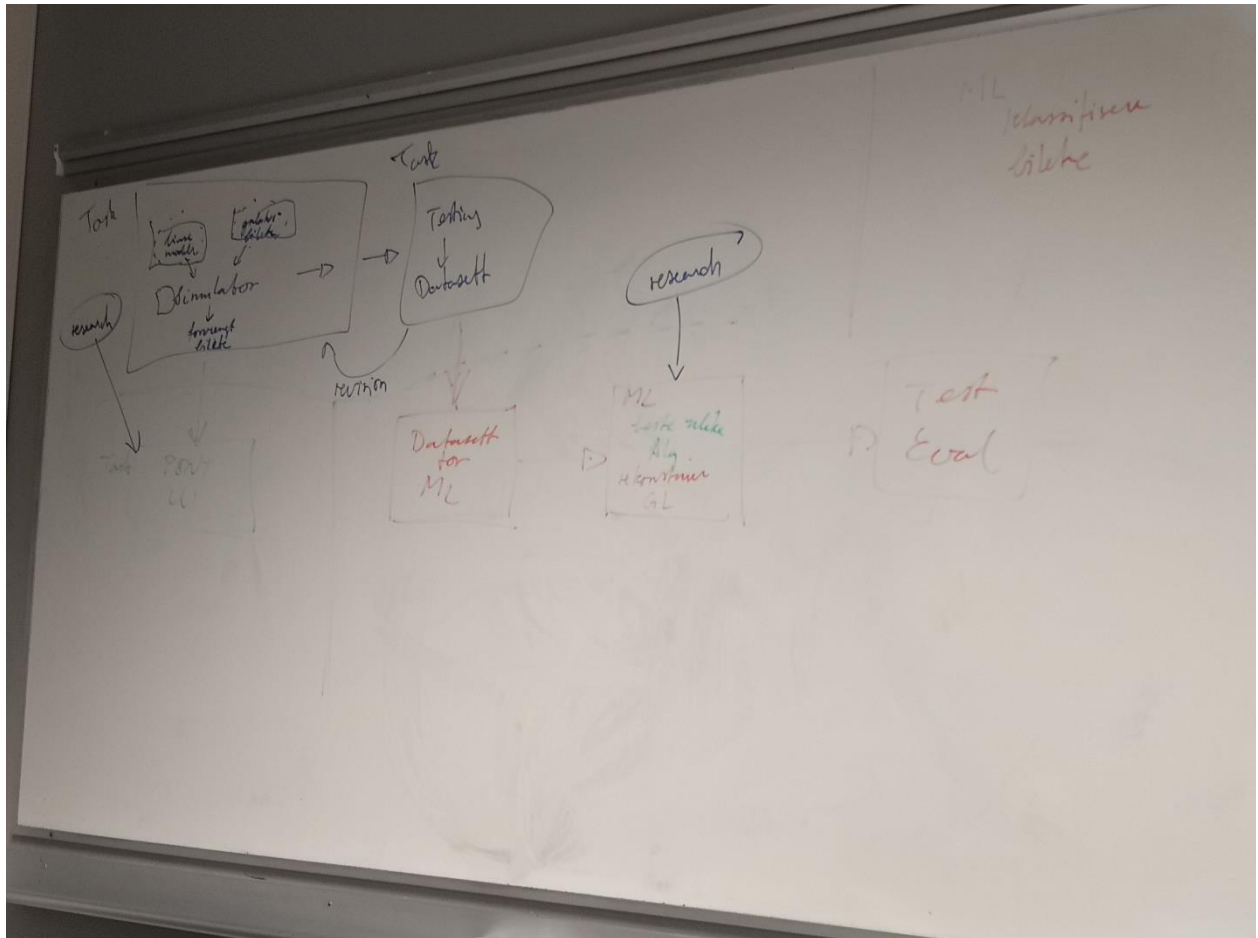
Klienter la frem ønsker om funksjoner til programvaren som skal utvikles.

- Ulike linsemodeller, BD skal utarbeide ellipsemmodell som kan brukes i simulering.
- HG ønsker CLI støtte i programvaren for å skape datasett.
- Brukervennlig GUI.

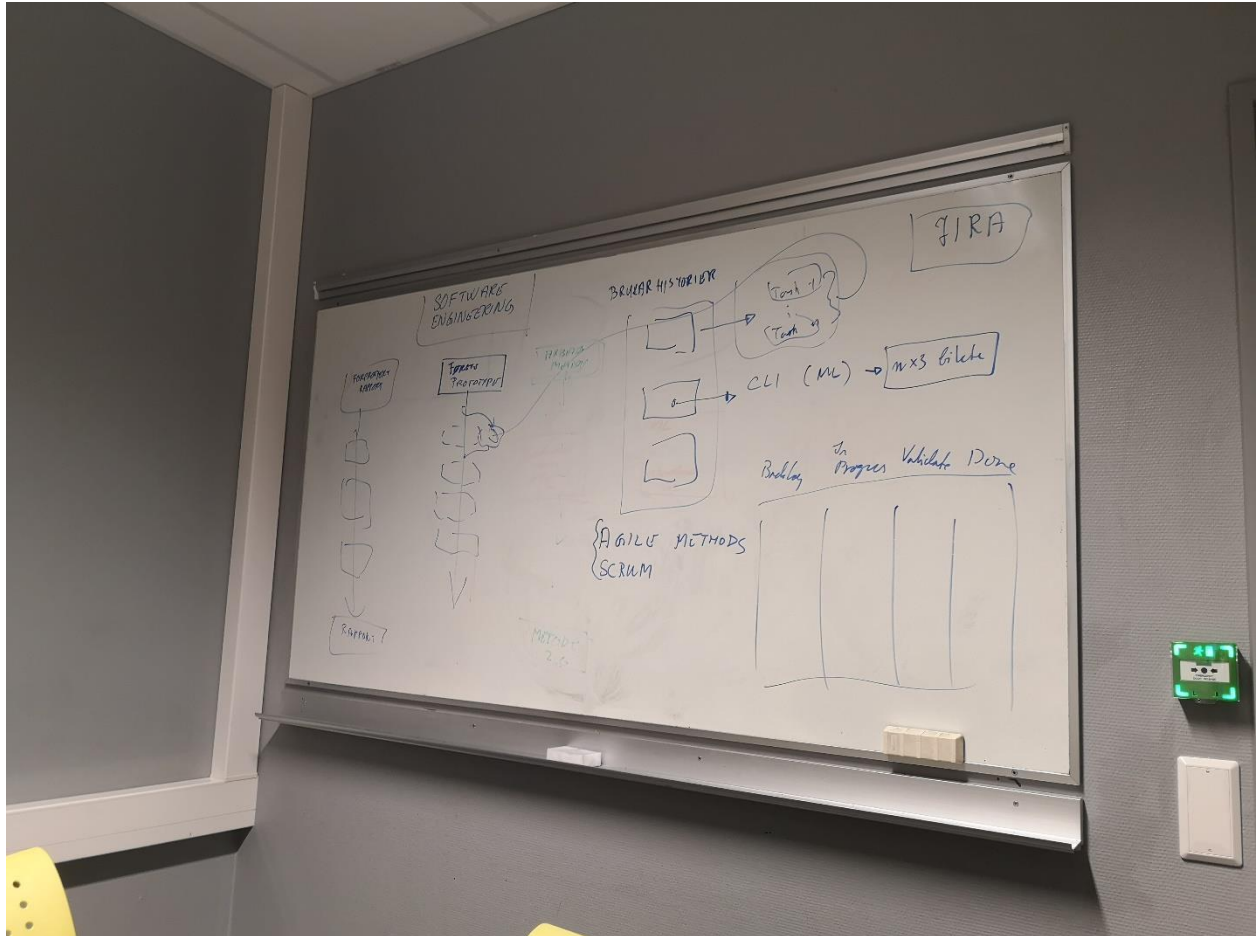
Hans Georg gikk gjennom ulike alternativer for prosjektmetodikk og samarbeidsmetoder for software-utvikling.



# Vedlegg 1 – Struktur, arbeidsoppdeling



## Vedlegg 2 – Metoder for software engineering



# CosmoAI – Møte med klienter/veiledere

## Møtereferat

19.01.2022, 15:00 – 16:10, Rundskue

Til stede:	Prosjektgruppe:	Klienter/veiledere:
	Sondre Westbø Remøy	Hans Georg Schaathun
	Simon Nedreberg Runde	Ben David Normann
	Simon Ingebrigtsen	
	Einar Leite Austnes	

Neste møte: 02.02.2022, sted/tid TBD

## Oppsummering

Veiledere ga tilbakemelding på første utkast av forprosjektrapporten og foreslo endringer og forbedringer. Foreslåtte endringer ligger tilgjengelig for gruppemedlemmene i møtenotatene på CosmoAI Discord.

Det ble diskutert om SCRUM/sprint modellen og hvordan den fungerer. Vi må forklare bedre i forprosjektrapporten hvordan vi erstatter tradisjonelle metoder og milepæler med dette.

Gruppen fikk mye info om hvordan Idun HPC kan benyttes til våre formål og hva som må gjøres før man setter i gang å bruke den. Det er viktig å fastsette at det som skal kjøres fungerer og at det ikke vil ta opp veldig mye prosessortid (for da er noe sannsynligvis galt).

Veiledere kom tidligere i uken med forslag om å skrive en artikkel for ECMS om simulatorimplementasjonen vi har så langt. Veiledere vil skrive det meste av problemskildringen og teori, mens gruppen må lage en pseudokode som er lett forståelig som forklarer hvordan simulatoren fungerer. Gruppen må også levere en demo av programvaren og screenshots av GUI og produserte bilder.

Det ble diskutert litt om presentasjonen neste uke. Der kan gruppen vise frem den tidlige simulatorprototypen og forklare hvordan ML kan hjelpe oss med å løse bacheloroppgaven.

# CosmoAI – Møte med klienter/veiledere

## Møtereferat

25.01.2022, 8:30 – 9:30, Digitalt

Til stede:

Prosjektgruppe:

Sondre Westbø Remøy

Simon Nedreberg Runde

Simon Ingebrigtsen

Einar Leite Austnes

Klienter/veiledere:

Ben David Normann

Neste møte:

## Oppsummering

Ekstra møte for veiledning og hjelp med matematikken bak simulatoren. Gruppen fikk oppklaring i noen spørsmål rundt variablene som inngår i den matematiske formelen.

# CosmoAI – Møte med klienter/veiledere

## Møtereferat

25.03.2022, 12:15 – 13:00, Digitalt

Til stede:	Prosjektgruppe:	Klienter/veiledere:
	Sondre Westbø Remøy	Hans Georg Schaathun
	Simon Nedreberg Runde	Ben David Normann
	Simon Ingebrigtsen	
	Einar Leite Austnes	

Neste møte: Om ca 2 uker

## Oppsummering

Utkast av rapport ble sendt i forkant.

Diskutert om rapporten og forbedringer om hva ulike kapitler skal inneholde. Også snakket om hvorvidt utledninger av matematikken er nødvendig i rapporten. Noen misplasserte kapitler av det som har blitt skrevet som skal fikses.

Generelle tips om rapportskrivningen fra HG.

BD har sendt prosjektbeskrivelse på e-post som kan hjelpe med rapportskrivningen.

CosmoAI – Møte med klienter/veiledere

## Møtereferat

27.04.2022, 12:00 – 13:30, Fogdegården

Til stede:	Prosjektgruppe:	Klienter/veiledere:
	Sondre Westbø Remøy	Hans Georg Schaathun
	Simon Nedreberg Runde	Ben David Normann
	Simon Ingebrigtsen	
	Einar Leite Austnes	

Neste møte: 11.05.2022

## Oppsummering

Begge veiledere har fått et kjørbart program som fungerer.

Mesteparten av møtet gikk til å diskutere den sfæriske modellen som er forsøkt implementert. Noe er galt med enten utledningen eller implementasjonen. Ben David skal dobbeltsjekke utregningene, og gruppen skal dobbeltsjekke programmet.

Veiledere ønsker en ring uten fyll som kilde i den sfæriske modellen.

Veiledere ønsker valg av antall ledd i GUI.

Mot slutten av møtet ble det diskutert om beste måten å bruke kildehenvisninger i rapporten, og om generell skrivestil og hva som er viktigst å fokusere på.

# CosmoAI – Gruppemøte

## Møtereferat

01.02.2022, 09:00 – 11:00, Digitalt

Til stede:                      Prosjektgruppe:  
   Sondre Westbø Remøy  
   Simon Nedreberg Runde  
   Simon Ingebrigtsen  
   Einar Leite Austnes

Neste møte:                      08.02.2021, TBD

## Oppsummering

Diskuserte fremdrift de siste 2 ukene og skrev fremdriftsrapport. Planla neste 2 ukene.

# CosmoAI – Gruppemøte

## Møtereferat

02.05.2022, 09:00 – 10:30, Digitalt

Til stede:                      Prosjektgruppe:  
   Sondre Westbø Remøy  
   Simon Nedreberg Runde  
   Simon Ingebrigtsen  
   Einar Leite Austnes

Neste møte:                      09.05.2022

## Oppsummering

Gruppen har skrevet en del i rapporten, simulator har nå finite terms, progresjon på SIS-modell. Diskutert progresjon med maskinlæring og planlagt videre arbeid.



# CosmoAI – Gruppemøte

## Møtereferat

04.04.2022, 09:00 – 11:00, Digitalt

Til stede:                      Prosjektgruppe:  
   Sondre Westbø Remøy  
   Simon Nedreberg Runde  
   Simon Ingebrigtsen  
   Einar Leite Austnes

Neste møte:                      11.04.2022

## Oppsummering

Siste uken har vært produktiv. Mye arbeid på ML og GUI. MacOS versjon av simulator kanskje mulig til onsdag.

Diskutert videre arbeidsoppgaver. Fortsette på AlexNet, trene mer på idun. GUI blir flyttet til ren qt, mangler fortsatt litt arbeid. Fortsette med rapporten.

# CosmoAI – Gruppemøte

## Møtereferat

08.02.2022, 13:00 – 14:00, Digitalt

Til stede:                      Prosjektgruppe:  
   Sondre Westbø Remøy  
   Simon Nedreberg Runde  
   Simon Ingebrigtsen  
   Einar Leite Austnes

Neste møte:                      15.02.2021, TBD

## Oppsummering

Diskutert fremgang. Problemer med KS og KL scaling. Progresjon på rotasjon.

Diskutert planer for GUI.

# CosmoAI – Gruppemøte

## Møtereferat

12.01.2022, 10:45 – 12:00, Ankeret B434

Til stede:                      Prosjektgruppe:  
                                      Sondre Westbø Remøy  
                                      Simon Nedreberg Runde  
                                      Simon Ingebrigtsen  
                                      Einar Leite Austnes

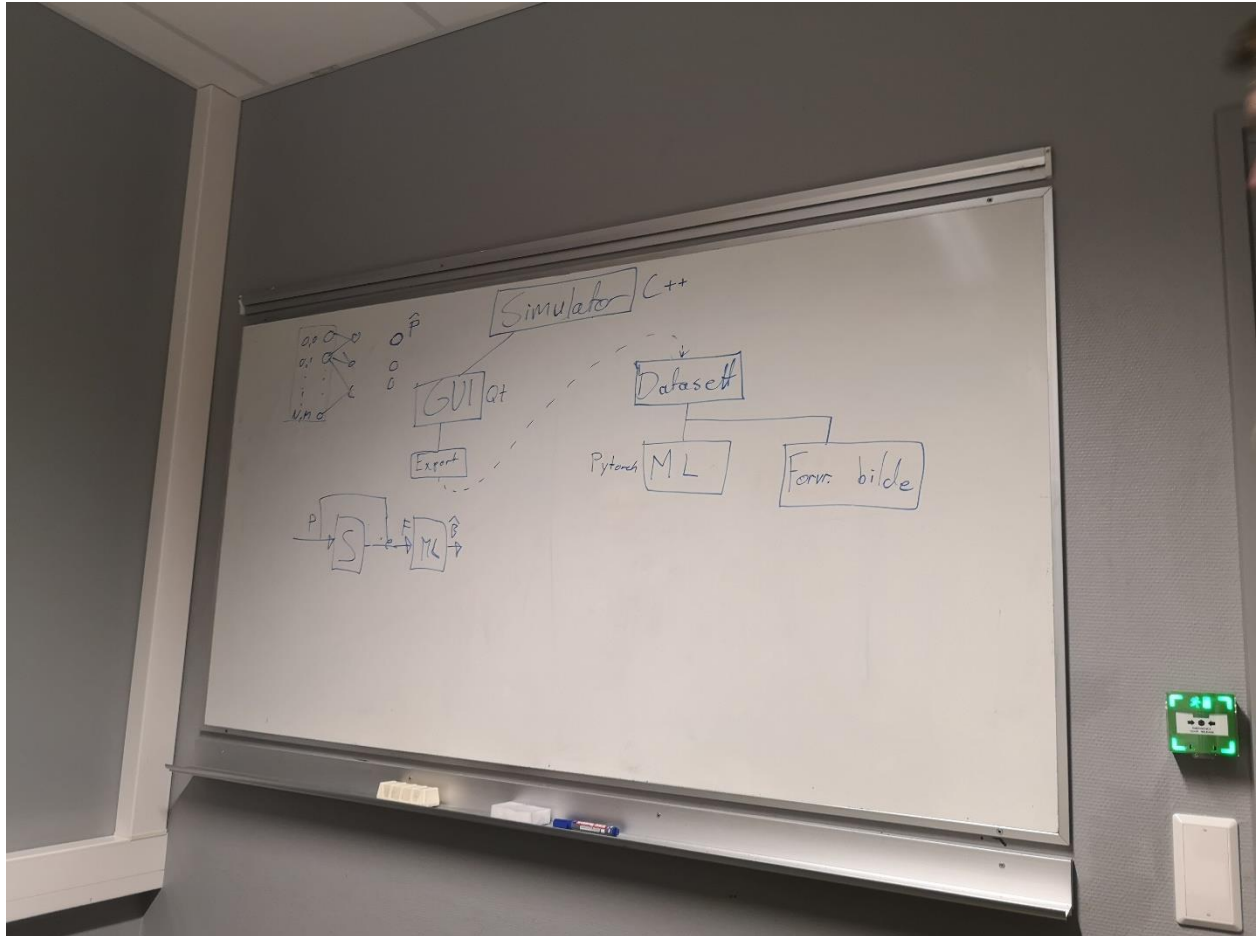
Neste møte:                    17.01.2021

## Oppsummering

Komt frem til en felles forståelse av problemstillingen.

