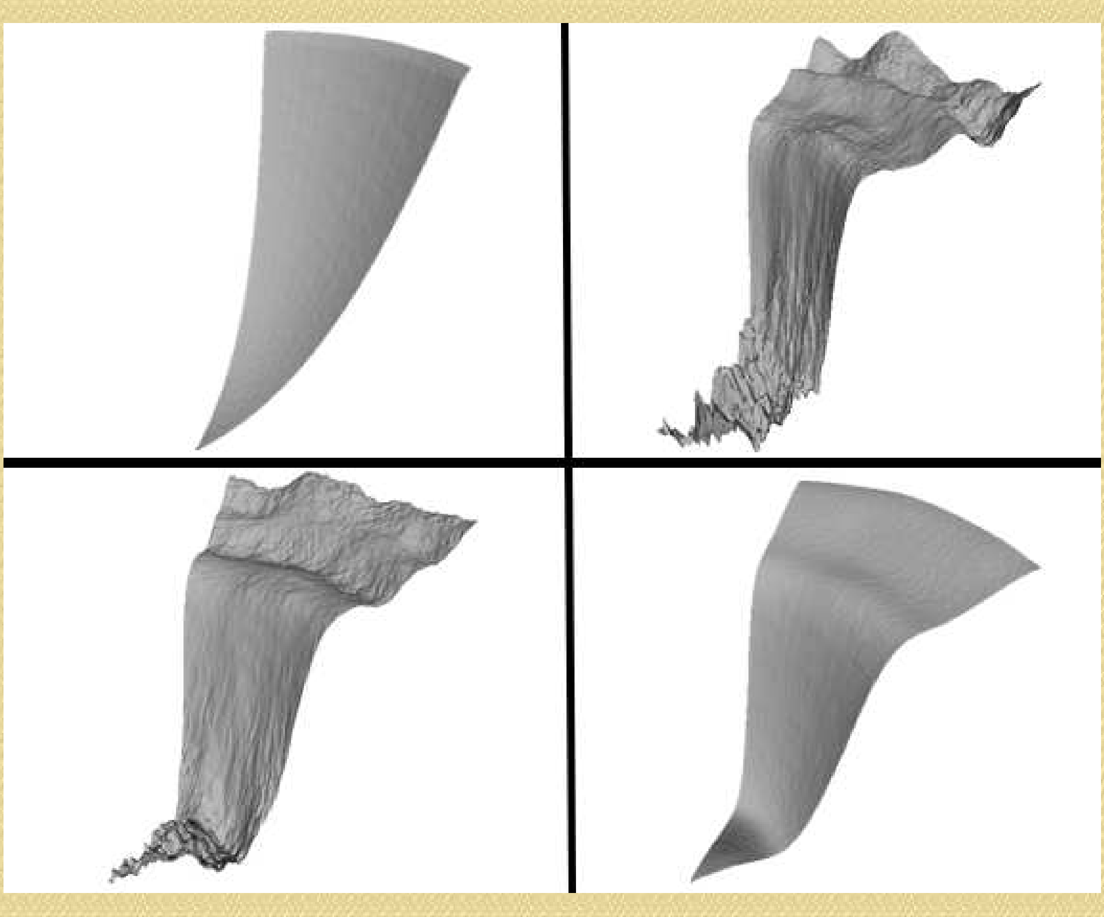


Diogo Filipe Pereira Duarte

Lenses optimization software for LED lighting

Master's Thesis

September 2014

· U · C ·

Universidade de Coimbra

Departamento de Engenharia Electrotécnica e de Computadores
Faculdade de Ciências e Tecnologia
Universidade de Coimbra

A Dissertation
for Graduate Study in MSc Program
Master of Science in Electrical and Computer Engineering

Lenses optimization software for LED lighting

Diogo Filipe Pereira Duarte

Research Developed Under Supervision of
Prof. Dr. António Paulo Mendes Breda Dias Coimbra
Prof. Dr. Manuel Marques Crisóstomo
Prof. Dr. Mateus Daniel Almeida Mendes

Jury
Prof. Dr. Henrique José Almeida da Silva
Prof. Dr. António Paulo Mendes Breda Dias Coimbra
Prof. Dr. Pedro Manuel Gens de Azevedo de Matos Faia

September 2014

Thanks

I want to thank Prof. Dr. António Paulo Mendes Breda Dias Coimbra, Prof. Dr. Manuel Marques Crisóstomo and Prof. Dr. Mateus Daniel Almeida Mendes for all the support, guidance and patience during the preparation of this project.

To all my friends, thank you. This would never have been possible without you.

At last, I want to thank my parents, António Duarte and Ana Duarte, for the endless patience and support during all these years. And most of all I want to thank you for never stop believing in me.

Abstract

This dissertation proposes a framework to automatically optimize the form of a secondary lens of an LED luminaire in order to optimize the light distribution towards a desired goal. An LED used for direct lighting creates a non-uniform illuminance distribution. To solve this problem additional optical devices are needed, so that the light is redistributed and a uniform illuminance distribution is created.

The proposed solution to the problem uses a photometric file to obtain the ideal light distribution. The framework models a secondary lens surface to create the best approximation possible to the ideal light distribution. Three main steps were considered: mesh and lens generation, simulation of light emission from the LED and search for optimized lens surface. The implementation of the framework required the use of Delaunay triangulation algorithm to generate the mesh representing the lens surface. Ray tracing method to simulate the light rays passing through the lens and a genetic algorithm to search for the optimized lens surface. To the best of our knowledge, this is the first time this approach is presented.

The framework was tested using different luminaires, with different light distributions using different configurations of the genetic algorithm and it was able to improve the form of a secondary lens of an LED luminaire.

Keywords: LED, Luminaire, Genetic Algorithm, Ray Tracing, Lenses

Resumo

Nesta dissertação é proposta uma abordagem para automaticamente otimizar a forma de uma lente secundária de uma luminária LED, por forma a otimizar a distribuição luminosa. Quando usado para iluminação direta, a maioria dos LEDs cria um distribuição luminosa não uniforme. Este problema é resolvido usando dispositivos óticos adicionais para que a luz seja redistribuída criando uma distribuição luminosa uniforme.

Utilizando um ficheiro fotométrico, como distribuição luminosa objetivo, a abordagem modela a superfície da lente secundária para que seja criada a melhor aproximação possível à distribuição luminosa pretendida. Foram consideradas três fases para implementar a solução proposta: gerar a malha e a lente, simular a emissão de luz de um LED e procurar a superfície da lente otimizada. Para completar esta implementação foi necessário utilizar a triangulação de Delaunay para gerar a malha, o método de Ray tracing para simular a emissão de luz de um LED e um algoritmo genético para procurar a superfície da lente otimizada. Até onde foi possível averiguar, esta é a primeira vez que esta abordagem é apresentada.

Para testar esta abordagem foram utilizadas diferentes luminárias, com diferentes distribuições luminosas. Foram também utilizadas diferentes configurações no algoritmo genético. Os resultados provaram que a abordagem implementada consegue melhorar a forma da lente secundária de uma luminária LED. Demonstrando que o uso de algoritmos genéticos produz resultados viáveis.