Diskutert og tegnet diagrammer for oppbygning av programvaren. Diskusjon om å bruke C++ for simulatoren og oppretting av datasett, PyTorch i Python for maskinlæringsdelen og QT i C++ for GUI.

# Vedlegg 1 – Diagram programstruktur



# CosmoAI – Gruppemøte

## Møtereferat

17.01.2022, 9:00 – 10:00, Ankeret F423a

Til stede:                      Prosjektgruppe:  
                                      Sondre Westbø Remøy  
                                      Simon Nedreberg Runde  
                                      Simon Ingebrigtsen  
                                      Einar Leite Austnes

Neste møte:                    24.01.2021, 09:00, Ankeret F420

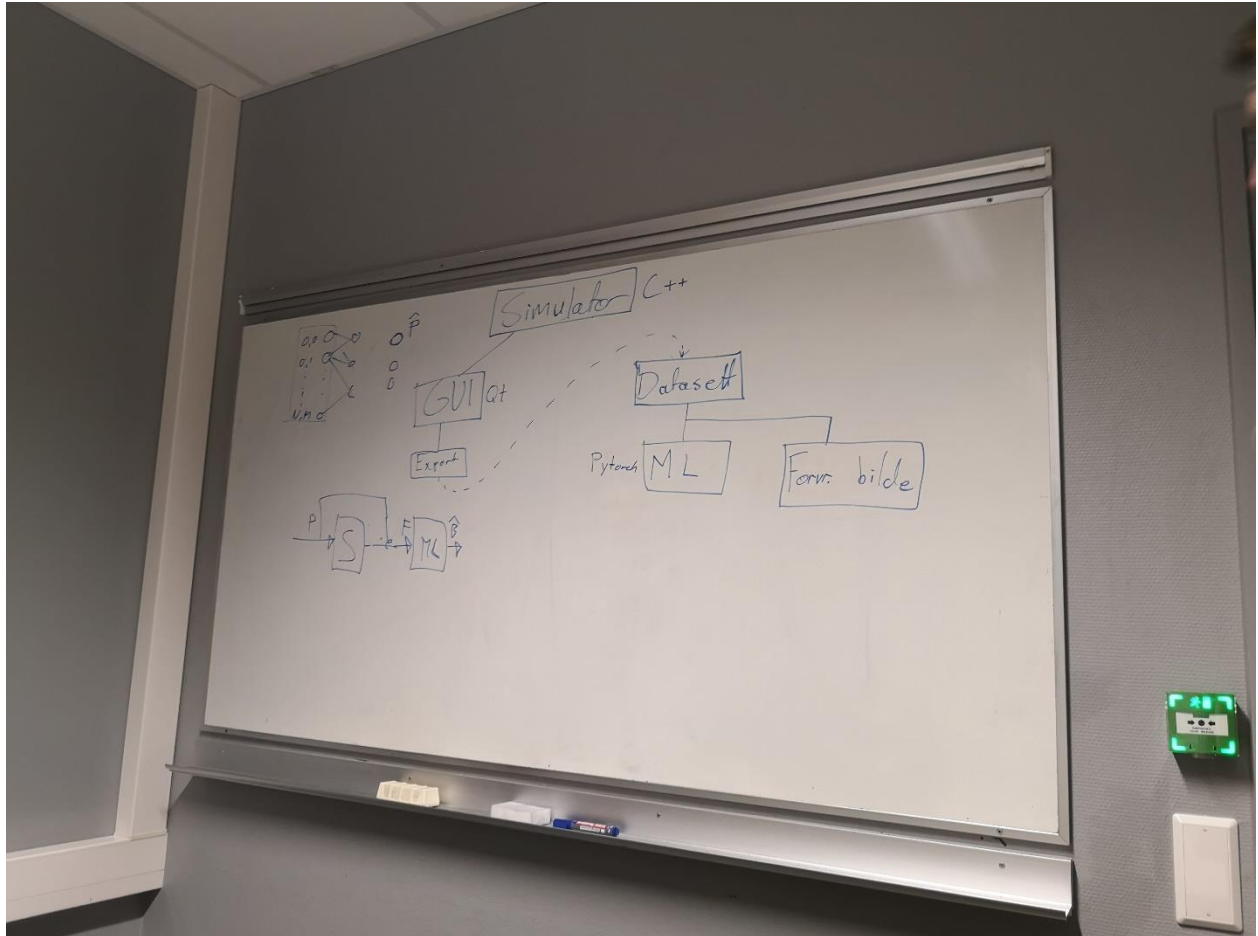
## Oppsummering

Diskutert og planlagt forprosjekt og arbeidsmetoder som skal benyttes i prosjektet.

- SCRUM smidig utvikling i stedet for GANTT.

Arbeid med forprosjekt etter møte.

# Vedlegg 1 – Diagram programstruktur



# CosmoAI – Gruppemøte

## Møtereferat

08.02.2022, 13:00 – 14:00, Digitalt

Til stede:                      Prosjektgruppe:  
                                      Sondre Westbø Remøy  
                                      Simon Nedreberg Runde  
                                      Simon Ingebrigtsen  
                                      Einar Leite Austnes

Neste møte:                    TBD

## Oppsummering

Lite fremgang grunnet parallelt fag. Diskutert implementering av mer avansert matematisk modell.  
Diskutert GUI og refaktorering av simulator i overgang til QT.

# CosmoAI – Gruppemøte

## Møtereferat

24.01.2022, 11:00 – 12:00, Ankeret F423a

Til stede:                      Prosjektgruppe:  
   Sondre Westbø Remøy  
   Simon Nedreberg Runde  
   Simon Ingebrigtsen  
   Einar Leite Austnes

Neste møte:                      01.02.2021, 09:00, Digitalt

## Oppsummering

Trenger hjelp med noe av matematikken som vi ikke forstår helt. Setter opp møte med Ben David så snart som råd.



# CosmoAI – Gruppemøte

## Møtereferat

25.04.2022, 09:00 – 11:00, Digitalt

Til stede:                      Prosjektgruppe:  
   Sondre Westbø Remøy  
   Simon Nedreberg Runde  
   Simon Ingebrigtsen  
   Einar Leite Austnes

Neste møte:                      02.05.2022

## Oppsummering

Diskutert videre arbeidsoppgaver og arbeid som har blitt gjort siste perioden. Det blir mye rapportskrivning fremover, også videre utvikling av ML og sfærisk modell samt fiksing av bugs i Qt simulator. Diskutert spørsmål til veiledere.

# CosmoAI – Gruppemøte

## Møtereferat

28.03.2022, 09:00 – 12:00, Digitalt

Til stede:                      Prosjektgruppe:  
   Sondre Westbø Remøy  
   Simon Nedreberg Runde  
   Simon Ingebrigtsen  
   Einar Leite Austnes

Neste møte:                      04.04.2022

## Oppsummering

Diskutert videre arbeidsoppgaver. Forske på nye nettverk for ML. Fortsette på GUI. Gjøre om fra opencv til qimage. Få til MacOS versjon.

# CosmoAI – Møte med klienter/veiledere

## Møtereferat

02.02.2022, 15:00 – 16:00, Digitalt

Til stede:	Prosjektgruppe:	Klienter/veiledere:
	Sondre Westbø Remøy	Hans Georg Schaathun
	Simon Nedreberg Runde	Ben David Normann
	Simon Ingebrigtsen	
	Einar Leite Austnes	

Neste møte: 16.02.2022, sted/tid TBD

## Oppsummering

Det ble diskutert i starten rundt forstørring i det forvrengte bildet i simulatoren som kommer på grunn av einstein radiusen. Kom frem til at dette gir mening.

Veiledere nevnte lenstronomy, som har gjort mye det samme som prosjektgruppen skal gjøre, bare med sterk formalisme i stedet for roulette. Gruppen burde utforske dette og se om det kan komme nye ideer ut ifra det andre har gjort.

Pseudokoden som ble skrevet kan bli forbedret. Den burde også legges i rapporten.

Diskusjon rundt oppgavens relevans i forhold til automasjon etter tilbakemeldinger fra presentasjonen. Kom frem til at det ikke er noe problem for gruppen. Oppgaven krever kunnskaper fra mange av fagene gruppen har hatt.

Fremdriftsrapporten var bra. Burde vært mer ambisiøse i punktet om rapportskrivning.

Mye diskusjon rundt den skalerende faktoren  $\chi_S/\chi_L$ . Gruppen skal skrive ned problemstillingen og diskutere den videre.

# **Appendix E**

## **Hour list**

### **E.1 Sondre**

## Worked hours - CosmoAI

Sondre Remøy

1.1.22

20.5.22

Total number of  
working hours

514,92

Date	Start time	End time	Working hours	Work done
10.1.22	11:42	12:46	1,07	First meeting with group after new years
11.1.22	15:00	22:00	7,00	Reading
12.1.22	09:00	13:00	4,00	Client meeting followed by group meeting
14.1.22	09:00	10:35	1,58	Writing pre-project report in meeting
14.1.22	11:15	12:00	0,75	Writing pre-project report
15.1.22	08:00	12:00	4,00	Writing pre-project report
17.1.22	09:30	15:30	6,00	Writing pre-project report
18.1.22	08:45	10:30	1,75	Writing pre-project report
18.1.22	10:00	15:30	5,50	Research
19.1.22	08:00	11:00	3,00	Writing pre-project report
19.1.22	14:00	16:00	2,00	Client meeting
22.1.22	12:00	15:00	3,00	Finishing pre-project report
22.1.22	15:00	21:00	6,00	Code and research
24.1.22	11:00	12:30	1,50	Daily group meeting
24.1.22	13:00	17:30	4,50	Code and research
25.1.22	08:30	09:30	1,00	Meeting with Ben David
25.1.22	10:00	19:30	9,50	Research, math and Qt
27.1.22	15:00	22:40	7,67	Research, math and Qt
28.1.22	12:00	14:30	2,50	Presentation/pitch of project
29.1.22	08:00	15:30	7,50	Research, reading papers
1.2.22	08:30	10:30	2,00	Progress report and group meeting
1.2.22	19:00	23:00	4,00	Research, gravitational lenses
2.2.22	15:00	16:15	1,25	Bi-weekly client meeting

10.1.22	11:42	12:46	1,07	First meeting with group after new years
3.2.22	16:00	23:00	7,00	Research, Qt and GUI
4.2.22	08:30	15:30	7,00	Research, Qt and GUI
5.2.22	09:00	17:00	8,00	Code and research
6.2.22	12:00	18:30	6,50	Research, Qt and GUI
6.2.22	20:30	21:00	0,50	Planning
8.2.22	08:00	11:00	3,00	Writing on final report
8.2.22	11:00	13:00	2,00	Research on GL
8.2.22	13:00	14:06	1,10	Informal group meeting/discussion
9.2.22	10:00	11:20	1,33	GL-lecture from BD and meeting
10.2.22	16:00	22:45	6,75	GUI
11.2.22	09:00	17:00	8,00	ML
12.2.22	09:00	17:00	8,00	ML
13.2.22	15:00	22:00	7,00	GUI
14.2.22	08:00	14:00	6,00	GUI
14.2.22	16:00	22:00	6,00	GUI
15.2.22	09:00	16:30	7,50	ML
16.2.22	13:00	13:30	0,50	Planning
18.2.22	08:00	12:00	4,00	Writing on final report
21.2.22	08:30	15:30	7,00	GUI
22.2.22	08:00	16:00	8,00	GUI
23.2.22	13:30	14:00	0,50	Planning
23.2.22	10:00	12:40	2,67	GL-lecture from BD and meeting
1.3.22	10:30	11:30	1,00	Planning
1.3.22	11:30	12:30	1,00	Progress report and group meeting
23.3.22	13:30	15:00	1,50	Planning
24.3.22	18:00	22:00	4,00	Writing progress report and final report
28.3.22	08:00	12:00	4,00	Group meeting and working on ML
29.3.22	08:00	15:30	7,50	Reasearch and testing, ML
30.3.22	08:00	15:00	7,00	Reasearch and testing, ML
1.4.22	16:00	23:00	7,00	Reasearch and testing, ML
4.4.22	09:00	11:30	2,50	Group meeting
4.4.22	12:00	19:30	7,50	GUI

10.1.22	11:42	12:46	1,07	First meeting with group after new years
5.4.22	08:00	13:00	5,00	GUI
5.4.22	13:00	14:30	1,50	Planning
5.4.22	16:30	23:00	6,50	Group meeting, writing progress report and working on GUI
6.4.22	08:00	12:00	4,00	GUI
6.4.22	12:30	14:30	2,00	Bi-weekly client meeting
7.4.22	15:00	20:00	5,00	Research
8.4.22	08:30	17:00	8,50	Writing on final report
11.4.22	08:00	17:00	9,00	Research
12.4.22	09:00	18:00	9,00	Writing on final report
13.4.22	16:00	20:00	4,00	Research
14.4.22	16:00	20:00	4,00	Research
15.4.22	08:00	10:00	2,00	Research
16.4.22	18:00	21:00	3,00	Research
17.4.22	18:00	23:00	5,00	Research
18.4.22	08:00	14:00	6,00	Research
19.4.22	08:00	16:00	8,00	GUI
20.4.22	08:00	09:00	1,00	Planning
20.4.22	09:00	14:30	5,50	GUI
21.4.22	16:00	18:00	2,00	Writing on final report
22.4.22	10:00	15:00	5,00	Writing on final report
25.4.22	08:00	09:00	1,00	Planning
25.4.22	09:00	11:00	2,00	Group meeting
25.4.22	11:00	16:00	5,00	Writing on final report
26.4.22	08:30	16:00	7,50	Writing on final report
26.4.22	18:00	20:00	2,00	Writing progress report
27.4.22	08:30	12:00	3,50	Writing on final report
27.4.22	12:00	13:00	1,00	Bi-weekly client meeting
27.4.22	13:00	16:00	3,00	Writing on final report
28.4.22	16:00	21:00	5,00	GUI
29.4.22	08:00	15:30	7,50	GUI and final report
30.4.22	10:00	17:00	7,00	Research
1.5.22	12:00	18:00	6,00	Writing on final report

10.1.22	11:42	12:46	1,07	First meeting with group after new years
2.5.22	08:30	09:00	0,50	Planning
2.5.22	09:00	10:30	1,50	Group meeting
2.5.22	10:30	16:30	6,00	Writing on final report
3.5.22	08:30	16:30	8,00	Writing on final report
4.5.22	08:30	15:30	7,00	Writing on final report
5.5.22	16:00	22:00	6,00	Writing on final report
6.5.22	08:00	17:00	9,00	Writing on final report
7.5.22	18:00	22:00	4,00	Writing on final report
8.5.22	17:00	22:00	5,00	Writing on final report
9.5.22	08:00	15:30	7,50	Writing on final report
10.5.22	08:00	15:30	7,50	Writing on final report
11.5.22	08:00	11:30	3,50	Writing on final report
11.5.22	12:00	14:00	2,00	Bi-weekly client meeting
12.5.22	14:00	15:30	1,50	Writing on final report
13.5.22	08:00	16:00	8,00	Writing on final report
14.5.22	10:00	18:00	8,00	Writing on final report
15.5.22	09:00	21:00	12,00	Writing on final report, and making poster
16.5.22	08:00	16:00	8,00	Writing on final report, and making poster
17.5.22	18:00	22:00	4,00	Writing script for video
18.5.22	08:00	17:00	9,00	Writing on final report and filming video
19.5.22	09:00	21:00	12,00	Writing on final report



## **E.2 Einar**

Navn: Einar  
Bachelor oppgave

Sum: 473

Dato	Timer
10.01.2022	2
11.01.2022	5
12.01.2022	4
13.01.2022	2
14.01.2022	5
15.01.2022	2
16.01.2022	
17.01.2022	6
18.01.2022	7
19.01.2022	8
20.01.2022	4
21.01.2022	6
22.01.2022	
23.01.2022	
24.01.2022	6
25.01.2022	4
26.01.2022	5
27.01.2022	4
28.01.2022	4
29.01.2022	
30.01.2022	
31.01.2022	
01.02.2022	4
02.02.2022	8
03.02.2022	6
04.02.2022	4
05.02.2022	
06.02.2022	
07.02.2022	
08.02.2022	
09.02.2022	
10.02.2022	5
11.02.2022	8
12.02.2022	
13.02.2022	
14.02.2022	
15.02.2022	
16.02.2022	
17.02.2022	4
18.02.2022	6
19.02.2022	
20.02.2022	
21.02.2022	
22.02.2022	
23.02.2022	3
24.02.2022	6

25.02.2022	8
26.02.2022	
27.02.2022	
28.02.2022	
01.03.2022	4
02.03.2022	
03.03.2022	
04.03.2022	
05.03.2022	
06.03.2022	
07.03.2022	
08.03.2022	
09.03.2022	
10.03.2022	4
11.03.2022	7
12.03.2022	
13.03.2022	
14.03.2022	
15.03.2022	
16.03.2022	
17.03.2022	7
18.03.2022	7
19.03.2022	
20.03.2022	
21.03.2022	
22.03.2022	
23.03.2022	8
24.03.2022	4
25.03.2022	8
26.03.2022	
27.03.2022	
28.03.2022	6
29.03.2022	6
30.03.2022	4
31.03.2022	4
01.04.2022	6
02.04.2022	
03.04.2022	
04.04.2022	3
05.04.2022	5
06.04.2022	5
07.04.2022	6
08.04.2022	5
09.04.2022	
10.04.2022	
11.04.2022	7
12.04.2022	7
13.04.2022	7
14.04.2022	
15.04.2022	