Palavras-chave: LED, Luminária, Algoritmo Genético, Ray Tracing, Lentes

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Nowadays, a big percentage of electricity consumption is spent in artificial lighting so, an inefficient lighting results in wasted resources [1]. Over recent years, the use of light emitting diodes (LED) as light sources in general illumination applications has increased. It is now one of the most promising illumination methods to produce white light [2]. White LEDs are currently used in street lights, vehicle headlamps, spotlights, residential illumination, etc.
Despite the evolution, when using LEDs for direct lighting the patterns shown are of circular symmetry with non-uniform illuminance distribution (Figure 1.1). This creates the necessity to use additional optical devices, such as lenses, to redistribute the light and create a uniform illumination [1, 2].

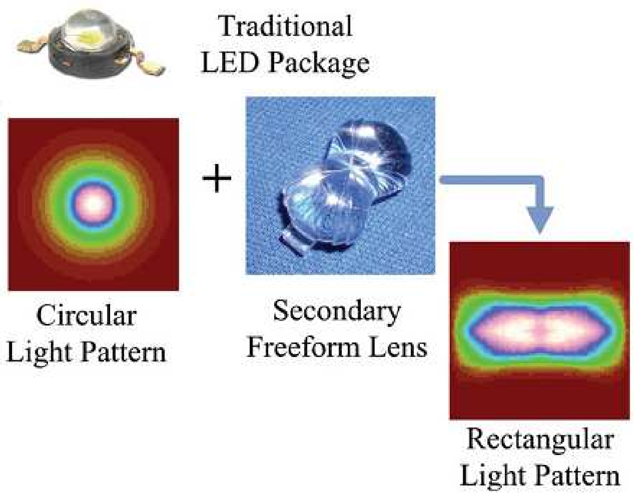


Figure 1.1: Schematic of an LED package [2].

1.1 Motivation and objectives

As stated before and shown in Figure 1.1, an LED used for direct lighting creates a non-uniform illuminance distribution. This problem is solved using secondary optical design, through algorithms of non-imaging optics. The complexity of optical systems can be lowered using free form lenses. Its high degree of freedom can simplify the optical system while still satisfying complex illumination requirements [3].

Commonly, lenses are designed with the help of CAD systems and photometric analysis programs. The designing of each lens is dependent on the experience of the person designing it. Also, it can consume a lot of time to finish, which in the industrial environment, where time is money, is a disadvantage. Considering these points, this dissertation proposes a approach to automatically achieve an improved design.

The objective is to automatically optimize the form of a secondary lens, to use in an LED luminaire, so it has a good light distribution. Using a photometric data file of a luminaire the goal is to design a secondary lens that in conjunction with an LED creates a luminaire with a light distribution similar to the one described in the file. Figure 4.1 shows the polar intensity diagrams of the luminaires used in the tests.

1.2 State of the art

The field of lens optimization is being subject of many researches. From all the works analysed, none of them proposed an approach similar to the one presented in this dissertation. Thus, to the best of our knowledge, it is the first time that this approach is presented. Still, there are some approaches with a few similarities. Some examples are presented below.

Barbosa *et al.* method

Barbosa *et al.* presented at [1] a method to optimize the geometry of an LED secondary lens, focusing the light distribution on a target plane. The geometric parameters defined in the method are:

R_1 , internal surface radius;

R_2 , external surface radius;

L , distance between the light source and the confinement opening;

G_T , added value to the lens total thickness;

β_{R1} , internal lens curvature;

β_{R2} , external lens curvature;

The method uses a ray tracing system to calculate the individual propagation of light rays. The optimization is done by a heuristic process, a genetic algorithm. The genetic algorithm modifies the lens geometry and requests a new simulation from the ray tracing system. The ray tracing returns to the genetic algorithm a vector containing the illuminance distribution curve on the target plane, through an evaluation function, which then analyses the new results of the simulation. This process occurs to achieve an optimized geometry of the lens. The method proposed by Barbosa *et al.* uses a set of geometric parameters to define the lens and optimizes those parameters to achieve the desired light distribution. These parameters impose some restrictions to the lens form, whereas, the method proposed in the present dissertation is able to design freeform lenses.

Liu *et al.* method

Liu *et al.* presented at [3] a method to design an LED freeform lens which produces a uniform circular pattern with high irradiance uniformity and high energy efficiency. According to the authors, three main aspects define the method:

1. Designing the initial guess;
2. Parametrization of the freeform surface;
3. Construction of the merit function;

The initial guess is created by solving an ordinary differential equation. For the parametrization of the freeform surface an approach for selecting the optimization points is introduced. Finally, the merit function is constructed using the irradiance uniformity and the efficiency of the lens. The method can be generalized to design freeform lenses with different lighting patterns or without rotational symmetry.

1.3 Main contributions

The approach proposed in the present dissertation is different from any of the approaches presented in the state of the art (Section 1.2). The method proposed uses photometric files to compare light distributions and the lenses are modelled using a mesh of points instead of defining the lens geometry with a set of parameters or differential equations. Also, whilst other approaches use third parties ray tracing algorithms the approach proposed uses its own ray tracing algorithm. The results obtained are encouraging, the proposed method was capable of automatically optimize a lens and its light distribution accordingly to the goal light distribution defined (Chapter 4).

The approach proposed in this dissertation contains the following contributions:

1. Proof that genetic algorithms can be used to search for the optimized lens surface, achieving viable lenses, which can be manufactured and employed in real applications;

2. Implementation of framework to automatically optimize the form of a secondary lens of an LED luminaire in order to optimize the light distribution;

3. Implementation of an algorithm to achieve ray tracing using data from a photometric file;

4. Implementation of an algorithm to generate a photometric file using the results of the ray tracing algorithm;

1.4 Structure of the thesis

This dissertation is structured along five chapters.

Chapter 1: This chapter introduces the existing problem and states the objectives and the contributions of the dissertation.

Chapter 2: In this chapter all the theoretical foundations and frameworks used in the elaboration of this dissertation are presented.

Chapter 3: This chapter presents the proposed approach to automatically optimize the form of a secondary lens. The approach chosen and the main algorithms developed are presented and described.

Chapter 4: In this chapter the tests made to the proposed framework are presented, as well as the experimental results.

Chapter 5: This chapter presents the conclusions and the future work.

Chapter 2

Theoretical foundations

2.1 Lighting

Lighting is the intentional manipulation of light to produce a desired effect. Lighting is achieved using both natural light sources, by capturing daylight (windows, skylights), and artificial light sources (lamps, light fixtures) [4].
The discovery of fire brought into the world the earliest forms of artificial lighting. Campfires or torches were used to illuminate areas. As the world evolved, new forms of artificial lighting were developed. Lamps made from all sorts of naturally occurring materials, fireflies and candles were used as forms of lighting. The discovery of the electricity and the invention of the incandescent light bulb changed everything and helped to shape the world the way it is today [4].

2.1.1 Lamps

Lamps are a replaceable part of a light fixture. They are commonly referred to as light bulbs and its prime purpose is the production of light using electricity. Lighting technology has come a long way since the incandescent light bulb and presents a variety of different types of lamps with several ways to create light [5].
According to the manner of light emission there are ten main families of lamps [6], as is shown in Figure 2.1.