16.04.2022	
17.04.2022	
18.04.2022	6
19.04.2022	6
20.04.2022	5
21.04.2022	7
22.04.2022	3
23.04.2022	
24.04.2022	4
25.04.2022	7
26.04.2022	6
27.04.2022	5
28.04.2022	5
29.04.2022	4
30.04.2022	4
01.05.2022	
02.05.2022	8
03.05.2022	8
04.05.2022	8
05.05.2022	7
06.05.2022	8
07.05.2022	
08.05.2022	8
09.05.2022	9
10.05.2022	8
11.05.2022	8
12.05.2022	10
13.05.2022	10
14.05.2022	9
15.05.2022	10
16.05.2022	12
17.05.2022	
18.05.2022	12
19.05.2022	12
20.05.2022	8

### **E.3 Simon I.**

Navn: Simon I

Sum: 530

Bachelor oppgave

Dato	Timer
1.10.2022	4
1.11.2022	6
1.12.2022	
1.13.2022	8
1.14.2022	6
1.15.2022	3
1.16.2022	
1.17.2022	8
1.18.2022	7
1.19.2022	6
1.20.2022	4
1.21.2022	8
1.22.2022	
1.23.2022	
1.24.2022	7
1.25.2022	5
1.26.2022	7
1.27.2022	6
1.28.2022	4
1.29.2022	
1.30.2022	
1.31.2022	
2.1.2022	5
2.2.2022	6
2.3.2022	7
2.4.2022	5
2.5.2022	
2.6.2022	
2.7.2022	
2.8.2022	
2.9.2022	
2.10.2022	10
2.11.2022	8
2.12.2022	
2.13.2022	
2.14.2022	
2.15.2022	
2.16.2022	
2.17.2022	5
2.18.2022	7
2.19.2022	
2.20.2022	
2.21.2022	
2.22.2022	
2.23.2022	5
2.24.2022	5

2.25.2022	5
2.26.2022	
2.27.2022	
2.28.2022	
3.1.2022	5
3.2.2022	
3.3.2022	
3.4.2022	
3.5.2022	
3.6.2022	
3.7.2022	
3.8.2022	
3.9.2022	
3.10.2022	10
3.11.2022	8
3.12.2022	
3.13.2022	
3.14.2022	
3.15.2022	
3.16.2022	
3.17.2022	8
3.18.2022	7
3.19.2022	
3.20.2022	
3.21.2022	
3.22.2022	
3.23.2022	8
3.24.2022	6
3.25.2022	8
3.26.2022	
3.27.2022	
3.28.2022	8
3.29.2022	7
3.30.2022	4
3.31.2022	5
4.1.2022	8
4.2.2022	
4.3.2022	
4.4.2022	5
4.5.2022	5
4.6.2022	6
4.7.2022	6
4.8.2022	8
4.9.2022	
4.10.2022	
4.11.2022	8
4.12.2022	8
4.13.2022	8
4.14.2022	
4.15.2022	

4.16.2022	
4.17.2022	
4.18.2022	6
4.19.2022	7
4.20.2022	
4.21.2022	8
4.22.2022	6
4.23.2022	
4.24.2022	8
4.25.2022	7
4.26.2022	5
4.27.2022	5
4.28.2022	5
4.29.2022	7
4.30.2022	7
5.1.2022	
5.2.2022	8
5.3.2022	8
5.4.2022	9
5.5.2022	8
5.6.2022	8
5.7.2022	
5.8.2022	8
5.9.2022	9
5.10.2022	8
5.11.2022	8
5.12.2022	10
5.13.2022	10
5.14.2022	9
5.15.2022	10
5.16.2022	12
5.17.2022	
5.18.2022	12
5.19.2022	12
5.20.2022	7



**E.4 Simon R.**

Navn: Simon

Sum: 542

Bachelor oppgave

Dato	Timer
10.01.2022	4
11.01.2022	5
12.01.2022	
13.01.2022	6
14.01.2022	7
15.01.2022	4
16.01.2022	2
17.01.2022	7
18.01.2022	7
19.01.2022	5
20.01.2022	3
21.01.2022	7
22.01.2022	3
23.01.2022	
24.01.2022	7
25.01.2022	6
26.01.2022	8
27.01.2022	7
28.01.2022	4
29.01.2022	2
30.01.2022	
31.01.2022	
01.02.2022	6
02.02.2022	6
03.02.2022	8
04.02.2022	6
05.02.2022	1
06.02.2022	
07.02.2022	
08.02.2022	
09.02.2022	4
10.02.2022	9
11.02.2022	8
12.02.2022	
13.02.2022	
14.02.2022	
15.02.2022	3
16.02.2022	
17.02.2022	4
18.02.2022	8
19.02.2022	
20.02.2022	
21.02.2022	
22.02.2022	
23.02.2022	7
24.02.2022	4

25.02.2022	8
26.02.2022	
27.02.2022	1
28.02.2022	
01.03.2022	6
02.03.2022	
03.03.2022	
04.03.2022	
05.03.2022	
06.03.2022	
07.03.2022	
08.03.2022	
09.03.2022	
10.03.2022	10
11.03.2022	9
12.03.2022	
13.03.2022	
14.03.2022	
15.03.2022	
16.03.2022	
17.03.2022	7
18.03.2022	7
19.03.2022	
20.03.2022	
21.03.2022	
22.03.2022	
23.03.2022	9
24.03.2022	7
25.03.2022	6
26.03.2022	
27.03.2022	
28.03.2022	9
29.03.2022	7
30.03.2022	5
31.03.2022	4
01.04.2022	7
02.04.2022	
03.04.2022	
04.04.2022	6
05.04.2022	5
06.04.2022	7
07.04.2022	4
08.04.2022	8
09.04.2022	
10.04.2022	
11.04.2022	9
12.04.2022	7
13.04.2022	8
14.04.2022	
15.04.2022	

16.04.2022	
17.04.2022	
18.04.2022	5
19.04.2022	7
20.04.2022	
21.04.2022	7
22.04.2022	8
23.04.2022	
24.04.2022	8
25.04.2022	8
26.04.2022	5
27.04.2022	6
28.04.2022	5
29.04.2022	7
30.04.2022	8
01.05.2022	
02.05.2022	8
03.05.2022	7
04.05.2022	9
05.05.2022	7
06.05.2022	8
07.05.2022	
08.05.2022	7
09.05.2022	9
10.05.2022	7
11.05.2022	8
12.05.2022	9
13.05.2022	10
14.05.2022	9
15.05.2022	9
16.05.2022	10
17.05.2022	
18.05.2022	12
19.05.2022	14
20.05.2022	1

# Appendix F

## Source code

### F.1 Simulator

#### CMakeLists.txt

---

```
cmake_minimum_required(VERSION 3.15)
project(CosmoAi CXX)

# Download automatically, you can also just copy the conan.cmake file
if(NOT EXISTS "${CMAKE_BINARY_DIR}/conan.cmake")
    message(STATUS "Downloading conan.cmake from https://github.com/conan-
io/conan-cmake")
    file(DOWNLOAD "https://raw.githubusercontent.com/conan-io/conan-
master/conan.cmake"
        "${CMAKE_BINARY_DIR}/conan.cmake")
endif()

include(${CMAKE_BINARY_DIR}/conan.cmake)

conan_cmake_run(REQUIRES
    symengine/0.9.0
    opencv/4.5.3
    BASIC_SETUP)

add_executable(Datagen Data_generator.cpp)
add_executable(Simulator_2 GL_Simulator_2.cpp)
add_library(SimLib Simulator.cpp)
```

```
target_link_libraries(Datagen ${CONAN_LIBS} SimLib)
target_link_libraries(Simulator_2 ${CONAN_LIBS} SimLib)
target_link_libraries(SimLib ${CONAN_LIBS})

set(CMAKE_EXE_LINKER_FLAGS "${CMAKE_EXE_LINKER_FLAGS} -Wl,-STACK:10000000")
```

---

**Simulator.h**


---

```

#ifndef SPHERICAL_SIMULATOR_H
#define SPHERICAL_SIMULATOR_H

#include "opencv2/opencv.hpp"
#include <symengine/expression.h>
#include <symengine/lambda_double.h>

using namespace SymEngine;

class Simulator {
private:
    int mode;
    double GAMMA;
    double CHI;
    double actualX {};
    double actualY {};
    double apparentX {};
    double apparentY {};
    double apparentX2 {};
    double apparentY2 {};
    double X {};
    double Y {};
    double phi {};
    double actualAbs {};
    double apparentAbs {};
    double apparentAbs2 {};
    double R {};
    int n;

    cv::Mat imgDistorted;

    std::array<std::array<LambdaRealDoubleVisitor, 52>, 51> alphas_l;
    std::array<std::array<LambdaRealDoubleVisitor, 52>, 51> betas_l;
    std::array<std::array<double, 52>, 51> alphas_val;
    std::array<std::array<double, 52>, 51> betas_val;

public:
    int size;
    std::string name;
    int CHI_percent;
    int sourceSize;

```

```

    int einsteinR;
    int xPosSlider;
    int yPosSlider;

public:
    Simulator();

    void update();

    void initGui();

    void writeToPngFiles(int);

private:
    void calculate();

//    void drawSource(cv::Mat& img, double xPos, double yPos) const;

    [[nodiscard]] std::pair<double, double> pointMass(double r, double
        theta) const;

    static void update_dummy(int, void*);

    cv::Mat formatImg(cv::Mat &imgDistorted, cv::Mat &imgActual, int
        displaySize) const;

    static void refLines(cv::Mat &target);

    void distort(int row, int col, const cv::Mat &src, cv::Mat &dst);

    void parallelDistort(const cv::Mat &src, cv::Mat &dst);

    void initAlphasBetas();

    std::pair<double, double> spherical(double r, double theta) const;

    void drawParallel(cv::Mat &img, int xPos, int yPos);

    void drawSource(int begin, int end, cv::Mat &img, int xPos, int yPos);
};

#endif //SPHERICAL_SIMULATOR_H

```





**Simulator.cpp**


---

```

#include "Simulator.h"
#include <symengine/expression.h>
#include <symengine/lambda_double.h>
#include <thread>
#include <symengine/parser.h>
#include <fstream>

#define PI 3.14159265358979323846

double factorial_(unsigned int n);

Simulator::Simulator() :
    size(300),
    CHI_percent(50),
    CHI(CHI_percent/100.0),
    einsteinR(size/20),
    sourceSize(size/20),
    xPosSlider(size/2 + 1),
    yPosSlider(size/2),
    mode(0), // 0 = point mass, 1 = sphere
    n(10)
{
    GAMMA = einsteinR/2.0;
}

void Simulator::update() {
    auto startTime = std::chrono::system_clock::now();

    GAMMA = einsteinR/2.0;
    calculate();
    cv::Mat imgActual(size, size, CV_8UC1, cv::Scalar(0, 0, 0));
    // cv::circle(imgActual, cv::Point(size/2 + (int)actualX, size/2 - (int)actualY), sourceSize, cv::Scalar::all(255), 2*sourceSize);
    drawParallel(imgActual, actualX, actualY);
    cv::Mat imgApparent;

    // if point
    if (mode == 0){
        imgApparent = cv::Mat(size, 2*size, CV_8UC1, cv::Scalar(0, 0, 0));
    }
}

```

```

        imgDistorted = cv::Mat(imgApparent.size(), CV_8UC1, cv::Scalar(0,
            0, 0));
//      cv::circle(imgApparent, cv::Point(size + (int)apparentAbs, size
/2), sourceSize, cv::Scalar::all(255), 2*sourceSize);
drawParallel(imgApparent, apparentAbs, 0);
parallelDistort(imgApparent, imgDistorted);
// rotate image
cv::Mat rot = cv::getRotationMatrix2D(cv::Point(size, size/2), phi
    *180/PI, 1);
cv::warpAffine(imgDistorted, imgDistorted, rot, cv::Size(2*size,
    size)); // crop distorted image
imgDistorted = imgDistorted(cv::Rect(size/2, 0, size, size));
}

// if Spherical
else if (mode == 1){

    // calculate all amplitudes for given X, Y, GAMMA, CHI
    for (int m = 1; m <= n; m++){
        for (int s = (m+1)%2; s <= (m+1); s+=2){
            alphas_val[m][s] = alphas_l[m][s].call({X, Y, GAMMA, CHI});
            ;
            betas_val[m][s] = betas_l[m][s].call({X, Y, GAMMA, CHI});
        }
    }

    imgApparent = cv::Mat(2*size, 2*size, CV_8UC1, cv::Scalar(0, 0, 0)
        );
    imgDistorted = cv::Mat(size, size, CV_8UC1, cv::Scalar(0, 0, 0));
//      cv::circle(imgApparent, cv::Point(size + (int)apparentX, size -
(int)apparentY), sourceSize, cv::Scalar::all(255), 2*sourceSize);
drawParallel(imgApparent, apparentX, apparentY);
//      cv::imshow("apparent", imgApparent);
parallelDistort(imgApparent, imgDistorted);
}

//      cv::cvtColor(imgDistorted, imgDistorted, cv::COLOR_GRAY2BGR);

//      const int displaySize = 500;
//      refLines(imgActual);
//      refLines(imgDistorted);
//      cv::Mat matDst = formatImg(imgDistorted, imgActual, displaySize);
cv::Mat matDst(cv::Size(2*size, size), imgActual.type(), cv::Scalar::
    all(255));
cv::Mat matRoi = matDst(cv::Rect(0, 0, size, size));

```

```

imgActual.copyTo(matRoi);
matRoi = matDst(cv::Rect(size, 0, size, size));
imgDistorted.copyTo(matRoi);

cv::imshow("GL_Simulator", matDst);

auto endTime = std::chrono::system_clock::now();
//    std::cout << "Time: " << std::chrono::duration_cast<std::chrono::
milliseconds>(endTime - startTime).count() << " milliseconds" << std::
endl;

}

void Simulator::parallelDistort(const cv::Mat& src, cv::Mat& dst) {
    int n_threads = std::thread::hardware_concurrency();
    std::vector<std::thread> threads_vec;
    for (int i = 0; i < n_threads; i++) {
        int begin = dst.rows/n_threads*i;
        int end = dst.rows/n_threads*(i+1);
        std::thread t([begin, end, src, &dst, this]() { distort(begin,
            end, src, dst); });
        threads_vec.push_back(std::move(t));
//        distort(row, col, src, dst);
    }
//    if(mode){
//        std::cout << 100.0 * row / dst.rows << "%" << std::endl;
//    }

    for (auto& thread : threads_vec) {
        thread.join();
    }
}

void Simulator::distort(int begin, int end, const cv::Mat& src, cv::Mat&
dst) {
    for (int row = begin; row < end; row++) {
        for (int col = 0; col < dst.cols; col++) {
            // if point mass
            int row_, col_;

            if (!mode) {
                // Set coordinate system with origin at x=R
                double x = (col - apparentAbs - dst.cols / 2.0) * CHI;

```

```

        double y = (dst.rows / 2.0 - row) * CHI;
        // Calculate distance and angle of the point evaluated
           relative to center of lens (origin)
        double r = sqrt(x * x + y * y);
        double theta = atan2(y, x);
        auto pos = pointMass(r, theta);
        // Translate to array index
        row_ = (int) round(src.rows / 2.0 - pos.second);
        col_ = (int) round(apparentAbs + src.cols / 2.0 + pos.
           first);
    }
    // if sphere
else {

        double x = col - dst.cols / 2.0 - X;
        double y = dst.rows / 2.0 - row - Y;
        double r = sqrt(x * x + y * y);

        double theta = atan2(y, x);
        auto pos = spherical(r, theta);
        // Translate to array index
        col_ = (int) round(apparentX + src.cols / 2.0 + pos.first)
           ;
        row_ = (int) round(src.rows / 2.0 - pos.second - apparentY
           );
    }

    // If (x', y') within source, copy value to imgDistorted
    if (row_ < src.rows && col_ < src.cols && row_ >= 0 && col_ >=
        0) {
        auto val = src.at<uchar>(row_, col_);
        dst.at<uchar>(row, col) = val;
    }
}
}

//for (int m = 1; m <= n; m++){
//for (int s = (m+1)%2; s <= (m+1); s+=2){

std::pair<double, double> Simulator::spherical(double r, double theta)
const {
    double ksi1 = 0;
    double ksi2 = 0;

```

```

for (int m=1; m<=n; m++){
    double frac = pow(r, m) / factorial_(m);
    double subTerm1 = 0;
    double subTerm2 = 0;
    for (int s = (m+1)%2; s <= (m+1); s+=2){
        double alpha = alphas_val[m][s];
        double beta = betas_val[m][s];
        int c_p = 1 + s/(m + 1);
        int c_m = 1 - s/(m + 1);
        subTerm1 += 1.0/4*((alpha*cos((s-1)*theta) + beta*sin((s-1)*
            theta))*c_p + (alpha*cos((s+1)*theta) + beta*sin((s+1)*
            theta))*c_m);
        subTerm2 += 1.0/4*((-alpha*sin((s-1)*theta) + beta*cos((s-1)*
            theta))*c_p + (alpha*sin((s+1)*theta) - beta*cos((s+1)*
            theta))*c_m);
    }
    double term1 = frac*subTerm1;
    double term2 = frac*subTerm2;
    ksi1 += term1;
    ksi2 += term2;
    // Break summation if term is less than 1/100 of ksi or if ksi is
    // well outside frame
    // if ( (std::abs(term1) < std::abs(ksi1)/1000) && (std::abs(term2)
    // < std::abs(ksi2)/1000) || (ksi1 < -1000*size || ksi1 > 1000*size ||
    // ksi2 < -1000*size || ksi2 > 1000*size) ){
    //     break;
    // }
    return {ksi1, ksi2};
}

```

```

std::pair<double, double> Simulator::pointMass(double r, double theta)
const {
    double frac = (einsteinR * einsteinR * r) / (r * r + R * R + 2 * r * R
        * cos(theta));
    double x_ = (r*cos(theta) + frac * (r / R + cos(theta))) / CHI;
    double y_ = (r*sin(theta) - frac * sin(theta)) / CHI; // Point mass lens
        equation
    return {x_, y_};
}

```

```

void Simulator::drawParallel(cv::Mat& dst, int xPos, int yPos){
    int n_threads = std::thread::hardware_concurrency();
}

```

```

std::vector<std::thread> threads_vec;
for (int i = 0; i < n_threads; i++) {
    int begin = dst.rows / n_threads * i;
    int end = dst.rows / n_threads * (i + 1);
    std::thread t([begin, end, &dst, xPos, yPos, this]() { drawSource(
        begin, end, dst, xPos, yPos); });
    threads_vec.push_back(std::move(t));
}
for (auto& thread : threads_vec) {
    thread.join();
}
}

```

```

void Simulator::drawSource(int begin, int end, cv::Mat& dst, int xPos, int
yPos) {
    for (int row = begin; row < end; row++) {
        for (int col = 0; col < dst.cols; col++) {
            int x = col - xPos - dst.cols/2;
            int y = row + yPos - dst.rows/2;
            auto value = (uchar)round(255 * exp((-x * x - y * y) / (2.0*
                sourceSize*sourceSize)));
            dst.at<uchar>(row, col) = value;
        }
    }
}

```

```

void Simulator::writeToPngFiles(int n_params) {
    std::ostringstream filename_path;
    std::ostringstream filename;

    if (n_params == 5) {
        filename << CHI_percent << ", ";
    }
    filename << einsteinR << ", " << sourceSize << ", " << xPosSlider << ", "
        << yPosSlider << ".png";
    filename_path << name + "/images/" + filename.str();
    cv::imwrite(filename_path.str(), imgDistorted);
    // cv::imshow(filename_path.str(), image);
    // cv::waitKey(0);
}

```

```

double factorial_(unsigned int n){

```

```

    double a = 1.0;
    for (int i = 2; i <= n; i++){
        a *= i;
    }
    return a;
}

void Simulator::calculate() {

    CHI = CHI_percent/100.0;

    // actual position in source plane
    actualX = xPosSlider - size / 2.0;
    actualY = yPosSlider - size / 2.0;

    // Absolute values in source plane
    actualAbs = sqrt(actualX * actualX + actualY * actualY);
    double ratio1 = 0.5 + sqrt(0.25 + einsteinR*einsteinR/(CHI*CHI*
        actualAbs*actualAbs));
    double ratio2 = 0.5 - sqrt(0.25 + einsteinR*einsteinR/(CHI*CHI*
        actualAbs*actualAbs));
    apparentAbs = actualAbs*ratio1;
    apparentAbs2 = actualAbs*ratio2;
    apparentX = actualX*ratio1;
    apparentX2 = actualX*ratio2;
    apparentY = actualY*ratio1;
    apparentY2 = actualY*ratio2;

    // Projection of apparent position in lens plane
    R = apparentAbs * CHI;
    X = apparentX * CHI;
    Y = apparentY * CHI;

    // Angle relative to x-axis
    phi = atan2(actualY, actualX);
}

//cv::Mat Simulator::formatImg(cv::Mat &imgDistorted, cv::Mat &imgActual,
    int displaySize) const {
//     if (!mode){
//         cv::circle(imgDistorted, cv::Point(size / 2, size / 2), (int)
//             round(einsteinR / CHI), cv::Scalar::all(60));

```



```

//      cv::drawMarker(imgDistorted, cv::Point(size / 2 + (int)
    apparentX, size / 2 - (int) apparentY), cv::Scalar(0, 0, 255), cv::
    MARKER_TILTED_CROSS, size / 30);
//      cv::drawMarker(imgDistorted, cv::Point(size / 2 + (int)
    apparentX2, size / 2 - (int) apparentY2), cv::Scalar(0, 0, 255), cv::
    MARKER_TILTED_CROSS, size / 30);
//      cv::drawMarker(imgDistorted, cv::Point(size / 2 + (int) actualX,
    size / 2 - (int) actualY), cv::Scalar(255, 0, 0), cv::
    MARKER_TILTED_CROSS, size / 30);
//  }
//  cv::resize(imgActual, imgActual, cv::Size(displaySize, displaySize))
;
//  cv::resize(imgDistorted, imgDistorted, cv::Size(displaySize,
    displaySize));
//  cv::Mat matDst(cv::Size(2*displaySize, displaySize), imgActual.type
    (), cv::Scalar::all(255));
//  cv::Mat matRoi = matDst(cv::Rect(0, 0, displaySize, displaySize));
//  imgActual.copyTo(matRoi);
//  matRoi = matDst(cv::Rect(displaySize, 0, displaySize, displaySize));
//  imgDistorted.copyTo(matRoi);
//  return matDst;
//}

```

```

//// Add some lines to the image for reference
//void Simulator::refLines(cv::Mat& target) {
//  int size_ = target.rows;
//  for (int i = 0; i < size_; i++) {
//      target.at<cv::Vec3b>(i, size_ / 2) = {60, 60, 60};
//      target.at<cv::Vec3b>(size_ / 2 - 1, i) = {60, 60, 60};
//      target.at<cv::Vec3b>(i, size_ - 1) = {255, 255, 255};
//      target.at<cv::Vec3b>(i, 0) = {255, 255, 255};
//      target.at<cv::Vec3b>(size_ - 1, i) = {255, 255, 255};
//      target.at<cv::Vec3b>(0, i) = {255, 255, 255};
//  }
//}

```

```

void Simulator::initGui() {
    initAlphasBetas();
    // Make the user interface and specify the function to be called when
    moving the sliders: update()
    cv::namedWindow("GL_Simulator", cv::WINDOW_AUTOSIZE);
    cv::createTrackbar("Lens_dist_%%_%%", "GL_Simulator", &CHI_percent,
        100, update_dummy, this);
}

```



```
    auto alpha_sym = SymEngine::parse(alpha);
    auto beta_sym = SymEngine::parse(beta);
    SymEngine::LambdaRealDoubleVisitor alpha_num, beta_num;
    alphas_l[std::stoi(m)][std::stoi(s)].init({x, y, g, c}, *
        alpha_sym);
    betas_l[std::stoi(m)][std::stoi(s)].init({x, y, g, c}, *
        beta_sym);
    }
}
}
```

---

**GL\_Simulator\_2.cpp**

---

```
#include <opencv2/opencv.hpp>
#include "Simulator.h"
#include <fstream>

int main()
{
    Simulator simulator;
    try{
        simulator.initGui();
    }
    catch (std::exception &e){
        std::cout << "initGui_returned_exception:" << e.what() << std::
            endl;
        return 1;
    }

    bool running = true;
    while (running) {
        int k = cv::waitKey(1);
        if ((cv::getWindowProperty("GL_Simulator", cv::WND_PROP_AUTOSIZE)
            == -1) || (k == 27)) {
            running = false;
        }
        if (k == 32) {
            simulator.update();
        }
    }
    cv::destroyAllWindows();
    return 0;
}
```

---

**Data\_Generator.cpp**

---

```
#include <thread>
#include <random>
#include <string>
#include <opencv2/opencv.hpp>
#include "Simulator.h"

int main(int, char *argv[]) {

    Simulator simulator;

    int DATAPOINTS_TO_GENERATE = atoi(argv[1]);
    simulator.size = atoi(argv[2]);
    simulator.name = std::string(argv[3]);
    int n_params = atoi(argv[4]);

    // Generate dataset:
    std::random_device dev;
    std::mt19937 rng(dev());
    std::uniform_int_distribution<std::mt19937::result_type>
        rand_lens_dist(30, 100);
    std::uniform_int_distribution<std::mt19937::result_type>
        rand_einsteinR(1, simulator.size/10);
    std::uniform_int_distribution<std::mt19937::result_type>
        rand_source_size(1, simulator.size/10);
    std::uniform_int_distribution<std::mt19937::result_type> rand_xSlider
        (100, simulator.size-100);
    std::uniform_int_distribution<std::mt19937::result_type> rand_ySlider
        (100, simulator.size-100);

    std::vector<std::vector<int>> parameters;
    for (int i = 0; i < DATAPOINTS_TO_GENERATE; i++) {
        if (n_params == 4) {
            simulator.CHI_percent = 50;
        }
        else {
            simulator.CHI_percent = rand_lens_dist(rng);
        }
        simulator.einsteinR = rand_einsteinR(rng);
        simulator.sourceSize = rand_source_size(rng);
        simulator.xPosSlider = rand_xSlider(rng);
        simulator.yPosSlider = rand_ySlider(rng);
    }
}
```

```
std::vector<int> params = {simulator.CHI_percent, simulator.
    einsteinR, simulator.sourceSize, simulator.xPosSlider,
    simulator.yPosSlider };

if ( (!std::count(parameters.begin(), parameters.end(), params)) )
    { // check for duplicate
      simulator.update();
      simulator.writeToPngFiles(n_params);
      parameters.push_back({ simulator.CHI_percent, simulator.
          einsteinR, simulator.sourceSize, simulator.xPosSlider,
          simulator.yPosSlider });
    }
else{
    i--;
}
if (parameters.size() % (DATAPOINTS_TO_GENERATE/10) == 0){
    std::cout << "_Datapoints_generated:_ " << parameters.size() <<
        std::endl;
}
}
}
```

---

## F.2 GUI

### mainwindow.h

---

```
#ifndef MAINWINDOW_H
#define MAINWINDOW_H

#include <QMainWindow>
#include <QMessageBox>

QT_BEGIN_NAMESPACE
namespace Ui { class MainWindow; }
QT_END_NAMESPACE

class MainWindow : public QMainWindow
{
    Q_OBJECT

private:
    Ui::MainWindow *ui;
    bool grid;
    bool markers;
    bool legendCheck;
    bool darkMode = true;
    int gridSize;
    int wSize = 600;
    int wSizeWide;
    int einsteinR;
    int srcSize;
    int actualX;
    int actualY;
    double phi;
    double CHI;
    double actualAbs;
    double apparentAbs;
    double apparentX;
    double apparentY;
    double apparentAbs2;
    double R;
    int apparentX2;
    int apparentY2;
    QImage imgApparent;
    QImage imgActual;
    QImage imgDistorted;
```

```

QPixmap pixApp;
QPixmap pixAct;
QPixmap pixDist;
QPixmap imgAppPix;
QPixmap imgActPix;
QPixmap imgDistPix;
QPixmap rocket;
QPixmap legend;
QString source;
QString lensType;
QMessageBox msgBox;
int terms = 0;
std::string mode = "finite";

```

**public:**

```

MainWindow(QWidget *parent = nullptr);
~MainWindow();

```

**private:**

```

void init_values();
void drawGrid(QPixmap &img);
void drawRadius(QPixmap& src, double);
void drawMarker(QPixmap &src, int x, int y, int size, QColor color);
void setup();
void updateImg();
void drawGaussianThreaded(QImage&, double, double);
void drawGaussian(int, int, QImage&, double, double);
void distort(int, int);
void distortThreaded();
QPixmap rotate(QPixmap src, double angle, int x, int y);
void resizeEvent(QResizeEvent *event);
void drawLegend(QPixmap&, int refSize);
void drawText(QPixmap& img, int x, int y, int fontSize, QString text);
void theme();
void drawSource();
void saveImage();
std::pair<double, double> pointMass(double r, double theta);
std::pair<double, double> spherical(double r, double theta) const;
std::pair<double, double> pointMassFinite(double r, double theta);

void calculateStuff();

```

**private slots:**

```

void on_einsteinSpinbox_valueChanged();
void on_srcSizeSpinbox_valueChanged();
void on_lensDistSpinbox_valueChanged(int);

```



```
void on_xSpinBox_valueChanged ();
void on_ySpinBox_valueChanged ();
void on_gridBox_stateChanged(int arg1);
void on_markerBox_stateChanged(int arg1);
void on_resetButton_clicked ();
void on_srcTypeComboBox_currentTextChanged(const QString &arg1);
void on_actionReset_triggered ();
void on_actionMarkers_toggled(bool arg1);
void on_actionLegend_toggled(bool arg1);
void on_actionOff_triggered ();
void on_action2x2_triggered ();
void on_action4x4_triggered ();
void on_action8x8_triggered ();
void on_action12x12_triggered ();
void on_actionChange_resolution_triggered ();
void on_actionCustom_triggered ();
void on_actionDark_mode_toggled(bool arg1);
void on_saveButton_clicked ();
void on_actionSave_image_as_triggered ();
void on_infTermsCheckbox_toggled(bool checked);
void on_termsSpinBox_valueChanged(int arg1);
void on_actionAbout_triggered ();
void on_actionValues_triggered ();
};

#endif // MAINWINDOW_H
```

---

**main.cpp**

---

```
#include "mainwindow.h"
```

```
#include <QApplication>
```

```
int main(int argc, char *argv[])
```

```
{
```

```
    QApplication a(argc, argv);
```

```
    MainWindow w;
```

```
    w.setWindowIcon(QIcon(":/icons/icons/CosmoAI.png"));
```

```
    w.show();
```

```
    return a.exec();
```

```
}
```

---

**mainwindow.cpp**

---

```
#include "mainwindow.h"
#include "../ui_mainwindow.h"
#include <string>
#include <iostream>
#include <QtCore>
#include <QString>
#include <QPainter>
#include <QDebug>
#include <QPainterPath>
#include <QInputDialog>
#include <QStyleFactory>
#include <QFileDialog>
#define PI 3.14159265358979323846

void MainWindow::updateImg() {
    if (actualX == 0){
        actualX = 1; // a cheap trick
    }
    // Reset images
    imgApparent.fill(Qt::black);
    imgActual.fill(Qt::black);
    imgDistorted.fill(Qt::black);

    calculateStuff();

    drawSource();

    // Convert image to pixmap
    imgAppPix = QPixmap::fromImage(imgApparent);
    // Pre rotate pixmap
    imgAppPix = rotate(imgAppPix, phi, 0, 0);

    // Convert pre-rotated pixmap to image and distort the image
    imgApparent = imgAppPix.toImage();
    distortThreaded();

    // Convert distorted image to pixmap, rotate and crop
    imgDistPix = QPixmap::fromImage(imgDistorted);
    imgDistPix = rotate(imgDistPix, -phi, 0, 0);
    QRect rect(wSize/2, wSize/2, wSize, wSize);
    imgDistPix = imgDistPix.copy(rect);
```

```

imgActPix = QPixmap::fromImage(imgActual);

// Scale pixmaps to fit in labels
int labelH = ui->actLabel->height();
imgDistPix = imgDistPix.scaled(labelH, labelH, Qt::KeepAspectRatio, Qt::SmoothTransformation);
imgActPix = imgActPix.scaled(labelH, labelH, Qt::KeepAspectRatio, Qt::SmoothTransformation);

double ratio = (double)labelH/wSize;
// Draw grids and markers
if (grid) {
    drawGrid(imgActPix);
//    drawGrid(imgAppPixDisp);
    drawGrid(imgDistPix);
    drawRadius(imgDistPix, ratio);
}
if (markers) {
    drawMarker(imgDistPix, ratio*(wSize/2 + apparentX), ratio*(wSize/2 - apparentY), 10, Qt::blue);
    drawMarker(imgDistPix, ratio*(wSize/2 + apparentX2), ratio*(wSize/2 - apparentY2), 10, Qt::blue);
    drawMarker(imgDistPix, ratio*(wSize/2 + actualX), ratio*(wSize/2 - actualY), 10, Qt::red);
}

// Draw legend after scaling to ensure text clarity
if (legendCheck && markers) {
    drawLegend(imgDistPix, ui->distLabel->height());
}

// Set values of the value message box every loop
QString msg = QStringLiteral("Apparent_1_X:_%1_\n"
                             "Apparent_1_Y:_%2_\n"
                             "Apparent_2_X:_%3_\n"
                             "Apparent_2_Y:_%4_\n"
                             "").arg(apparentX).arg(apparentY).arg(apparentX2).arg(apparentY2);

msgBox.setText(msg);

// Draw pixmaps on QLabels
ui->actLabel->setPixmap(imgActPix);
ui->distLabel->setPixmap(imgDistPix);
}

```

```

// Split the image into (number of threads available) pieces and distort
// the pieces in parallel
void MainWindow::distortThreaded() {
    unsigned int num_threads = std::thread::hardware_concurrency();
    int tasks = imgDistorted.height() / num_threads;
    int remainder = imgDistorted.height() % num_threads;

    std::vector<std::thread> threads_vec;
    for (unsigned int k = 0; k < num_threads; k++) {
        unsigned int thread_begin = tasks * k;
        unsigned int thread_end = tasks * (k + 1);

        if (k == num_threads - 1 && remainder != 0) {
            thread_end = thread_end + remainder;
        }

        std::thread t(&MainWindow::distort, this, thread_begin, thread_end);
        threads_vec.push_back(std::move(t));
    }
    for (auto& thread : threads_vec) {
        thread.join();
    }
}

void MainWindow::distort(int begin, int end) {
    int rows = imgDistorted.height();
    int cols = imgDistorted.width();
    // Evaluate each point in imgDistorted plane ~ lens plane
    for (int row = begin; row < end; row++) {
        for (int col = 0; col <= cols; col++) { // <=
            ?????????????????????????????????????????????????????????

            double x = (col - apparentAbs - cols/2.0) * CHI;
            double y = (rows/2.0 - row) * CHI;

            // Calculate distance and angle of the point evaluated
            // relative to center of lens (origin)
            double r = sqrt(x*x + y*y);
            double theta = atan2(y, x);

            std::pair<double, double> pos;