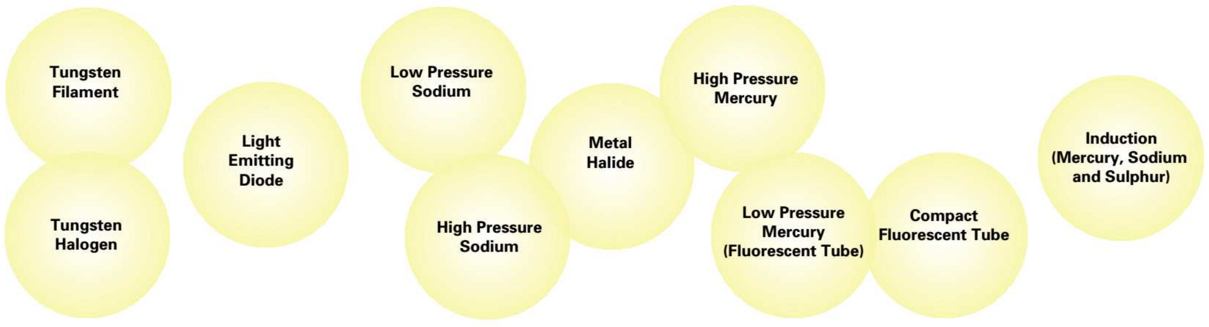


Figure 2.1: Lamp families [6].

2.1.2 LED lamps

Light-emitting diode, commonly known as LED, is a semiconductor light source. When an electrical current flows through it, energy is released in the form of photons. This effect is called electroluminescence. The energy of the photon corresponds to the colour of the light and is determined by the energy band gap of the semiconductor [7].
The earliest LEDs appeared in 1962 and emitted low-intensity infrared light. Later that year appeared the first visible-light LEDs, which were also of low intensity. Nowadays, modern LEDs are available across the ultraviolet, visible and infrared regions of the electromagnetic spectrum with very high brightness [7].

Figure 2.2: XLamp XP-E LED, made by Cree, Inc. [8].

Early LEDs were often used as indicator lamps, however recent developments in LED technology have caused their efficiency and light output to rise exponentially, growing in a way similar to Moore's law. This allowed LEDs to be used in environmental and task lighting, leading to LED lamps, which are sets of LEDs assembled into lamps that are used in light fixtures [7]. Generally lighting needs white light. LEDs can emit white light through two methods. The first method is to use multiple LEDs, mixing red, green and blue light. This allows the overall colour to be changed by adjusting the intensity of each LED. The other method is to use LEDs in conjunction with phosphor, which converts light to other colours [9]. In comparation with other lamp types, LED lamps have several advantages. Their lifespan and electrical efficiency are better than most incandescent and fluorescent lamps. In Table 2.1 it is possible to see a comparison of these three types [9, 10, 11].

Features	**Incandescent**	**Fluorescent**	**LED**
Durability	Fragile	Fragile	Durable
Turns on instantly	Yes	Slight delay	Yes
Frequent on/off cycling	Some effect	Shortens lifespan	No effect
Heat emitted	High	Medium	Low
Frequency of replacement (over 50,000 hours)	4	5	1
Hazardous materials used	None	5mg mercury/bulb	None

Table 2.1: Comparison of features of Incandescent, Fluorescent and LED lamps [10, 11].

2.1.3 Photometry

Photometry is the science of measurement of light, which is defined as the electromagnetic radiation detectable by the human eye. Many different units of measure are used for photometric measurements [12, 13].

2.1.3.1 Luminous flux

Luminous flux (Φ_v), is the measure of the perceived power of light, i.e. the total amount of light a lamp emits. In the International System of Units, SI, the luminous flux is expressed in a unit called lumen (lm) [13, 14].

2.1.3.2 Luminous intensity

Luminous intensity (I_v), quantifies the luminous flux emitted by a source in a certain direction per unit solid angle. The SI unit is the candela (cd) [13, 15].

2.1.3.3 Luminance

Luminance (L_v), is a photometric measure of the luminous intensity of a certain location on a reflecting or emitting surface when viewed from a certain direction, i.e. it describes the amount of light that passes through or is emitted from a particular area and falls within a given solid angle. The SI unit is the candela per square meter (cd/m^2) [13, 16].

2.1.3.4 Illuminance

Illuminance (E_v), describes the luminous flux per area impinging upon a certain location of an irradiated surface. The SI units are the lux (lx) or the lumens per square meter (lm/m^2) [13, 17].

2.1.3.5 Polar intensity diagrams

Polar intensity diagrams illustrate the distribution of luminous intensity, in candelas, for the transverse and axial planes of the luminaire [18].

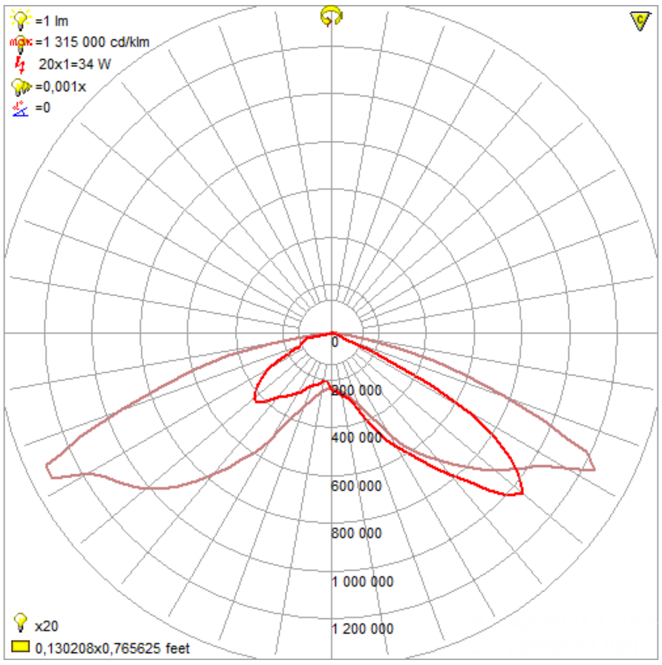


Figure 2.3: Polar intensity diagram.

2.1.3.6 IES

IES or Illuminating Engineering Society was created in 1986 when the Illuminating Engineering Society of North America (IESNA) created the standard LM-63-86, “IES Recommended Standard File Format for Electronic Transfer of Photometric Data.”. This standard has been updated twice, the last time in 2002. The IES format is the most common format in North America but is also widely used in Europe. IES photometric data files have the *.ies* filename extension [18]. IES stores data about the total luminous flux and luminous intensity of a luminaire. With an IES file is possible to create a polar intensity diagram, as shown in Figure 2.3.

2.2 Optics

Generally, optics is the study of any phenomena associated with electromagnetic radiation in the spectral range extending from the long-wave edge of the x-ray region to the short-wave edge of the radio region [19].