```

```

        if (mode == "infinite"){
            pos = pointMass(r, theta);
        }
        else if (mode == "finite"){
            pos = pointMassFinite(r, theta);
        }

        // Translate to array index
        int row_ = (int)round(rows/2.0 - pos.second);
        int col_ = (int)round(apparentAbs + cols/2.0 + pos.first);

        // If (x', y') within source, copy value to imgDistorted
        if (row_ < rows && col_ < cols && row_ > 0 && col_ >= 0) {
            imgDistorted.setPixel(col, row, imgApparent.pixel(col_,
                row_));
        }
    }
}

std::pair<double, double> MainWindow::pointMass(double r, double theta){
    // Point mass lens equation
    double frac = (einsteinR * einsteinR * r) / (r * r + R * R + 2 * r * R
        * cos(theta));
    double x_ = (r*cos(theta) + frac * (r / R + cos(theta))) / CHI;
    double y_ = (r*sin(theta) - frac * sin(theta)) / CHI;
    return {x_, y_};
}

std::pair<double, double> MainWindow::pointMassFinite(double r, double
theta){
    double xTemp = 0;
    double yTemp = 0;
    for(int m = 1; m <= terms; m++){
        int sign = 1 - 2*(m%2);
        double frac = std::pow((r/R), m);
        xTemp += sign * frac * std::cos(m*theta);
        yTemp -= sign * frac * std::sin(m*theta);
    }
    double x_ = (r * std::cos(theta) - einsteinR*einsteinR/R*xTemp)/CHI;
    double y_ = (r * std::sin(theta) - einsteinR*einsteinR/R*yTemp)/CHI;
    return {x_, y_};
}

```

```

QPixmap MainWindow::rotate(QPixmap src, double angle, int x, int y) {
    QPixmap r(src.size());
    //    QSize s = src.size();
    r.fill(QColor::fromRgb(Qt::black));
    QPainter m(&r);
    m.setRenderHint(QPainter::SmoothPixmapTransform);
    m.translate(src.width()/2 + x, src.height()/2 + y);
    m.rotate(angle*180/PI);
    m.translate(-src.width()/2 - x, -src.height()/2 - y);
    m.drawPixmap(0,0, src);
    return r;
}

void MainWindow::drawSource() {
    if (source == "Gauss") {
        drawGaussianThreaded(imgActual, actualX, actualY);
    //    std::cout << "actualX: " << actualX << " actualY: " << actualY
    << " apparentX: " << apparentX << " apparentY: " << apparentY << std::
    endl;
        drawGaussianThreaded(imgApparent, apparentX, apparentY);
    }
    else {
        QPainter pAct(&imgActual);
        QPainter pApp(&imgApparent);
        QPen pen(Qt::white, wSize/200);
        pAct.setPen(pen);
        pApp.setPen(pen);

        if (source == "Rocket") {
            QPixmap rocket1 = rocket.scaled(6*srcSize, 6*srcSize, Qt::
            KeepAspectRatio, Qt::SmoothTransformation);
            QPoint posApp(apparentX + imgApparent.width()/2 - rocket1.
            width()/2, -apparentY + imgApparent.height()/2 - rocket1.
            height()/2);
            QPoint posAct(actualX + imgActual.width()/2 - rocket1.width()
            /2, imgActual.height()/2 - actualY - rocket1.height()/2);
            pAct.drawPixmap(posAct, rocket1);
            pApp.drawPixmap(posApp, rocket1);
        }
        QRect rectApp(apparentX + imgApparent.width()/2 - srcSize, -
        apparentY + imgApparent.height()/2 - srcSize, 2*srcSize, 2*
        srcSize);
        QRect rectAct(actualX + imgActual.width()/2 - srcSize, imgActual.
        height()/2 - actualY - srcSize, 2*srcSize, 2*srcSize);
    }
}

```

```

    if (source == "Circle") {
        pAct.drawEllipse(rectAct);
        pApp.drawEllipse(rectApp);
    }
    else if (source == "Square") {
        pAct.drawRect(rectAct);
        pApp.drawRect(rectApp);
    }

//     else if (source == "Triangle") {
//         QPainterPath pathAct;
//         pathAct.moveTo(rectAct.left() + (rectAct.width() / 2),
// rectAct.top());
//         pathAct.lineTo(rectAct.bottomLeft());
//         pathAct.lineTo(rectAct.bottomRight());
//         pathAct.lineTo(rectAct.left() + (rectAct.width() / 2),
// rectAct.top());
//         pAct.fillPath(pathAct, QBrush(Qt::white));

//         QPainterPath pathApp;
//         pathApp.moveTo(rectApp.left() + (rectApp.width() / 2),
// rectApp.top());
//         pathApp.lineTo(rectApp.bottomLeft());
//         pathApp.lineTo(rectApp.bottomRight());
//         pathApp.lineTo(rectApp.left() + (rectApp.width() / 2),
// rectApp.top());
//         pApp.fillPath(pathApp, QBrush(Qt::white));
//     }
}

void MainWindow::drawGaussianThreaded(QImage& img, double xPos, double
yPos) {
    unsigned int num_threads = std::thread::hardware_concurrency();
    int tasks = img.height() / num_threads;
    int remainder = img.height() % num_threads;

    std::vector<std::thread> threads_vec;
    for (unsigned int k = 0; k < num_threads; k++) {
        unsigned int thread_begin = tasks * k;
        unsigned int thread_end = tasks * (k + 1);

        if (k == num_threads - 1 && remainder != 0) {
            thread_end = thread_end + remainder;
        }
    }
}

```



```

        std::thread t(&MainWindow::drawGaussian, this, thread_begin,
                    thread_end, std::ref(img), xPos, yPos);
        threads_vec.push_back(std::move(t));
    }
    for (auto& thread : threads_vec) {
        thread.join();
    }
}

void MainWindow::drawGaussian(int begin, int end, QImage& img, double xPos
, double yPos) {
    int rows = img.height();
    int cols = img.width();
    for (int row = begin; row < end; row++) {
        for (int col = 0; col < cols; col++) {
            double x = col - xPos - cols/2;
            double y = -yPos - row + rows/2;
            int val = round(255 * exp((-x * x - y * y) / (2.0*srcSize*
                srcSize)));
            img.setPixel(col, row, qRgb(val, val, val));
        }
    }
}

void MainWindow::drawRadius(QPixmap& src, double ratio){
    QPointF center(src.width()/2, src.height()/2);
    QPainter painter(&src);
    QPen pen(Qt::gray, 2, Qt::DashLine);
    painter.setPen(pen);
    painter.setOpacity(0.3);
    painter.drawEllipse(center, (int)round(einsteinR/CHI*ratio), (int)
        round(einsteinR/CHI*ratio));
}

void MainWindow::drawGrid(QPixmap& img) {
    QPainter painter(&img);
    QPen pen(Qt::gray, 2, Qt::SolidLine);
    painter.setPen(pen);
    painter.setOpacity(0.3);

    if (gridSize > 0) {
        int remainder = (img.height()%gridSize)/2;
        for (int var = img.height()/gridSize; var < img.height()*1.5;) {

```

```

        QLineF lineVert (img.height()-var-remainder, 0, img.height()-
            var-remainder, img.height()-remainder);
        QLineF lineHor(0, img.height()-var-remainder, img.height()-
            remainder, img.height()-var-remainder);
        painter.drawLine(lineVert);
        painter.drawLine(lineHor);
        var+=img.height()/gridSize;
    }
}

void MainWindow::drawMarker(QPixmap& src, int x, int y, int size, QColor
color) {
    QPointF point(x, y);
    QPainter painter(&src);
    QPen pen(color, size);
    painter.setPen(pen);
    painter.setOpacity(0.4);
    painter.drawPoint(point);
}

void MainWindow::resizeEvent(QResizeEvent *event) {
    QMainWindow::resizeEvent(event);
    updateImg();
}

void MainWindow::drawText(QPixmap& img, int x, int y, int fontSize,
    QString text) {
    QPointF point(x,y);
    QPainter painter(&img);
    QFont font;
    font.setPixelSize(fontSize);
    painter.setFont(font);
    painter.drawText(point, text);
}

void MainWindow::drawLegend(QPixmap& img, int refSize) {

    // Create legend pixmap
    int legendHeight = refSize/14;
    int legendWidth = refSize/4;
    QPixmap legend(legendWidth, legendHeight);

    // Background color of legend
    legend.fill(Qt::gray);

```

```

    // Draw markers in legend
    int markerSize = legendWidth/16;
    drawMarker(legend, legendWidth/10, legendHeight/2 - markerSize,
        markerSize, Qt::red);
    drawMarker(legend, legendWidth/10, legendHeight/2 + markerSize,
        markerSize, Qt::blue);

    int fontSize = legendWidth/11.5;
    drawText(legend, legendWidth/15 + markerSize*2, legendHeight/2 -
        markerSize + markerSize/2, fontSize, "Actual_position");
    drawText(legend, legendWidth/15 + markerSize*2, legendHeight/2 +
        markerSize + markerSize/2, fontSize, "Apparent_positions");

    // Set legend opacity and draw to main pixmap
    QPainter painter(&img);
    int xPos = 0;
    if (apparentX < -wSize/8 && apparentY > wSize/8) {
        xPos += refSize - legendWidth;
    }
    painter.setOpacity(0.6);
    painter.drawPixmap(xPos, 0, legend);
}

void MainWindow::theme() {
    if (darkMode) {
        qApp->setStyle(QStyleFactory::create("Fusion"));
        QPalette darkPalette;
        darkPalette.setColor(QPalette::Window, QColor(53,53,53));
        darkPalette.setColor(QPalette::WindowText, Qt::white);
        darkPalette.setColor(QPalette::Disabled, QPalette::WindowText,
            QColor(127,127,127));
        darkPalette.setColor(QPalette::Base, QColor(42,42,42));
        darkPalette.setColor(QPalette::AlternateBase, QColor(66,66,66));
        darkPalette.setColor(QPalette::ToolTipBase, Qt::white);
        darkPalette.setColor(QPalette::ToolTipText, Qt::white);
        darkPalette.setColor(QPalette::Text, Qt::white);
        darkPalette.setColor(QPalette::Disabled, QPalette::Text, QColor
            (127,127,127));
        darkPalette.setColor(QPalette::Dark, QColor(35,35,35));
        darkPalette.setColor(QPalette::Shadow, QColor(20,20,20));
        darkPalette.setColor(QPalette::Button, QColor(53,53,53));
        darkPalette.setColor(QPalette::ButtonText, Qt::white);
        darkPalette.setColor(QPalette::Disabled, QPalette::ButtonText,
            QColor(127,127,127));
    }
}

```

```

    darkPalette.setColor(QPalette::BrightText, Qt::red);
    darkPalette.setColor(QPalette::Link, QColor(42,130,218));
    darkPalette.setColor(QPalette::Highlight, QColor(42,130,218));
    darkPalette.setColor(QPalette::Disabled, QPalette::Highlight, QColor
        (80,80,80));
    darkPalette.setColor(QPalette::HighlightedText, Qt::white);
    darkPalette.setColor(QPalette::Disabled, QPalette::HighlightedText,
        QColor(127,127,127));
    darkPalette.setColor(QPalette::ToolTipBase, QColor(42,42,42));
    qApp->setPalette(darkPalette);

} else {
    qApp->setPalette(this->style()->standardPalette());
}
}

void MainWindow::saveImage() {
    if (markers && legendCheck) {
        drawLegend(imgDistPix, wSize);
    }

    QString defaultFileName = QDate::currentDate().toString("'cosmoai_'
        yyyy-MM-dd");
    defaultFileName.append(QTime::currentTime().toString("-hh-mm-ss '.png'"
        ));

    QImage imageDist = imgDistPix.toImage();
    QImage imageAct = imgActPix.toImage();

    QImage combined(2*imageDist.width(), imageDist.height(), QImage::
        Format_RGB32);
    QPainter painter(&combined);
    painter.drawImage(0, 0, imageAct);
    painter.drawImage(imageDist.width(), 0, imageDist);
    painter.end();

    QString fileName = QFileDialog::getSaveFileName(this, tr("Save_Image")
        , QString(defaultFileName), tr("PNG_(*.png)"));

    if (!fileName.isEmpty())
    {
        combined.save(fileName);
    }
}

```

```

}

void MainWindow::calculateStuff () {
    // Calculate positions and angles
    actualAbs = sqrt(actualX*actualX + actualY*actualY);
    double ratio1 = 0.5 + sqrt(0.25 + einsteinR*einsteinR/(CHI*CHI*
        actualAbs*actualAbs));
    double ratio2 = 0.5 - sqrt(0.25 + einsteinR*einsteinR/(CHI*CHI*
        actualAbs*actualAbs));
    apparentAbs = actualAbs*ratio1;
    apparentAbs2 = actualAbs*ratio2;
    apparentX = actualX * ratio1;
    apparentY = actualY * ratio1;
    apparentX2 = actualX * ratio2;
    apparentY2 = actualY * ratio2;
    R = apparentAbs * CHI;
    phi = atan2(actualY, actualX);

    // Spherical
    // X = apparentX * CHI;
    // Y = apparentY * CHI;
}

MainWindow::MainWindow(QWidget *parent)
    : QMainWindow(parent)
    , ui(new Ui::MainWindow)
{
    ui->setupUi(this);
    this->setWindowTitle("CosmoAI");
    init_values();
    setup();
    theme();
    MainWindow::adjustSize();
}

MainWindow::~MainWindow()
{
    delete ui;
}

void MainWindow::setup () {
    imgActual = QImage(wSize, wSize, QImage::Format_RGB32);
    imgApparent = QImage(2*wSize, 2*wSize, QImage::Format_RGB32);
    imgDistorted = QImage(2*wSize, 2*wSize, QImage::Format_RGB32);
}

```

```

rocket = QPixmap(":/images/images/Tintin.png");
lensType = "point";

// Set max/min values for UI elements
ui->einsteinSlider->setMaximum(0.1*wSize);
ui->einsteinSpinBox->setMaximum(0.1*wSize);
ui->srcSizeSlider->setMaximum(0.1*wSize);
ui->srcSizeSpinBox->setMaximum(0.1*wSize);
ui->lensDistSlider->setMaximum(100);
ui->lensDistSpinBox->setMaximum(100);
ui->lensDistSlider->setMinimum(30);
ui->lensDistSpinBox->setMinimum(30);
ui->xSlider->setMaximum(wSize/2);
ui->xSpinBox->setMaximum(wSize/2);
ui->xSlider->setMinimum(-wSize/2);
ui->xSpinBox->setMinimum(-wSize/2);
ui->ySlider->setMaximum(wSize/2);
ui->ySpinBox->setMaximum(wSize/2);
ui->ySlider->setMinimum(-wSize/2);
ui->ySpinBox->setMinimum(-wSize/2);
ui->termsSpinBox->setMaximum(100);

// Connect sliders and spinboxes
connect(ui->einsteinSlider, SIGNAL(valueChanged(int)), ui->
    einsteinSpinBox, SLOT(setValue(int)));
connect(ui->einsteinSpinBox, SIGNAL(valueChanged(int)), ui->
    einsteinSlider, SLOT(setValue(int)));
connect(ui->srcSizeSpinBox, SIGNAL(valueChanged(int)), ui->
    srcSizeSlider, SLOT(setValue(int)));
connect(ui->srcSizeSlider, SIGNAL(valueChanged(int)), ui->
    srcSizeSpinBox, SLOT(setValue(int)));
connect(ui->lensDistSpinBox, SIGNAL(valueChanged(int)), ui->
    lensDistSlider, SLOT(setValue(int)));
connect(ui->lensDistSlider, SIGNAL(valueChanged(int)), ui->
    lensDistSpinBox, SLOT(setValue(int)));
connect(ui->xSpinBox, SIGNAL(valueChanged(int)), ui->xSlider, SLOT(
    setValue(int)));
connect(ui->xSlider, SIGNAL(valueChanged(int)), ui->xSpinBox, SLOT(
    setValue(int)));
connect(ui->ySpinBox, SIGNAL(valueChanged(int)), ui->ySlider, SLOT(
    setValue(int)));
connect(ui->ySlider, SIGNAL(valueChanged(int)), ui->ySpinBox, SLOT(
    setValue(int)));
}

```

```

void MainWindow::init_values () {
    grid = true;
    markers = true;
    legendCheck = true;
    gridSize = 2;
    einsteinR = wSize/20;
    srcSize = wSize/20;
    CHI = 0.65;
    actualX = 0;
    actualY = 0;
    source = ui->srcTypeComboBox->currentText ();

    // Set initial values for UI elements
    ui->einsteinSpinBox->setValue (einsteinR);
    ui->einsteinSlider->setSliderPosition (einsteinR);
    ui->srcSizeSpinBox->setValue (srcSize);
    ui->srcSizeSlider->setSliderPosition (srcSize);
    ui->lensDistSpinBox->setValue (CHI*100);
    ui->lensDistSlider->setSliderPosition (CHI*100);
    ui->xSpinBox->setValue (actualX);
    ui->xSlider->setSliderPosition (actualX);
    ui->ySpinBox->setValue (actualY);
    ui->ySlider->setSliderPosition (actualY);
    ui->gridBox->setChecked (grid);
    ui->markerBox->setChecked (markers);
    ui->actionMarkers->setChecked (markers);
    ui->actionLegend->setChecked (legendCheck);
    ui->infTermsCheckbox->setChecked (false);
}

void MainWindow::on_einsteinSpinBox_valueChanged ()
{
    // Set variables to current spinbox values
    einsteinR = ui->einsteinSpinBox->value ();
    updateImg ();
}

void MainWindow::on_srcSizeSpinBox_valueChanged ()
{
    srcSize = ui->srcSizeSpinBox->value ();
    updateImg ();
}

```

```
void MainWindow::on_lensDistSpinbox_valueChanged(int arg1)
{
    CHI = arg1/100.0;
    updateImg();
}
```

```
void MainWindow::on_xSpinbox_valueChanged()
{
    actualX = ui->xSpinbox->value();
    updateImg();
}
```

```
void MainWindow::on_ySpinbox_valueChanged()
{
    actualY = ui->ySpinbox->value();
    updateImg();
}
```

```
void MainWindow::on_gridBox_stateChanged(int arg1)
{
    grid = arg1;
    updateImg();
}
```

```
void MainWindow::on_markerBox_stateChanged(int arg1)
{
    markers = arg1;
    ui->actionMarkers->setChecked(arg1);
    updateImg();
}
```

```
void MainWindow::on_resetButton_clicked()
{
    init_values();
    updateImg();
}
```



```
void MainWindow::on_srcTypeComboBox_currentTextChanged(const QString &arg1
)
{
    source = arg1;
    updateImg();
}
```

```
void MainWindow::on_actionReset_triggered()
{
    init_values();
    updateImg();
}
```

```
void MainWindow::on_actionMarkers_toggled(bool arg1)
{
    markers = arg1;
    ui->markerBox->setChecked(arg1);
}
```

```
void MainWindow::on_actionLegend_toggled(bool arg1)
{
    legendCheck = arg1;
    updateImg();
}
```

```
void MainWindow::on_actionOff_triggered()
{
    ui->gridBox->setChecked(false);
    updateImg();
}
```

```
void MainWindow::on_action2x2_triggered()
{
    gridSize = 2;
    ui->gridBox->setChecked(true);
    updateImg();
}
```

```
void MainWindow::on_action4x4_triggered()
{
```

```
        gridSize = 4;
        ui->gridBox->setChecked(true);
        updateImg();
    }

void MainWindow::on_action8x8_triggered()
{
    gridSize = 8;
    ui->gridBox->setChecked(true);
    updateImg();
}

void MainWindow::on_action12x12_triggered()
{
    gridSize = 12;
    ui->gridBox->setChecked(true);
    updateImg();
}

void MainWindow::on_actionChange_resolution_triggered()
{
    QString currentSize = QString::number(wSize);
    unsigned int input = QDialog::getInt(this, "Change_resolution", "
        Current:_" + currentSize, wSize, 50, 2048, 50);
    wSize = input;
    setup();
    updateImg();
}

void MainWindow::on_actionCustom_triggered()
{
    QString currentSize = QString::number(gridSize);
    unsigned int input = QDialog::getInt(this, "Custom_grid_size", "
        Current:_" + currentSize + "x" + currentSize, gridSize, 2, 100);
    gridSize = input;
    updateImg();
}

void MainWindow::on_actionDark_mode_toggled(bool arg1)
{
    darkMode = arg1;
}
```

```

    theme () ;
}

void MainWindow::on_saveButton_clicked ()
{
    saveImage () ;
}

void MainWindow::on_actionSave_image_as_triggered ()
{
    saveImage () ;
}

void MainWindow::on_infTermsCheckbox_toggled (bool checked)
{
    if (checked) {
        mode = "infinite";
    } else {
        mode = "finite";
    }
    ui->termsSpinbox->setDisabled (checked);
    updateImg () ;
}

void MainWindow::on_termsSpinbox_valueChanged (int arg1)
{
    terms = arg1;
    updateImg () ;
}

//std::pair<double, double> Simulator::spherical(double r, double theta)
    const {
//    double ksi1 = 0;
//    double ksi2 = 0;

//    for (int m=1; m<=n; m++){
//        double frac = pow(r, m) / factorial_(m);
//        double subTerm1 = 0;
//        double subTerm2 = 0;
//        for (int s=(m+1)%2; s<=m+1 && s<n; s+=2){

```

```

//          double alpha = alphas_val[m][s];
//          double beta = betas_val[m][s];
//          int c_p = 1 + s/(m + 1);
//          int c_m = 1 - s/(m + 1);
//          subTerm1 += 1.0/4*((alpha*cos((s-1)*theta) + beta*sin((s-1)*
theta))*c_p + (alpha*cos((s+1)*theta) + beta*sin((s+1)*theta))*c_m);
//          subTerm2 += 1.0/4*(-alpha*sin((s-1)*theta) + beta*cos((s-1)
*theta))*c_p + (alpha*sin((s+1)*theta) - beta*cos((s+1)*theta))*c_m);
//      }
//      double term1 = frac*subTerm1;
//      double term2 = frac*subTerm2;
//      ksi1 += term1;
//      ksi2 += term2;
//      // Break summation if term is less than 1/100 of ksi or if ksi
is well outside frame
//      if ( (std::abs(term1) < std::abs(ksi1)/100000) && (std::abs(
term2) < std::abs(ksi2)/100000) || (ksi1 < -100000*size || ksi1 >
100000*size || ksi2 < -100000*size || ksi2 > 100000*size) ){
//          break;
//      }
//  }
//  return {ksi1, ksi2};
//}

```

```

void MainWindow::on_actionAbout_triggered()

```

```

{
    QMessageBox::about(this, "About_CosmoAI",
        "<h2>"
        "A_simulation_tool_for_gravitational_lensing"
        "</h2>"
        "<br>"
        "A_Bachelor_of_Science_project_by:"
        "<ul>"
        "<li>_Simon_Ingebrigtsen_</li>"
        "<li>_Sondre_W._Rem_y_</li>"
        "<li>_Einar_L._Austnes_</li>"
        "<li>_Simon_N._Runde_</li>"
        "</ul>"
        "<p><a_href='https://github.com/Simon-Ing/Cosmo_Ai'>Source_code</a></p>"
    );
}

```

```

void MainWindow::on_actionValues_triggered()

```

```

{

```

```
    msgBox.setWindowTitle (" Simulator_values " );  
    msgBox.setWindowModality ( Qt :: WindowModal );  
    msgBox.exec ( ) ;  
}
```

---

**mainwindow.ui**


---

```

<?xml version="1.0" encoding="UTF-8"?>
<ui version="4.0">
  <class>MainWindow</class>
  <widget class="QMainWindow" name="MainWindow">
    <property name="geometry">
      <rect>
        <x>0</x>
        <y>0</y>
        <width>842</width>
        <height>693</height>
      </rect>
    </property>
    <property name="windowTitle">
      <string>MainWindow</string>
    </property>
    <widget class="QWidget" name="centralwidget">
      <layout class="QGridLayout" name="gridLayout_2">
        <item row="0" column="0">
          <layout class="QGridLayout" name="gridLayout_7">
            <item row="3" column="1">
              <widget class="QSlider" name="xSlider">
                <property name="orientation">
                  <enum>Qt::Horizontal</enum>
                </property>
              </widget>
            </item>
            <item row="1" column="1">
              <widget class="QSlider" name="srcSizeSlider">
                <property name="orientation">
                  <enum>Qt::Horizontal</enum>
                </property>
              </widget>
            </item>
            <item row="1" column="0">
              <widget class="QLabel" name="label_3">
                <property name="toolTip">
                  <string>&lt ;html&gt;&lt ;head/&gt;&lt ;body&gt;&lt ;p&gt;Changes
                    size of source&lt ;/p&gt;&lt ;p&gt;In the case of a Gaussian
                    source, sets the value of &lt ;span style="font-size:10pt
                    ;&quot;&gt; &lt ;/span&gt;&lt ;/p&gt;&lt ;/body&gt;&lt ;/html&gt
                    ;</string>
                </property>
              </widget>
            </item>
          </layout>
        </item>
      </layout>
    </widget>
  </widget>
</ui>

```



```

        quot; font-size:10pt;&quot;&gt;ratio&lt;/span&gt;&lt;/p&gt;&lt;/body&gt;&lt;/html&gt;</string>
    </property>
    <property name="text">
        <string>Lens distance</string>
    </property>
</widget>
</item>
<item row="0" column="1">
    <widget class="QSlider" name="einsteinSlider">
        <property name="orientation">
            <enum>Qt::Horizontal</enum>
        </property>
    </widget>
</item>
<item row="4" column="1">
    <widget class="QSlider" name="ySlider">
        <property name="toolTip">
            <string/>
        </property>
        <property name="orientation">
            <enum>Qt::Horizontal</enum>
        </property>
    </widget>
</item>
<item row="4" column="0">
    <widget class="QLabel" name="label_6">
        <property name="toolTip">
            <string>&lt;/html&gt;&lt;/body&gt;&lt;/p&gt;Changes the
                actual Y position of the source&lt;/p&gt;&lt;/body&gt;&lt;/html&gt;</string>
        </property>
        <property name="toolTipDuration">
            <number>-1</number>
        </property>
        <property name="text">
            <string>Y position</string>
        </property>
    </widget>
</item>
<item row="4" column="2">
    <widget class="QSpinBox" name="ySpinbox"/>
</item>
<item row="0" column="0">
    <widget class="QLabel" name="label_7">

```



```

    <property name="toolTip">
      <string>&lt ;html&gt;&lt ;head/&gt;&lt ;body&gt;&lt ;p&gt;Change the
        Einstein radius &lt ;span style="font-size:10pt; font-
          weight:700; font-style:italic;"&gt;R&lt ;/span&gt;&lt ;span
            style="font-size:10pt; font-weight:700; font-style:
              italic; vertical-align:sub;"&gt;E&lt ;/span&gt;&lt ;/p&gt;&
              lt ;/body&gt;&lt ;/html&gt;</string>
    </property>
    <property name="text">
      <string>Einstein radius</string>
    </property>
  </widget>
</item>
</layout>
</item>
<item row="0" column="1">
  <layout class="QGridLayout" name="gridLayout_6">
    <property name="sizeConstraint">
      <enum>QLayout::SetDefaultConstraint</enum>
    </property>
    <item row="0" column="0">
      <widget class="QLabel" name="label_4">
        <property name="toolTip">
          <string>&lt ;html&gt;&lt ;head/&gt;&lt ;body&gt;&lt ;p&gt;Set terms &
            lt ;span style="font-weight:700; font-style:italic;"&gt;
              &gt;m&lt ;/span&gt; in Roulette&lt ;/p&gt;&lt ;/body&gt;&lt ;/
              html&gt;</string>
          </property>
        <property name="text">
          <string>Terms</string>
        </property>
      </widget>
    </item>
    <item row="4" column="0">
      <widget class="QCheckBox" name="gridBox">
        <property name="toolTip">
          <string/>
        </property>
        <property name="text">
          <string>Grid</string>
        </property>
      </widget>
    </item>
    <item row="5" column="1">
      <widget class="QPushButton" name="resetButton">

```

```

    <property name="toolTip">
      <string>Reset inputs to initial values</string>
    </property>
    <property name="toolTipDuration">
      <number>1</number>
    </property>
    <property name="text">
      <string>Reset</string>
    </property>
  </widget>
</item>
<item row="0" column="2">
  <layout class="QHBoxLayout" name="horizontalLayout_3"/>
</item>
<item row="3" column="1">
  <widget class="QComboBox" name="srcTypeComboBox">
    <item>
      <property name="text">
        <string>Gauss</string>
      </property>
    </item>
    <item>
      <property name="text">
        <string>Circle</string>
      </property>
    </item>
    <item>
      <property name="text">
        <string>Square</string>
      </property>
    </item>
    <item>
      <property name="text">
        <string>Rocket</string>
      </property>
    </item>
  </widget>
</item>
<item row="4" column="1">
  <widget class="QCheckBox" name="markerBox">
    <property name="toolTip">
      <string/>
    </property>
    <property name="text">
      <string>Markers</string>

```

```

    </property>
    <property name="checked">
      <bool>false </bool>
    </property>
  </widget>
</item>
<item row="3" column="0">
  <widget class="QLabel" name="label_5">
    <property name="toolTip">
      <string/>
    </property>
    <property name="text">
      <string>Source type</string>
    </property>
  </widget>
</item>
<item row="5" column="0">
  <widget class="QPushButton" name="saveButton">
    <property name="toolTip">
      <string>Save images to computer</string>
    </property>
    <property name="text">
      <string>Save</string>
    </property>
  </widget>
</item>
<item row="0" column="1">
  <widget class="QSpinBox" name="termsSpinbox">
    <property name="sizePolicy">
      <sizepolicy hstretch="Fixed" vstretch="Fixed">
        <horstretch>0</horstretch>
        <verstretch>0</verstretch>
      </sizepolicy>
    </property>
  </widget>
</item>
<item row="1" column="1">
  <widget class="QCheckBox" name="infTermsCheckbox">
    <property name="toolTip">
      <string>&lt ;html&gt;&lt ;head/&gt;&lt ;body&gt;&lt ;p&gt; Infinite
        terms mode&lt ;/p&gt;&lt ;/body&gt;&lt ;/html&gt;</string>
    </property>
    <property name="text">
      <string>Infinite</string>
    </property>
  </widget>
</item>

```

```

    </widget>
  </item>
</layout>
</item>
<item row="1" column="0" colspan="2">
  <layout class="QGridLayout" name="gridLayout_5">
    <item row="0" column="0">
      <widget class="QLabel" name="actLabel">
        <property name="sizePolicy">
          <sizepolicy hstretch="Expanding" vstretch="Expanding">
            <horstretch>0</horstretch>
            <verstretch>0</verstretch>
          </sizepolicy>
        </property>
        <property name="minimumSize">
          <size>
            <width>300</width>
            <height>300</height>
          </size>
        </property>
        <property name="toolTip">
          <string>&lt ;html&gt;&lt ;head/&gt;&lt ;body&gt;&lt ;p&gt;This is the
            actual image showing the undistorted source in its actual
            position&lt ;/p&gt;&lt ;/body&gt;&lt ;/html&gt;</ string>
        </property>
        <property name="text">
          <string>TextLabel</ string>
        </property>
      </widget>
    </item>
    <item row="0" column="1">
      <widget class="QLabel" name="distLabel">
        <property name="sizePolicy">
          <sizepolicy hstretch="Expanding" vstretch="Expanding">
            <horstretch>0</horstretch>
            <verstretch>0</verstretch>
          </sizepolicy>
        </property>
        <property name="minimumSize">
          <size>
            <width>300</width>
            <height>300</height>
          </size>
        </property>
        <property name="toolTip">

```

```

        <string>&lt;html&gt;&lt;head/&gt;&lt;body&gt;&lt;p&gt;This is the
            distorted image showing the source being distorted by the
            lens&lt;/p&gt;&lt;/body&gt;&lt;/html&gt;</string>
    </property>
    <property name="text">
        <string>TextLabel</string>
    </property>
</widget>
</item>
</layout>
</item>
</layout>
</widget>
<widget class="QMenuBar" name="menubar">
    <property name="geometry">
        <rect>
            <x>0</x>
            <y>0</y>
            <width>842</width>
            <height>22</height>
        </rect>
    </property>
    <widget class="QMenu" name="menuFile">
        <property name="title">
            <string>File</string>
        </property>
        <addaction name="actionReset"/>
        <addaction name="actionSave_image_as"/>
    </widget>
    <widget class="QMenu" name="menuView">
        <property name="title">
            <string>View</string>
        </property>
    <widget class="QMenu" name="menuGrid_2">
        <property name="title">
            <string>Grid</string>
        </property>
        <addaction name="actionOff"/>
        <addaction name="action2x2"/>
        <addaction name="action4x4"/>
        <addaction name="action8x8"/>
        <addaction name="action12x12"/>
        <addaction name="actionCustom"/>
    </widget>
    <addaction name="menuGrid_2"/>

```

```
<addaction name="actionMarkers" />
<addaction name="actionLegend" />
<addaction name="actionValues" />
<addaction name="separator" />
<addaction name="actionChange_resolution" />
<addaction name="separator" />
<addaction name="actionDark_mode" />
</widget>
<widget class="QMenu" name="menuHelp">
  <property name="title">
    <string>Help</string>
  </property>
  <addaction name="actionAbout" />
</widget>
<addaction name="menuFile" />
<addaction name="menuView" />
<addaction name="menuHelp" />
</widget>
<widget class="QStatusBar" name="statusbar" />
<action name="actionReset">
  <property name="text">
    <string>Reset</string>
  </property>
</action>
<action name="actionExport">
  <property name="text">
    <string>Export</string>
  </property>
</action>
<action name="actionPrint">
  <property name="text">
    <string>Print</string>
  </property>
</action>
<action name="actionMarkers">
  <property name="checkable">
    <bool>true</bool>
  </property>
  <property name="checked">
    <bool>>false</bool>
  </property>
  <property name="text">
    <string>Markers</string>
  </property>
</action>
```

```
<action name="actionLegend">
  <property name="checkable">
    <bool>true</bool>
  </property>
  <property name="checked">
    <bool>>false</bool>
  </property>
  <property name="text">
    <string>Legend</string>
  </property>
</action>
<action name="actionOff">
  <property name="checkable">
    <bool>>false</bool>
  </property>
  <property name="text">
    <string>Off</string>
  </property>
</action>
<action name="action2x2">
  <property name="checkable">
    <bool>>false</bool>
  </property>
  <property name="text">
    <string>2x2</string>
  </property>
</action>
<action name="action4x4">
  <property name="text">
    <string>4x4</string>
  </property>
</action>
<action name="action8x8">
  <property name="text">
    <string>8x8</string>
  </property>
</action>
<action name="action12x12">
  <property name="text">
    <string>12x12</string>
  </property>
</action>
<action name="actionChange_resolution">
  <property name="text">
    <string>Change resolution</string>
```

```
    </property>
</action>
<action name="actionCustom">
  <property name="text">
    <string>Custom</string>
  </property>
</action>
<action name="actionDark_mode">
  <property name="checkable">
    <bool>true</bool>
  </property>
  <property name="checked">
    <bool>true</bool>
  </property>
  <property name="text">
    <string>Dark mode</string>
  </property>
</action>
<action name="actionSave_image_as">
  <property name="text">
    <string>Save image as... </string>
  </property>
</action>
<action name="actionAbout">
  <property name="text">
    <string>About</string>
  </property>
</action>
<action name="actionValues">
  <property name="text">
    <string>Show additional values</string>
  </property>
</action>
</widget>
<resources/>
<connections/>
</ui>
```