The beginning of optics dates back to 700 BC, with the development of lenses by the ancient Assyrians [20]. Ancient Romans and Greeks continued to work on the development of lenses, leading to the establishment of the first theories of light and vision by ancient Greeks and Indian philosophers, and to the development of geometrical optics, in the Greco-Roman world [21]. Was not until the early seventeenth century that the basis of optics were formed. This was due to the discoveries of many experimentalists, e.g., W. Snell with the statement of the law of refraction, C. Huygens and P. de Fermat with the principles of the propagation of light and Isaac Newton with the publication of *Opticks* [19].

In the early nineteenth century the transverse-wave nature of light was established. Later the discovery of the relationship between optical and magnetic phenomena by M. Faraday led to the electromagnetic theory of J. C. Maxwell [19]. Maxwell’s theory served as basis for understanding the interaction of light with matter which provided the basis for treatment of the phenomena of physical optics. The twentieth century brought a dramatic change in physical thinking caused by the theory of relativity and the quantum theory, and optics has been in the vanguard of this revolution. In order to explain many phenomena of physical optics it has been necessary to radically abandon the mechanisms of interaction of radiation and matter [19].

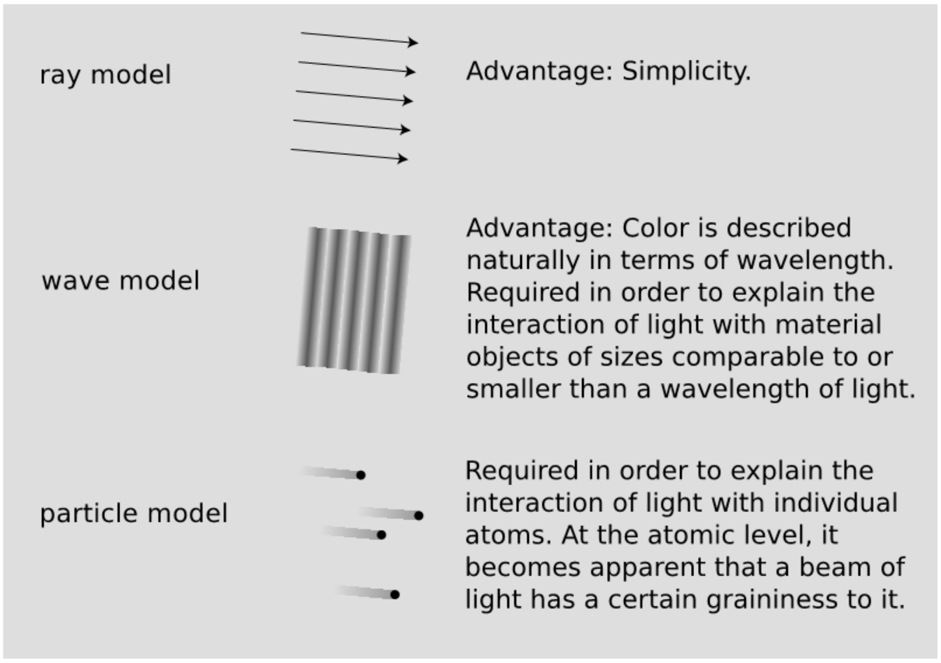

ray model
Advantage: Simplicity.
wave model
Advantage: Color is described naturally in terms of wavelength. Required in order to explain the interaction of light with material objects of sizes comparable to or smaller than a wavelength of light.
particle model
Required in order to explain the interaction of light with individual atoms. At the atomic level, it becomes apparent that a beam of light has a certain graininess to it.

Figure 2.4: Three models of light [22].

Nowadays it is possible to establish three models by which we can describe light, as shown in Figure 2.4: the ray model (geometrical optics), the wave model (classical optics) and the particle model (modern quantum theory) [22].

2.2.1 Geometrical optics

Geometrical optics is the geometry of light rays and their images, through an optical system [19]. It's possible to think of light as rays pointing along the direction of wave propagation. This is useful when the interest is the macroscopic flow of light energy [23].
A closer look to light waves allows a better understanding of geometrical optics. An easy way to visualize waves is to imagine the water waves created by a bobbing cork on a quiet lake. Successive circular waves move radially outward from the cork. Each circular wave represents a wavefront. A wavefront is defined as a locus of points that connect identical wave displacements, as seen in Figure 2.5, and a ray is a line perpendicular to a series of successive wavefronts specifying the direction of energy flow in the wave. With the geometrical construct of a light ray it's possible to describe propagation, reflection and refraction of light [24].

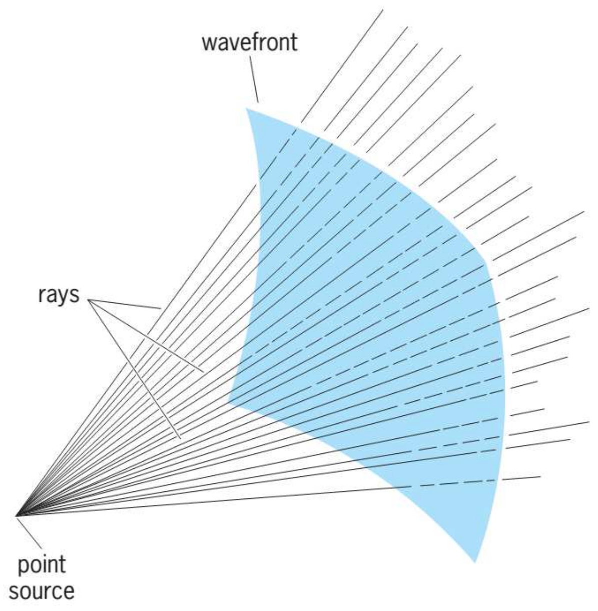


Figure 2.5: Wavefront [19].

Geometrical optics is a robust model and a practical tool to solve optical problems. Some of its applications include analysing laser resonators, solving interference and diffraction problems and even analysing the behaviour of wave guides ,although it would seem incorrect to apply it to this kind of problem. Even so, the principal application of geometrical optics is in the field of optical design [19].

2.2.1.1 Reflection

Reflection is the change of direction of electromagnetic radiation, such as light, by a surface upon which the radiation is incident [19]. Depending on the sort of surface, reflection of light is either specular, as in mirrors, or diffuse, as in white marble.
The law of reflection describes specular reflection. It states that the incident ray, the reflected ray and the normal to the reflecting surface all lie in one plane. And that the angle between the incident ray and the normal is equal to the angle between the reflected ray and the normal, as shown in Figure 2.6.

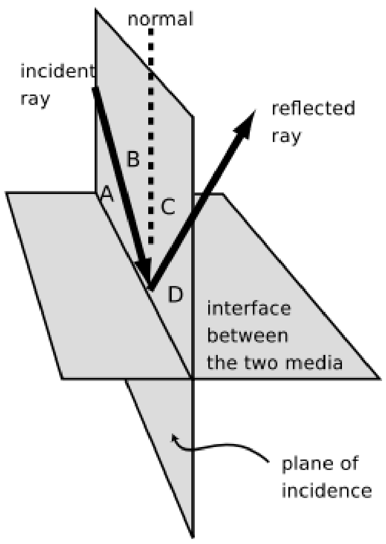


Figure 2.6: Geometry of specular reflection [22].

Opposing to specular reflection where an incident ray is reflected at only one angle, in diffuse reflection an incident ray is reflected in many angles.