---



## F.3 Machine Learning

deepshit.py

---

```

import os
import platform
import shutil
import warnings
from typing import Optional, List, Callable, Tuple

import cv2
import torch
from torch import Tensor
import torch.nn as nn
import torch.nn.functional as func
from torchvision.datasets import ImageFolder
from torchvision.models import InceptionOutputs
from torchvision.models.inception import BasicConv2d, InceptionB,
    InceptionD, InceptionAux, InceptionA, InceptionC, \
    InceptionE
from torchvision.transforms import transforms
import smtplib
import ssl

class ConvNet(nn.Module):
    def __init__(self):
        super(ConvNet, self).__init__()
        self.conv1 = nn.Conv2d(1, 4, (5, 5))
        self.conv2 = nn.Conv2d(4, 7, (5, 5))
        self.conv3 = nn.Conv2d(7, 10, (5, 5))
        self.conv4 = nn.Conv2d(10, 10, (5, 5))
        self.pool = nn.MaxPool2d(2, 2)
        self.fc1 = nn.Linear(40, 12)
        self.fc2 = nn.Linear(12, 5)

    def forward(self, x):
        x = self.pool(func.relu(self.conv1(x)))
        x = self.pool(func.relu(self.conv2(x)))
        x = self.pool(func.relu(self.conv3(x)))
        x = self.pool(func.relu(self.conv4(x)))
        x = self.pool(func.relu(self.conv4(x)))
        x = self.pool(func.relu(self.conv4(x))).view(-1, 40)
        x = self.fc2(func.relu((self.fc1(x))))

```

```
return x
```

```
class ConvNetNew(nn.Module):
```

```
    def __init__(self):
```

```
        super(ConvNetNew, self).__init__()
```

```
        self.conv1 = nn.Conv2d(1, 4, (10, 10))
```

```
        self.conv2 = nn.Conv2d(4, 8, (10, 10))
```

```
        self.conv3 = nn.Conv2d(8, 8, (5, 5))
```

```
        self.pool = nn.MaxPool2d(2, 2)
```

```
        self.fc1 = nn.Linear(512, 64)
```

```
        self.fc2 = nn.Linear(64, 4)
```

```
    def forward(self, x):
```

```
        x = self.pool(func.relu(self.conv1(x)))
```

```
        x = self.pool(func.relu(self.conv2(x)))
```

```
        x = self.pool(func.relu(self.conv3(x)))
```

```
        x = self.pool(func.relu(self.conv3(x)))
```

```
        x = self.pool(func.relu(self.conv3(x))).view(-1, 512)
```

```
        x = self.fc2(func.relu(self.fc1(x)))
```

```
        return x
```

```
class ConvNet3(nn.Module):
```

```
    def __init__(self):
```

```
        super(ConvNet3, self).__init__()
```

```
        self.conv1 = nn.Conv2d(1, 4, (5, 5))
```

```
        self.conv2 = nn.Conv2d(4, 8, (5, 5))
```

```
        self.conv3 = nn.Conv2d(8, 8, (5, 5))
```

```
        self.pool = nn.MaxPool2d(2, 2)
```

```
        self.fc1 = nn.Linear(32, 4)
```

```
    def forward(self, x):
```

```
        x = self.pool(func.relu(self.conv1(x)))
```

```
        x = self.pool(func.relu(self.conv2(x)))
```

```
        x = self.pool(func.relu(self.conv3(x)))
```

```
        x = self.pool(func.relu(self.conv3(x)))
```

```
        x = self.pool(func.relu(self.conv3(x)))
```

```
        x = self.pool(func.relu(self.conv3(x))).view(-1, 32)
```

```
        # print(x.shape)
```

```
        x = self.fc1(x)
```

```
        return x
```

```
class ProConvNet(nn.Module):
```

```
    def __init__(self):
```

```

super(ProConvNet, self).__init__()
self.conv1 = nn.Conv2d(1, 4, 3)
self.conv2 = nn.Conv2d(4, 8, 3)
self.conv3 = nn.Conv2d(8, 16, 2)
self.conv4 = nn.Conv2d(16, 16, 2)

self.fc1 = nn.Linear(1024)
self.fc2 = nn.Linear(1024, 1024)
self.fc3 = nn.Linear(1024, 128)
self.fc4 = nn.Linear(128, 4)

def forward(self, x):
    # Max pooling over a (2, 2) window

    x = func.max_pool2d(func.relu(self.conv1(x)), 2)
    x = func.max_pool2d(func.relu(self.conv2(x)), 2)
    x = func.max_pool2d(func.relu(self.conv3(x)), 2)
    x = func.max_pool2d(func.relu(self.conv4(x)), 2)
    x = func.max_pool2d(func.relu(self.conv4(x)), 2)

    #print(x.shape)
    x = torch.flatten(x, 1) # flatten all dimensions except the batch
        dimension
    x = func.relu(self.fc1(x))
    x = func.relu(self.fc2(x))
    x = func.relu(self.fc2(x))
    x = func.relu(self.fc3(x))
    x = self.fc4(x)
    return x

class AlexNet(nn.Module):
    def __init__(self, num_classes: int = 1000, dropout: float = 0.5) ->
    None:
        super() . __init__()

    self.features = nn.Sequential(
        nn.Conv2d(1, 64, kernel_size=11, stride=4, padding=2),
        nn.ReLU(inplace=True),
        nn.MaxPool2d(kernel_size=3, stride=2),
        nn.Conv2d(64, 192, kernel_size=5, padding=2),
        nn.ReLU(inplace=True),
        nn.MaxPool2d(kernel_size=3, stride=2),
        nn.Conv2d(192, 384, kernel_size=3, padding=1),
        nn.ReLU(inplace=True),

```

```

        nn.Conv2d(384, 256, kernel_size=3, padding=1),
        nn.ReLU(inplace=True),
        nn.Conv2d(256, 256, kernel_size=3, padding=1),
        nn.ReLU(inplace=True),
        nn.MaxPool2d(kernel_size=3, stride=2),
    )
    self.avgpool = nn.AdaptiveAvgPool2d((6, 6))
    self.classifier = nn.Sequential(
        nn.Dropout(p=dropout),
        nn.Linear(256 * 6 * 6, 4096),
        nn.ReLU(inplace=True),
        nn.Dropout(p=dropout),
        nn.Linear(4096, 4096),
        nn.ReLU(inplace=True),
        nn.Linear(4096, 5),
    )

    def forward(self, x: torch.Tensor) -> torch.Tensor:
        x = self.features(x)
        x = self.avgpool(x)
        x = torch.flatten(x, 1)
        #print(x.shape)
        x = self.classifier(x)
        return x

class AlexMulti(nn.Module):
    def __init__(self, num_classes: int = 1000) -> None:
        super().__init__()
        self.features = nn.Sequential(
            nn.Conv2d(1, 64, kernel_size=11, stride=4, padding=2),
            nn.ReLU(inplace=True),
            nn.MaxPool2d(kernel_size=3, stride=2),
            nn.Conv2d(64, 192, kernel_size=5, padding=2),
            nn.ReLU(inplace=True),
            nn.MaxPool2d(kernel_size=3, stride=2),
            nn.Conv2d(192, 384, kernel_size=3, padding=1),
            nn.ReLU(inplace=True),
            nn.Conv2d(384, 256, kernel_size=3, padding=1),
            nn.ReLU(inplace=True),
            nn.Conv2d(256, 256, kernel_size=3, padding=1),
            nn.ReLU(inplace=True),
            nn.MaxPool2d(kernel_size=3, stride=2),
        )
        self.avgpool = nn.AdaptiveAvgPool2d((6, 6))

```

```

self.fc = nn.Sequential(
    nn.Linear(256 * 6 * 6 + 1, 4096),
    nn.ReLU(inplace=True),
    nn.Linear(4096, 400),
    nn.ReLU(inplace=True),
    nn.Linear(400, 40),
    nn.ReLU(inplace=True),
    nn.Linear(40, num_classes)
)

def forward(self, x: torch.Tensor, chi: torch.Tensor) -> torch.Tensor:
    x = self.features(x)
    x = self.avgpool(x)
    x = torch.flatten(x, 1)
    chi = chi.view(chi.shape[0], 1)
    x = torch.cat((x, chi), 1)
    x = self.fc(x)
    return x

def cuda_if_available():
    if torch.cuda.is_available():
        print("Running_Cuda!")
        return torch.device('cuda')
    print("Running_on_cpu")
    return torch.device("cpu")

def load_model(model):
    ans = ""
    while True:
        while ans not in ("y", "Y", "N", "n"):
            ans = input("Load_previous_model?(y/n):_")
        if ans == "n" or ans == "N":
            break
        if ans == "y" or ans == "Y":
            try:
                path = "Models/" + input("enter_path:_")
                model.load_state_dict(torch.load(path))
                break
            except FileNotFoundError as e:
                print(e)
                ans = ""
        except IsADirectoryError:

```

```

        print("You_must_enter_a_file_name_goddamnit!")
        ans = ""

def save_model(model):
    pass
    ans = ""
    while ans not in ("y", "Y", "N", "n"):
        ans = input("Save_model?(y/n):_")
    if ans == "y" or ans == "Y":
        path = "Models/" + input("Enter_a_name:_")
        torch.save(model.state_dict(), path)

def dataset_from_png(n_samples, size, folder, gen_new):
    if platform.system() == 'Windows':
        _, current_folder = os.path.split(os.getcwd())
        if current_folder != "Python":
            os.chdir('Python')
        if gen_new:
            print(f"Started_generating_{folder}_data")
            shutil.rmtree(folder)
            os.makedirs(f'{folder}/images')
            os.system('Datagen.exe_' + str(n_samples) + "_" + str(size) +
                "_" + str(folder) + "_" + str(5))
            print("Done_generating,_start_loading")
        else:
            if gen_new:
                print(f"Started_generating_{folder}_data")
                os.system(f'm_r_{folder}')
                os.system(f'mkdir_{folder}')
                os.system(f'mkdir_{folder}/images')
                os.system('./Datagen_' + str(n_samples) + "_" + str(size) + "_"
                    + str(folder) + "_" + str(5))
                print("Done_generating,_start_loading")

            dataset = CosmoDatasetPng(str(folder))
            print("Done_loading")
            return dataset

def test_network(loader, model_, criterion_, device, print_results=False):
    total_loss = 0
    n_batches = 0

```

```

with torch.no_grad():
    for images, params in loader:
        images = images.to(device)
        params = params.to(device)
        output = model_(images)
        loss_ = criterion_(output, params)
        total_loss += loss_
        n_batches += 1
        if print_results:
            for i, param in enumerate(params):
                niceoutput = [round(n, 4) for n in output[i].tolist()]
                niceparam = [round(n, 4) for n in param.tolist()]
                print(f"{'Correct:_' + niceparam + '<50'}{'Output:_' +
                    niceoutput + '^50'}")
    return total_loss / n_batches

def print_images(images, params):
    for i, img in enumerate(images):
        image = images[i].numpy().reshape(images[i].shape[1], images[i].
            shape[2], 1)
        image = cv2.resize(image, (400, 400))
        image = cv2.putText(image, f'_{dist:_' + str(params[i][0].item()) + '_in
            :_' + str(params[i][1].item()) + '_sigma:_' + str(params[i][2].item()) + '_
            x:_' + str(params[i][3].item()) + '_y:_' + str(params[i][4].item()) + '}',
            (0, 50), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (255, 255, 255), 1, cv2
            .LINE_AA)
        cv2.imshow("img", image)
        cv2.waitKey(0)

class CosmoDatasetPng(ImageFolder):
    def __init__(self, root):
        super(CosmoDatasetPng, self).__init__(root, transform=transforms.
            ToTensor())
        if platform.system() == 'Windows':
            self.targets = torch.tensor([[int(a), int(b), int(c), int(d),
                int(e)] for (a, b, c, d, e) in [t[0].lstrip(
                    root + "\\images\\").rstrip(".png").split(",") for t in
                    self.imgs]], dtype=torch.float)
        else:
            self.targets = torch.tensor([[int(a), int(b), int(c), int(d),
                int(e)] for (a, b, c, d, e) in [t[0].lstrip(

```

```

        root + "/images/").rstrip(".png").split(",") for t in self
            .imgs]], dtype=torch.float)

def __getitem__(self, item):
    path, _ = self.samples[item]
    sample = self.loader(path)
    if self.transform is not None:
        sample = self.transform(sample)
    return sample[0].view(1, sample.shape[1], sample.shape[2]), self.
        targets[item]

def send_email(receiver_email, message):
    port = 465 # For SSL
    smtp_server = "smtp.gmail.com"
    sender_email = "simon.ing.dev@gmail.com" # Enter your address
    password = "developer69"
    message = "Subject: Training finished\n\n" + message
    context = ssl.create_default_context()
    with smtplib.SMTP_SSL(smtp_server, port, context=context) as server:
        server.login(sender_email, password)
        server.sendmail(sender_email, receiver_email, message)

class Inception3(nn.Module):
    def __init__(
        self,
        num_outputs: int = 5,
        aux_logits: bool = True,
        transform_input: bool = False,
        inception_blocks: Optional[List[Callable[... , nn.Module]]] =
            None,
        init_weights: Optional[bool] = None,
        dropout: float = 0.5,
    ) -> None:
        super().__init__()
        # _log_api_usage_once(self)
        if inception_blocks is None:
            inception_blocks = [BasicConv2d, InceptionA, InceptionB,
                InceptionC, InceptionD, InceptionE, InceptionAux]
        if init_weights is None:
            warnings.warn(
                "The default weight initialization of inception_v3 will be
                changed in future releases of "

```



```

        "torchvision._If_you_wish_to_keep_the_old_behavior_(which_
          leads_to_long_initialization_times"
        "_due_to_scipy/scipy#11299),_please_set_init_weights=True."
        ",
        FutureWarning,
    )
    init_weights = True
if len(inception_blocks) != 7:
    raise ValueError(f"length_of_inception_blocks_should_be_7_
        instead_of_{len(inception_blocks)}")
conv_block = inception_blocks[0]
inception_a = inception_blocks[1]
inception_b = inception_blocks[2]
inception_c = inception_blocks[3]
inception_d = inception_blocks[4]
inception_e = inception_blocks[5]
inception_aux = inception_blocks[6]

self.aux_logits = aux_logits
self.transform_input = transform_input
self.Conv2d_1a_3x3 = conv_block(1, 32, kernel_size=3, stride=2)
self.Conv2d_2a_3x3 = conv_block(32, 32, kernel_size=3)
self.Conv2d_2b_3x3 = conv_block(32, 64, kernel_size=3, padding=1)
self.maxpool1 = nn.MaxPool2d(kernel_size=3, stride=2)
self.Conv2d_3b_1x1 = conv_block(64, 80, kernel_size=1)
self.Conv2d_4a_3x3 = conv_block(80, 192, kernel_size=3)
self.maxpool2 = nn.MaxPool2d(kernel_size=3, stride=2)
self.Mixed_5b = inception_a(192, pool_features=32)
self.Mixed_5c = inception_a(256, pool_features=64)
self.Mixed_5d = inception_a(288, pool_features=64)
self.Mixed_6a = inception_b(288)
self.Mixed_6b = inception_c(768, channels_7x7=128)
self.Mixed_6c = inception_c(768, channels_7x7=160)
self.Mixed_6d = inception_c(768, channels_7x7=160)
self.Mixed_6e = inception_c(768, channels_7x7=192)
self.AuxLogits: Optional[nn.Module] = None
if aux_logits:
    self.AuxLogits = inception_aux(768, num_outputs)
self.Mixed_7a = inception_d(768)
self.Mixed_7b = inception_e(1280)
self.Mixed_7c = inception_e(2048)
self.avgpool = nn.AdaptiveAvgPool2d((1, 1))
self.dropout = nn.Dropout(p=dropout)
self.fc = nn.Linear(2048, num_outputs)
if init_weights:

```

```

    for m in self.modules():
        if isinstance(m, nn.Conv2d) or isinstance(m, nn.Linear):
            stddev = float(m.stddev) if hasattr(m, "stddev") else
                0.1 # type: ignore
            torch.nn.init.trunc_normal_(m.weight, mean=0.0, std=
                stddev, a=-2, b=2)
        elif isinstance(m, nn.BatchNorm2d):
            nn.init.constant_(m.weight, 1)
            nn.init.constant_(m.bias, 0)

def _transform_input(self, x: Tensor) -> Tensor:
    if self.transform_input:
        x_ch0 = torch.unsqueeze(x[:, 0], 1) * (0.229 / 0.5) + (0.485 -
            0.5) / 0.5
        x_ch1 = torch.unsqueeze(x[:, 1], 1) * (0.224 / 0.5) + (0.456 -
            0.5) / 0.5
        x_ch2 = torch.unsqueeze(x[:, 2], 1) * (0.225 / 0.5) + (0.406 -
            0.5) / 0.5
        x = torch.cat((x_ch0, x_ch1, x_ch2), 1)
    return x

def _forward(self, x: Tensor) -> Tuple[Tensor, Optional[Tensor]]:
    # N x 3 x 299 x 299
    x = self.Conv2d_1a_3x3(x)
    # N x 32 x 149 x 149
    x = self.Conv2d_2a_3x3(x)
    # N x 32 x 147 x 147
    x = self.Conv2d_2b_3x3(x)
    # N x 64 x 147 x 147
    x = self.maxpool1(x)
    # N x 64 x 73 x 73
    x = self.Conv2d_3b_1x1(x)
    # N x 80 x 73 x 73
    x = self.Conv2d_4a_3x3(x)
    # N x 192 x 71 x 71
    x = self.maxpool2(x)
    # N x 192 x 35 x 35
    x = self.Mixed_5b(x)
    # N x 256 x 35 x 35
    x = self.Mixed_5c(x)
    # N x 288 x 35 x 35
    x = self.Mixed_5d(x)
    # N x 288 x 35 x 35
    x = self.Mixed_6a(x)
    # N x 768 x 17 x 17

```

```

x = self.Mixed_6b(x)
# N x 768 x 17 x 17
x = self.Mixed_6c(x)
# N x 768 x 17 x 17
x = self.Mixed_6d(x)
# N x 768 x 17 x 17
x = self.Mixed_6e(x)
# N x 768 x 17 x 17
aux: Optional[Tensor] = None
if self.AuxLogits is not None:
    if self.training:
        aux = self.AuxLogits(x)
# N x 768 x 17 x 17
x = self.Mixed_7a(x)
# N x 1280 x 8 x 8
x = self.Mixed_7b(x)
# N x 2048 x 8 x 8
x = self.Mixed_7c(x)
# N x 2048 x 8 x 8
# Adaptive average pooling
x = self.avgpool(x)
# N x 2048 x 1 x 1
x = self.dropout(x)
# N x 2048 x 1 x 1
x = torch.flatten(x, 1)
# N x 2048
x = self.fc(x)
# N x 1000 (num_classes)
return x, aux

@torch.jit.unused
def eager_outputs(self, x: Tensor, aux: Optional[Tensor]) ->
InceptionOutputs:
    if self.training and self.aux_logits:
        return InceptionOutputs(x, aux)
    else:
        return x # type: ignore[return-value]

def forward(self, x: Tensor) -> InceptionOutputs:
    # x = self._transform_input(x)
    x, aux = self._forward(x)
    return x
    # aux_defined = self.training and self.aux_logits
    # if torch.jit.is_scripting():
    #     if not aux_defined:

```

```
#         warnings.warn("Scripted Inception3 always returns  
#         Inception3 Tuple")  
#         return InceptionOutputs(x, aux)  
#     else:  
#         return self.eager_outputs(x, aux)
```

---

**CosmoCNN.py**

---

```
from torch.utils.data import DataLoader
from deepshit import *
import time
import torch.optim.lr_scheduler as lr_scheduler
from tqdm import tqdm
from torch.cuda.amp import autocast

# Training parameters
num_epochs = 100
batch_size = 32
learning_rate = 0.001

n_train_samples = 100000
n_test_samples = 5000
img_size = 512

# set to true when you want new data points
gen_new_train = 0
gen_new_test = 0
load_checkpoint = False
checkpoint_path = "Models/autosave/autosave_epoch40"

device = cuda_if_available() # Use cuda if available

# Initialize your network, loss function, optimizer and scheduler
model = Inception3().to(device)
criterion = nn.MSELoss()
optimizer = torch.optim.Adam(model.parameters(), lr=learning_rate)
scheduler = lr_scheduler.ReduceLROnPlateau(optimizer, patience=5, factor
=0.5)

# Load a dataset for training and one for verification
train_dataset = dataset_from_png(n_samples=n_train_samples, size=img_size,
    folder="train", gen_new=gen_new_train)
train_loader = DataLoader(dataset=train_dataset, batch_size=batch_size,
    shuffle=True)
test_dataset = dataset_from_png(n_samples=n_test_samples, size=img_size,
    folder="test", gen_new=gen_new_test)
test_loader = DataLoader(dataset=test_dataset, batch_size=batch_size)

# Load from checkpoint if desired:
```

```

if (load_checkpoint):
    checkpoint = torch.load(checkpoint_path)
    model.load_state_dict(checkpoint['model_state_dict'])
    optimizer.load_state_dict(checkpoint['optimizer_state_dict'])
    epoch = checkpoint['epoch']
    loss = checkpoint['loss']

# Test network prior to training
loss = test_network(test_loader, model, criterion, device)
print(f'\nAverage_loss_over_test_data_before_training:_{loss}\n')

timer = time.time()
print(f'Start_training, _num_epochs:_{num_epochs}, _batch_size:_{batch_size}
    }, _lr:_{learning_rate}, \
    _____train_samples:_{n_train_samples}_test_samples:_{n_test_samples}_
    img_size:_{img_size}')

scaler = torch.cuda.amp.GradScaler()

# Training loop
try:
    for epoch in tqdm(range(num_epochs), desc="Total"):
        model.train()
        for i, (images, params) in enumerate(tqdm(train_loader, desc='
            Epoch')):
            images = images.to(device)
            params = params.to(device)

            optimizer.zero_grad()

            # Forward pass

            with autocast():
                output = model(images)
                loss = criterion(output, params)
            # Backward pass
            scaler.scale(loss).backward()
            scaler.step(optimizer)
            scaler.update()

            scheduler.step(loss)
        # Test network for each epoch
        model.eval()
        loss = test_network(test_loader, model, criterion, device,
            print_results=False)

```

```

print (f"\nEpoch:_{epoch+1},_Loss:_{loss}_lr:_{optimizer.state_dict
    ()['_param_groups'][0]['lr']},_time:_{time.time()-_timer}\n")

# Save checkpoint
if (epoch % 20 == 0) and (epoch > 0):
    autosave_path = "Models/autosave/" + "autosave_epoch" + str(
        epoch)
    torch.save({
        'epoch': epoch,
        'model_state_dict': model.state_dict(),
        'optimizer_state_dict': optimizer.state_dict(),
        'loss': loss,
    }, autosave_path)

except KeyboardInterrupt:
    print ("Training_aborted_by_keyboard_interrupt.")
except TypeError:
    print ("Training_aborted_by_keyboard_interrupt.")

# Test network after training
loss = test_network(test_loader, model, criterion, device, print_results=
    True)
message = f'Loss:_{loss}\nEpochs:_{num_epochs}\nbatch_size:_{batch_size}\
    nlearning_rate:_{learning_rate}\nn_train_samples:_{n_train_samples}\
    nimg_size:_{img_size}'

print(message)

# send_email('simon.ing.89@gmail.com', message)

# Save your model if you want to
save_model(model)

```

---

**best\_test\_in\_the\_west.py**

---

```
from deepshit import *
from torch.utils.data import DataLoader

#load model
device = cuda_if_available() # Use cuda if available
model = AlexNet().to(device)
#model = torch.nn.DataParallel(model)
PATH = "Python/Models/73loss_distratio_200k"
#checkpoint = torch.load(PATH)

#model.load_state_dict(checkpoint['model_state_dict'])
model.load_state_dict(torch.load(PATH))
model.eval()

#load test data
batch_size = 1
test_dataset = dataset_from_png(n_samples=5000, size=512, folder="test",
    gen_new=False)
test_loader = DataLoader(dataset=test_dataset, batch_size=batch_size)

# Test network
criterion = nn.MSELoss()
loss = test_network(test_loader, model, criterion, device, print_results=
    True)
message = f'Loss:_{loss}\n_{batch_size}'
print(message)
```

---



**analyze.py**


---

```

import matplotlib.pyplot as plt
import matplotlib
from numpy import arctan2

matplotlib.use('TkAgg')
from math import factorial, cos, sin, pi, sqrt
import dill
dill.settings['recurse'] = True

from sympy import symbols
import numpy as np

x, y, g, c = symbols("x_y_g_c")

n = 20

size = 100

# alphas = [[0 for i in range(n)]for j in range(n)]
# betas = [[0 for i in range(n)]for j in range(n)]
#
# with open('../functions_20.txt') as file:
#     lines = file.readlines()
#     for line in lines:
#         i, j, alpha_, beta_ = line.split(":")
#         alphas[int(i)][int(j)] = lambdify((x, y, g, c), parse_expr(
alpha_))
#         betas[int(i)][int(j)] = lambdify((x, y, g, c), parse_expr(beta_
)
#
# with open("alphas", "wb") as outfile:
#     dill.dump(alphas, outfile)
#
# with open("betas", "wb") as outfile:
#     dill.dump(betas, outfile)

with open("alphas", "rb") as infile:
    alphas = dill.load(infile)

```

```

with open("betas", "rb") as infile:
    betas = dill.load(infile)

# alphas, betas = multi(n)

img = np.zeros((size, size))
dist = np.zeros((size, size))
CHI = 1
GAMMA = 1
X = 10
Y = 0

# for row in range(100):
#     for col in range(100):
#         x = col - 50 - app_pos
#         y = 50 - row

r = 10
theta = 1
ksi1 = 0
ksi2 = 0
ksils_ = []
ksi2s_ = []

fig = plt.figure(figsize=(10, 8))

for i, r in enumerate([5, 10, 15, 20]):
    ksils = []
    ksi2s = []
    k = 0
    for m in range(1, n):
        frac = r**m / factorial(m)
        sub_term1 = 0
        sub_term2 = 0
        for s in range((m+1) % 2, min(m+2, n), 2):
            alpha = alphas[m][s](X, Y, CHI, GAMMA)
            beta = betas[m][s](X, Y, CHI, GAMMA)
            c_p = 1 + s/(m + 1)
            c_m = 1 - s/(m + 1)
            sub_term1 += 1/2*((alpha*cos((s-1)*theta) + beta*sin((s-1)*
                theta)*c_p + (alpha*cos((s+1)*theta) + beta*sin(s+1)*theta)
                )*c_m)

```

```

        sub_term2 += 1/2*((-alpha*sin((s-1)*theta) + beta*cos((s-1)*
            theta)*c_p + (alpha*sin((s+1)*theta) - beta*cos(s+1)*theta)
            )*c_m)
        term1 = frac*sub_term1
        term2 = frac*sub_term2
        ksi1 += term1
        ksi2 += term2
        ksi1s.append(ksi1)
        ksi2s.append(ksi2)
    ax = plt.subplot(2, 2, i+1)
    # ax.title.set_text(f'theta = {theta}')
    t = [i for i, _ in enumerate(ksi1s)]
    ax.title.set_text(f'r_{i}={r}')
    plt.plot(t, ksi1s, label="ksi_1")
    plt.plot(t, ksi2s, 'r', label="ksi_2")
    plt.legend()
    plt.grid()

title = f'theta_{i}={r}'
fig.suptitle(title, fontsize=16)
# plt.savefig("r = 10+-")
plt.show()

#     double frac = pow(r, m) / factorial_(m);
#     double subTerm1 = 0;
#     double subTerm2 = 0;
#     for (int s=(m+1)%2; s<=m+1 && s<n; s+=2){
#     double alpha = alphas_lambda[m][s].call({X, Y, GAMMA, CHI});
#     double beta = betas_lambda[m][s].call({X, Y, GAMMA, CHI});
#     int c_p = 1 + s/(m + 1);
#     int c_m = 1 - s/(m + 1);
#     subTerm1 += theta*((alpha*cos(s-1) + beta*sin(s-1))*c_p + (alpha*cos(s
        +1) + beta*sin(s+1)*c_m));
#     subTerm2 += theta*((-alpha*sin(s-1) + beta*cos(s-1))*c_p + (alpha*sin(s
        +1) - beta*cos(s+1)*c_m));
# //         std::cout << "\nm: " << m << " s: " << s << "\nsubterm1: "
        << subTerm1 << "\nsubterm2: " << subTerm2 << std::endl;
#     }
#     double term1 = frac*subTerm1;
#     double term2 = frac*subTerm2;
# //         std::cout << "\nm: " << m << "\nterm1: " << term1 << "\nterm2:
        " << term2 << "\nksi1: " << ksi1 << "\nksi2: " << ksi2 << std::endl;
#     ksi1 += term1;
#     ksi2 += term2;

```



**Amplitudes\_gen.py**


---

```

import multiprocessing as mp
import sys
import time

from sympy import simplify, symbols, sqrt, diff, factor

# from symengine import diff

n = 50#int(sys.argv[1])

fn = str(n) + '.txt'

def listener(q):
    '''listens for messages on the q, writes to file. '''
    with open(fn, 'w') as f:
        while 1:
            # print(f'Jobs running: {}')
            m = q.get()
            # print("got write job:", m)
            if m == 'kill':
                print("Done")
                break
            f.write(str(m) + '\n')
            f.flush()

def simpl(x):
    return factor(x)

def func(n, m, s, alpha, beta, x, y, g, chi, q):
    print(f'm:_{m}_s:_{s}')# alpha: {alpha} beta: {beta}')
    while s > 0 and m < n:
        m += 1
        s -= 1
        c = ((m + 1.0) / (m + 1.0 - s) * (1.0 + (s != 0.0)) / 2.0) * chi
        # start calculate
        alpha_ = simpl(c * (diff(alpha, x) + diff(beta, y)))
        beta_ = simpl(c * (diff(beta, x) - diff(alpha, y)))

```

```

    alpha, beta = alpha_, beta_
    print(f'm:_{m}_s:_{s}') # alpha: {alpha} beta: {beta} c: {c}')
    # print("gen write job")
    res = f'{m}:{s}:{alpha}:{beta}'
    # print(f'{m}, {s} Done')
    q.put(res)

def main():

    global num_processes

    start = time.time()

    x, y = symbols('x,y', real=True)
    g, chi = symbols("g,c", positive=True, real=True)
    psi = (2*g)/(chi**2) * sqrt(x ** 2 + y ** 2)
    # alphas = [[0 for a in range(n)] for b in range(n)]
    # betas = [[0 for c in range(n)] for d in range(n)]

    #must use Manager queue here, or will not work
    manager = mp.Manager()
    q = manager.Queue()
    with mp.Pool(processes=n) as pool:

        # use a separate process to write to file to avoid
        # ksgjladsfkghld f
        pool.apply_async(listener, (q,))

    jobs = []
    for m in range(0, n+1):
        # print(m)
        s = m + 1
        # print(m, s)
        if m == 0:
            alpha = simpl(-chi * diff(psi, x))
            beta = simpl(-chi * diff(psi, y))
        else:
            c = (m + 1.0) / (m + s + 1.0) * chi
            # calc sym func
            alpha_ = simpl(c * (diff(alpha, x) - diff(beta, y)))
            beta_ = simpl(c * (diff(beta, x) + diff(alpha, y)))
            alpha, beta = alpha_, beta_

```

```
# print(f'{m}, {s} Done')
res = f'{m}:{s}:{alpha}:{beta}'
# print("send write job:", res)
q.put(res)

job = pool.apply_async(func, (n, m, s, alpha, beta, x, y, g,
    chi, q))
jobs.append(job)

# collect results from the workers through the pool result queue
for job in jobs:
    job.get()

#now we are done, kill the listener
q.put('kill')
pool.close()
pool.join()

print(time.time() - start)

if __name__ == "__main__":
    main()
```

---

**job.slurm**

---

```
#!/bin/sh
#SBATCH --partition=GPUQ
#SBATCH --account=ie-idi
#SBATCH --time=96:00:00
#SBATCH --nodes=1
#SBATCH --gres=gpu:8
#SBATCH --constraint="A100"
#SBATCH --job-name="cosmo_anet"
#SBATCH --output=results.out
#SBATCH --mail-user=einarlau@stud.ntnu.no
#SBATCH --mail-type=ALL

WORKDIR=${SLURM_SUBMIT_DIR}
cd ${WORKDIR}
echo "we are running from this directory: $SLURM_SUBMIT_DIR"
echo " the name of the job is: $SLURM_JOB_NAME"
echo "Th job ID is $SLURM_JOB_ID"
echo "The job was run on these nodes: $SLURM_JOB_NODELIST"
echo "Number of nodes: $SLURM_JOB_NUM_NODES"
echo "We are using $SLURM_CPUS_ON_NODE cores"
echo "We are using $SLURM_CPUS_ON_NODE cores per node"
echo "Total of $SLURM_NTASKS cores"

module purge
module load torchvision/0.8.2-fosscuda-2020b-PyTorch-1.7.1
module swap NCCL/2.8.3-CUDA-11.1.1 NCCL/2.8.3-GCCcore-10.2.0-CUDA-11.1.1
module swap PyTorch/1.7.1-fosscuda-2020b PyTorch/1.8.1-fosscuda-2020b
module list
python3 Cosmo_Ai/Python/CosmoCNN.py
```

---



**Linux deployment script**

---

```
#!/bin/sh
echo ""
echo "This script probably requires Ubuntu 18.04 with Qt installed"
echo "Current OS:"
lsb_release -d
echo ""
sleep 2

# Might need to change this path if your build folder is different
buildPath='build-CosmoAI_qt-Desktop-Release'

ldqt='https://github.com/probonopd/linuxdeployqt/releases/download/
      continuous/linuxdeployqt-continuous-x86_64.AppImage'

cd ..

if [ ! -d $buildPath ]
then
    echo "Can not find build folder. Please edit $BASH_SOURCE file with
        correct buildPath"
    exit 0

fi

cd $buildPath || exit

mkdir -p deploy/usr/bin
mkdir -p deploy/usr/share/applications
mkdir -p deploy/usr/share/icons/hicolor/256x256/apps

cd ..
cp CosmoAI_qt/icons/CosmoAI_png.png $buildPath/deploy/usr/share/icons/
  hicolor/256x256/apps
cd $buildPath || exit

echo '[Desktop Entry]
Type=Application
Name=CosmoAI GUI
Comment=CosmoAI GUI Simulator for Linux
Exec=/usr/bin/CosmoAI_qt
Icon=/usr/share/icons/hicolor/256x256/apps/CosmoAI_png
Terminal=false'
```

```
Categories=Graphics;' >deploy/usr/share/applications/CosmoAI_qt.desktop  
cp CosmoAI_qt deploy/usr/bin  
wget $ldqt  
chmod +x linuxdeployqt-continuous-x86_64.AppImage  
  
if ./linuxdeployqt-continuous-x86_64.AppImage deploy/usr/share/  
    applications/CosmoAI_qt.desktop -appimage  
then  
    echo ""  
    echo "Successfully deployed AppImage"  
else  
    echo "ERROR could not get and run linuxdeployqt :("  
fi  
  
rm linuxdeployqt-continuous-x86_64.AppImage
```

---