2.2.1.2 Refraction

Refraction is the change of direction of a wave at an interface between two optical media.
An optical media can be described in terms of index of refraction, which is defined as the ratio between the speed of light in a vacuum and the speed of light in the medium. Air and most gases have an index of refraction nearly equal to one, while other materials have values greater than one [24].

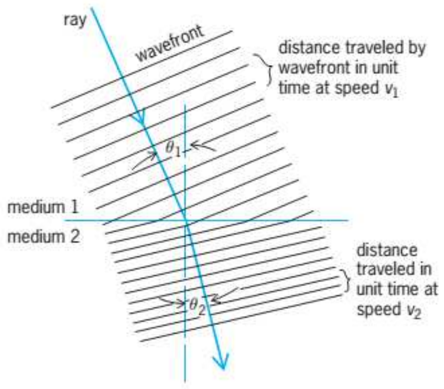

ray
wavefront
distance traveled by wavefront in unit time at speed v_1
θ_1
medium 1
medium 2
distance traveled in unit time at speed v_2
θ_2

Figure 2.7: Physical basis for refraction [19].

Figure 2.7 illustrates the refraction phenomenon, where a wave, propagating in the optical medium 1, approaches the boundary surface obliquely. The velocity of propagation of medium 2 is slower than in medium 1, which forces waves to slow down as they enter medium 2. This makes the direction of travel to bend towards the perpendicular of the surface boundary. By the contrary if the velocity of medium 2 is greater than that of medium 1 the direction of travel would bent outwards [19].
In the seventeenth century Willebrord Snell formulated a law, known as Snell's law, which determines the direction of a refracted ray by knowing the indexes of refraction of the two media (n_r and n_i) and the direction of the incident ray. Snell's law is stated in equation 2.1. Both the incident ray, the normal to the boundary surface and the refracted ray always lie in the same geometrical plane, and the angles of incidence θ_i and refraction θ_r are measured with respect to the surface normal [24].

$$\frac{\sin\theta_i}{\sin\theta_r} = \frac{n_r}{n_i} \tag{2.1}$$

2.3 Delaunay triangulation

Delaunay triangulation is named after Boris Delaunay for his work on the subject from 1934. Delaunay triangulation is a technique that has been applied in a variety of areas such as computational geometry, metallurgy, virtual reality, solid modelling and many others [25].

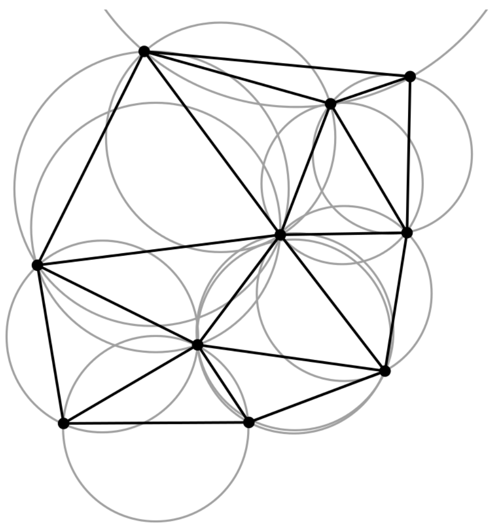


Figure 2.8: Circumcircles of a Delaunay triangulation [26].

Delaunay triangulation is used for connecting a set of points in a space into triangles in such way that no point in the set of points is inside the circumcircle of any triangle from the Delaunay triangulation of the set of points [27, 28]. The Delaunay triangulation seeks to avoid long thin triangles by maximizing the minimum angle of all the angles of the triangles in the triangulation [29]. All mathematical formulation is purposely omitted due to the complexity of the subject.

The Delaunay triangulation was used in the framework developed to generate a mesh that represents a surface of a lens. An implementation of this method is available in OpenCV and that implementation was used in the framework.

OpenCV is an open source computer vision library. It is a cross-platform library written in C and C++ [29]. The OpenCV project started at an Intel Research initiative in 1998. Since then the library became one of the most used tools in computer vision industry as well as in university courses and research projects [30].

OpenCV was designed for computational efficiency and with a strong focus on real time applications [29]. The library has more than 2500 optimized algorithms that can be used to detect and recognize faces, identify objects, extract 3D models of objects, create a Delaunay triangulation of a set of points and many other applications [31].

2.4 Ray tracing

Ray tracing is an illumination algorithm for generating synthetic images [32]. Due to its capacity of achieving a high degree of photorealism it's used in animation systems and in a variety of high-end 3D rendering applications, e.g., medical visualisation systems [33].

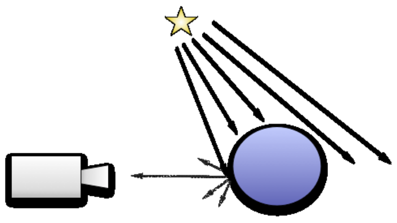


Figure 2.9: Forward ray tracing scheme [34].

Ray tracing algorithms simulate the physical behaviour of light to generate the images. It takes into account phenomena such as reflection and refraction [32, 33]. The common utilization of the ray tracing algorithm is the backward ray tracing, where rays are cast from the view point and followed throughout the scene towards a light source [32]. The other method is the forward method (figure 2.9), where rays are cast from the light source [35].

Although ray tracing algorithms have tremendous potential they also have some disadvantages. The "super-realistic" images produced is one of them, in the real world scenes are rarely as perfect as in those images. A lot of research has been done in order to enhance ray tracing and make it more realistic [33]. Another disadvantage of ray tracing is the computational cost of the algorithm, as it is necessary to calculate all intersections of every ray with the primitives in the scene in order to determine the pixel colour [32].

Even though, ray tracing has the versatility to include numerous optimizations, due to its uncomplicated algorithm.

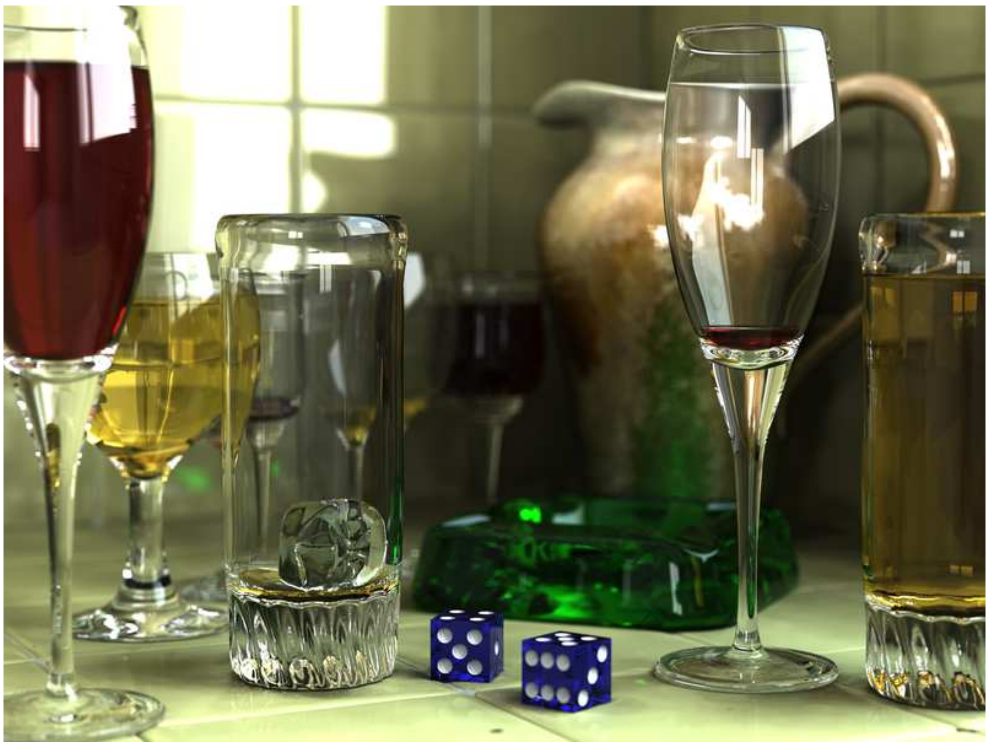


Figure 2.10: Super-realistic image obtained with ray tracing [36].

2.4.1 Computation of intersections

The major drawback of ray tracing algorithms is the computation of the intersections of the rays with the objects in the scene. So, it's extremely important to use efficient algorithms. Computations must be reduced to the minimum and they must be as simple as possible. For this reason it is usual to restrict the intersections of rays with objects in the scene to the intersections of straight lines with simple geometric shapes [37].

2.4.1.1 Rays

A ray is a semi-straight line with origin in a point, R_0, and a given direction, $\vec{R_d}$.

$$R_0 = (x_0, y_0, z_0) \tag{2.2}$$

$$\vec{R_d} = (x_d, y_d, z_d) \tag{2.3}$$

The ray is described as:

$$\vec{R}(t) = R_0 + \vec{R_d} * t \tag{2.4}$$

where t is the parametric description parameter of the straight line. If $t < 0$ the point is behind the projection centre and doesn't belong to the ray, else if $0 \leqslant t \leqslant 1$ the point is between the projection centre and the projection plane, else if $t > 1$ the point is beyond the projection plane [37].

2.4.1.2 Intersection of a ray and a plane

The linear equation of a plane P is:

$$Ax + By + Cz + D = 0 \tag{2.5}$$

where $A^2 + B^2 + C^2 = 1$, and the unit normal vector is:

$$\vec{P_N} = (A, B, C) \tag{2.6}$$

To find the intersection of $\vec{R}$ and P, it is necessary to substitute (2.4) in (2.5):

$$A(x_0 + x_d t) + B(y_0 + y_d t) + C(z_0 + z_d t) + D = 0 \tag{2.7}$$

Then, solving for t:

$$t_i = \frac{-(Ax_0 + By_0 + Cz_0 + D)}{Ax_d + By_d + Cz_d} \tag{2.8}$$

or:

$$t_i = -\frac{\vec{P_N} \bullet \vec{R_0} + D}{\vec{P_N} \bullet \vec{R_d}} \tag{2.9}$$

2.4.2 Reflection

Consider Figure 2.11. The incident ray, $\vec{I}$, is reflected at the intersection point and results in the reflection ray, $\vec{R}$. The incidence ray and the surface normal, $\vec{N}$, create an incidence angle, θ_i, also the reflection ray and the surface normal create a reflection angle, θ_r. In perfect specular reflection the incidence angle and the reflection angle are equal [33, 37].

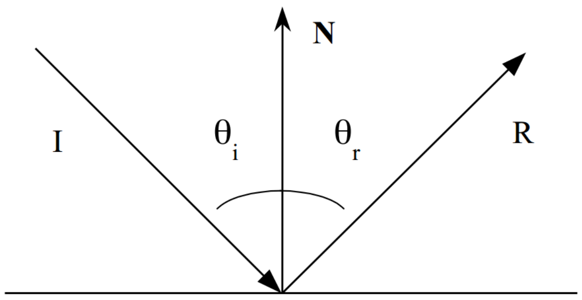


Figure 2.11: Perfect specular reflection [37].

The reflection ray is defined as:

$$\vec{R} = \alpha\vec{I} + \beta\vec{N} \tag{2.10}$$

where both $\vec{I}$ and $\vec{N}$ are normalized, and $\vec{R}$, $\vec{I}$ and $\vec{N}$ are all planar. This way:

$$\cos\theta_i = -\vec{I} \bullet \vec{N} \tag{2.11}$$

$$\cos\theta_r = \vec{R} \bullet \vec{N} \tag{2.12}$$

and as θ_i is equal to θ_r:

$$\begin{aligned} \cos\theta_i &= \cos\theta_r \\ \Leftrightarrow\ -\vec{I} \bullet \vec{N} &= \vec{R} \bullet \vec{N} \\ \Leftrightarrow\ -\vec{I} \bullet \vec{N} &= \alpha(\vec{N} \bullet \vec{I}) + \beta \end{aligned} \tag{2.13}$$

Let α be 1, then $\beta = -2(\vec{N} \bullet \vec{I})$ and considering (2.10) results:

$$\vec{R} = \vec{I} - 2(\vec{N} \bullet \vec{I})\vec{N} \tag{2.14}$$

2.4.3 Refraction

As a ray goes through an interface between two media its propagation direction is changed. As Figure 2.12 shows, the angle, θ_i, formed between the incident ray, $\vec{I}$, and the surface normal, $\vec{N}$, is different from the angle, θ_t, formed between the surface normal and the transmission ray, $\vec{T}$.

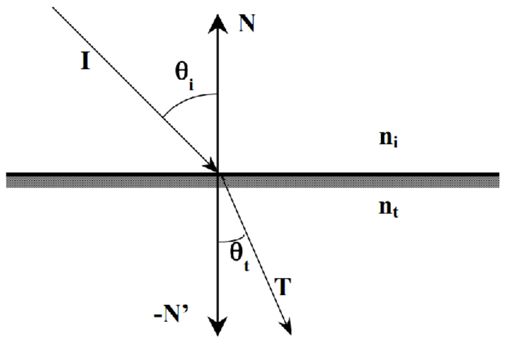


Figure 2.12: Refraction [37].

Snell's law establishes the relation between the angles and the refraction indexes (n):

$$\frac{\sin\theta_i}{\sin\theta_t} = n_{ti} = \frac{n_t}{n_i} \tag{2.15}$$

where n_i is the index of refraction of the medium of incidence, n_t is the index of refraction of the medium of transmission and n_{ti} is the index of refraction of the medium of transmission in respect to the medium of incidence. Each medium has its own index of refraction.
As Figure 2.12 shows, the incidence vector, $\vec{I}$, bends at the intersection point between the incidence medium and the transmission medium, resulting in the transmission vector $\vec{T}$, which can be defined as:

$$\vec{T} = \alpha\vec{I} + \beta\vec{N} \tag{2.16}$$

where $\vec{N}$ is the normal vector of the surface in the intersection point. Being:

$$\cos\theta_i = \vec{N} \bullet -\vec{I} \tag{2.17}$$ $$\cos\theta_r = -\vec{N} \bullet \vec{T} \tag{2.18}$$

Applying Snell's equation and squaring it results in:

$$\sin^2\theta_i n_{it}^2 = \sin^2\theta_t \tag{2.19}$$

where $n_{it} = \frac{n_i}{n_t}$. Using the Pythagorean trigonometric identity[1]:

$$(1 - \cos^2\theta_i)n_{it}^2 = 1 - \cos^2\theta_t \tag{2.20}$$

which can be written as:

$$(1 - (-\vec{N} \bullet \vec{I})^2)n_{it}^2 = -(\alpha(-\vec{N} \bullet \vec{I}) - \beta)^2 \tag{2.21}$$

All vectors are normalized, so the final transmission vector $\vec{T}$ can be written as:

$$\vec{T} = n_{it}\vec{I} + (n_{it}(-\vec{N} \bullet \vec{I}) - \sqrt{1 + n_{it}^2((-\vec{N} \bullet \vec{I})^2 - 1)})\vec{N} \tag{2.22}$$

If the root term isn't positive there is no transmission ray and there is total internal reflection.

2.5 Genetic algorithm

Genetic algorithm is a computational model inspired in the evolution theory, in which the process of natural selection is simulated [38].

Science emerged from the human desire to understand and control the environment around him. Driven by this desire, in the 1960s, John Holland invented the genetic algorithm and jointly with his students developed it through the 1970s [39]. Holland's genetic algorithm is a method to move from one population of individuals to a new one by using a natural selection process along with genetic operators (e.g., mutation, crossover) [40].

In a looser interpretation, a genetic algorithm is any model that generates new solutions in a search space by applying a set of selection and recombination operators to a population of

[1]Defined as: $\cos^2\theta + \sin^2\theta = 1$

individuals [38].
Nowadays many computational problems require searching through countless possibilities, so genetic algorithms are applied in fields such as engineering, economics, computational science, mathematics and many others.

2.5.1 Methodology

In genetic algorithms the evolution is an iterative process with the initial population usually being randomly generated. In each iteration the population is called a generation, and in each generation the fitness of each individual is evaluated. The fittest individuals in the population are selected and possibly modified in order to form a new generation. This new generation is then used in the next iteration. It is usual for the algorithm to stop either when a maximum number of generations is reached or when an individual reaches a certain fitness.

Initialization

Regardless of the problem nature it's common that the individuals from the initial population are randomly generated. Even so, the nature of the problem dictates the population size (bigger or smaller), as well as the individuals encoding (bit strings, floating point).

Selection

As previously stated, in each generation the best individuals in the population are selected to breed the new generation. This selection is based on the fitness of each individual. The fittest the individual, the higher is the probability of him being selected.
To calculate the fitness of each individual it is necessary a fitness function, which depends on the problem's nature. This function measures the quality of the individual (solution) in relation to the problem.

Operators

A set of genetic operators are applied to the selected individuals in order to create the new generation. The simplest operators are the single point crossover and the mutation.
The single point crossover operator randomly selects a point in the individual and exchanges the subsequences before and after the point between two individuals to create two offsprings.
The mutation operator randomly changes some points in the individual.

Termination

The evolution process can continue endlessly: in order to stop it is necessary to set some conditions. Some common termination conditions are a maximum number of generations and a minimum desired fitness, or a combination of both.

2.5.2 Open BEAGLE

Open BEAGLE is a versatile C++ evolutionary computation framework. The name BEAGLE stands for the acronym *the Beagle Engine is an Advanced Genetic Learning Environment*. In the 1980s the name Beagle was used for a pattern recognition software developed by Forsyth, and in order to distinguish from Forsyth's software the adjective Open was added to the name of the framework. The adjective also stresses the open source aspect of the project [40].
The Open BEAGLE architecture is based on object oriented programming, thus allowing the code to be easily reused and to represent some abstractions as loosely objects. Open BEAGLE allows the user to implement his own algorithms with minimal code writing.

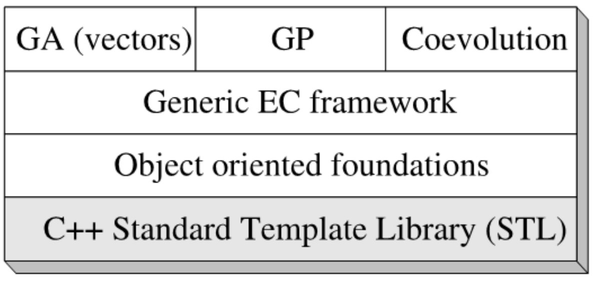

GA (vectors)
GP
Coevolution
Generic EC framework
Object oriented foundations
C++ Standard Template Library (STL)

Figure 2.13: Open BEAGLE framework architecture [41].

As figure 2.13 shows, Open BEAGLE architecture is divided into three different levels. C++ and the Standard Template Library (STL) form the basis for the object oriented foundations, while the generic framework is built on those foundations and is composed of elements from all types of evolutionary computation. At the top, different modules specialize each evolutionary computation flavour [40, 41].

Genericity criteria	ECJ 13	EO 0.9.3a	GAlib 2.4.6	lil-gp 1.1	GPLAB 2	Open BEAGLE 2.2.0
Generic representation	2	2	2	0	0	2
Generic fitness	2	2	0	0	0	2
Generic operations	2	2	1	2	2	2
Generic evolutionary model	2	2	1	1	1	2
Parameter management	2	2	2	1	2	2
Configurable output	2	1	0	1	0	2

Table 2.2: Comparison of features of different evolutionary computation software libraries [41]. (2 = complete; 1 = partial; 0 = missing)

A great variety of evolutionary computation software are available but not all provide the necessary tools. When compared to other evolutionary computation software, Open BEAGLE is better than the most. Table 2.2 compares six different evolutionary computation software libraries: it is possible to observe that ECJ and Open BEAGLE are the best tools. The decision to use Open BEAGLE rather then ECJ was due to the programming language in which each one is coded in: while Open BEAGLE is coded in C++, ECJ is coded in Java.

Chapter 3

Development

Nowadays software tools are everywhere, working both explicitly and behind the scenes in virtually all aspects of modern lives [42].
The problem is quite complex and the need to build the entire framework from scratch revealed itself to be a challenge. The path to find a solution to the problem was to analyse it, breaking it into smaller problems and dealing with them [42]. The solution was then, a composition of the solutions to the various smaller problems.

3.1 Problem analysis

The main goal was to create a software capable of optimizing a lens. The idea to achieve this goal, as presented in Figure 3.1, was to write a software that given a set of parameters (e.g., the lens radius, the desired light distribution), would be capable of finding the best possible lens. In the scope of this particular problem, to optimize a lens means to find such a shape that when the lens is attached to an LED the resulting light distribution is as close as possible to the one set as goal. With the increase of knowledge of the problem it became clear that it was possible to divide its resolution in four steps: the representation of a lens, the light distribution of a light source, the simulation of light passing through a lens and the optimization process.

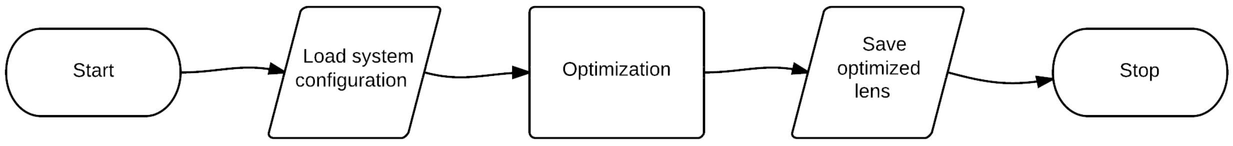


Figure 3.1: Flowchart of the lenses optimization software.

3.1.1 Representation of a lens

The first issue to be solved was how to represent a lens. This question has two inherent sub-questions: how to represent each surface of the lens and how to represent the lens as a whole.
To represent each surface of the lens the choice was to create an approximation of the surface using a set of triangles (mesh). Given the usual circular or elliptical form of the lens, the points are generated in annuli, trying to guarantee that nearby points are equally spaced. For the triangulation the method chosen was the Delaunay triangulation (Section 2.3). The distribution of the points together with the properties of the Delaunay triangulation assure that the triangles are as equilateral as possible. This is important because the size of the triangles is directly proportional to the error in the shape approximation and the smaller the triangles the smaller the error.
Next it was necessary to represent the lens as a whole. The solution was to model the lens as three separated surfaces, the top surface, the bottom surface and the lateral surface. The representation of both top and bottom surfaces of the lens is solved, to represent the lateral surface the solution chosen uses the external annulus of the top surface and of the bottom surface and connects the points in groups of four, creating rectangles which form the lateral surface of the lens.

3.1.2 Light distribution of a light source

For the software to achieve its purpose it is necessary to use the light distribution of lamps and lamp fixtures. This light distribution is already available in some formats, namely, IES (Subsection 2.1.3.6), EULUMDAT[1] and CIBSE[2]. The format chosen was the IES because it is the most common format in North America and widely used in Europe as well [18].

[1]EULUMDAT is the main format used in Europe.

[2]CIBSE is a format used primarily in Great Britain.

3.1.3 Simulation of lighting

The simulation of lighting was achieved by using the ray tracing method (Section 2.4). This technique simulates the light passing throughout the lens and striking the ground, enabling the calculation of the light distribution of the LED together with the lens.

3.1.4 Optimization process

For the optimization process it was chosen the genetic algorithm (Section 2.5). The optimization of the lenses is done by searching for a shape that, as close as possible, creates the desired light distribution. The search space where the answer is looked for is huge and even with a good answer there is no guaranty it will be the only or the best solution. The optimal solution may never be found but with a genetic algorithm it is possible to obtain an approximate solution, which can work just as well in the problem context.

3.2 Software development

The software development was rather complex. All the software was written in C++ using the object-oriented features of this programming language.

3.2.1 Mesh and lens generation

The development started with the creation of the mesh to represent the lens surface. Using the idea described in (Subection 3.1.1) the mesh was created using a set of points distributed in annuli, and subsequently triangulated using the Delaunay triangulation function provided by OpenCV. This required the definition of the lens radius, the aspect ratio and the distance between points. The point generation is presented in (Algorithm 1). The number of annulus (layers) is calculated by:

$$NLayers = \lceil \frac{radius}{points\ distance} \rceil \tag{3.1}$$

this way the distance between layers is as close as possible to the distance between points. The points are generated around the origin of the Cartesian system and the last point added to the list of points is the point $(0, 0)$, which is the origin of the Cartesian system and the centre of the lens.

Algorithm 1: Generate points.

```
calculate number of layers;
for each layer do
    set number of points;
    for each point do
        calculate the X coordinate;
        calculate the Y coordinate;
        save point in list of points;
    end
end
set X coordinate to 0;
set Y coordinate to 0;
save point in list of points;
```

After generating the points it was necessary to triangulate them. The Open CV has a function to apply the Delaunay triangulation to a set of points. Algorithm 2 presents the algorithm to do the triangulation of the generated points and save the resulting triangles.

Algorithm 2: Apply triangulation and to get the triangles.

```
set the Delaunay subdivision;
for each point of the list of points do
    insert point into the Delaunay subdivision;
end
get triangles list;
for each triangle of the list of triangles do
    if vertices inside the lens then
        call findIndexes with triangle vertices;
        insert triangle into triangle list;
    end
end
```

After the triangulation was done it was necessary to retrieve the triangles. The triangles returned by the Open CV function had the coordinates of the points in the vertices but the goal was to have the index of the point in the point list, rather than the point coordinates. For this purpose it was necessary to create Algorithm 3, to make the desired conversion.

Even though it is not mentioned in the algorithms, the Open CV has a bug that caused some problems in the development process. When the coordinates of the points used to do the triangulation are of the order of microns the function gives an error. To avoid this, the coordinates of the points were multiplied by a constant when inserted into the subdivision. A second found difficulty had to be solved: the function doesn't have a great precision so the triangulation function did some rounding to the coordinates of the points. When doing the conversion from point coordinates to point index it was necessary to divide the coordinate by the same constant it had been multiplied before and it was necessary to search in a range around the point coordinate, instead of just the exact coordinate.

Algorithm 3: Find the indices of the triangle vertices.

```
for each vertex of the triangle do
    read the coordinates from the vertex;
    for each point in the point list do
        if point coordinates matches to vertex 1 coordinates then
            set triangle vertex 1 to current point index;
        end else if point coordinates matches to vertex 2 coordinates then
            set triangle vertex 2 to current point index;
        end else if point coordinates matches to vertex 3 coordinates then
            set triangle vertex 3 to current point index;
        end
    end
end
return triangle vertices
```

After having the lens representation of the top and bottom surfaces it was necessary to calculate each triangle plane equation, the method used is described in [43]. As stated in Subsection 3.1.1, the lateral surface of the lens is composed by rectangles formed by four points, where two points belong to the top surface and the other two points are the corresponding points in the bottom surface. The points are then saved and three of those points are used to determine the plane equation.

3.2.2 IES manipulation

As stated in Subsection 3.1.2, the light distribution information is contained in IES files. Due to the lack of libraries to handle this type of files it was necessary to develop a variety of functions to handle these files. The IES format has a standard, and using the rules of that standard it was possible to create a function that would load all the data from the file and a function that would write a valid IES file.
The IES files have the light intensity data associated to a pair of angles (vertical and horizontal) and it is usual for each group of angles, vertical or horizontal, to have a constant resolution. The framework required that IES files were compared, in other words, it was necessary to compare the light intensity in two different files, returning the difference between them. This comparison is only meaningful if the light intensity in the two files is given by the same pairs of angles, if this

condition is met the function returns the total absolute error between the light intensity in the two files. For both files to have the same groups of angles it was created a function that would change those groups of angles resolutions.
The Algorithm 4 describes the process used to do this.

Algorithm 4: Change IES angles resolution.

```
read horizontal angles and vertical angles;
set nHorizontal as number of horizontal angles;
set nVertical as number of vertical angles;
read candela values;
read new horizontal anglesa and new vertical angles;
for each new horizontal angle do
    while new horizontal angle > horizontal angle do
        move to next horizontal angle;
        move nVertical to next candela value;
    end
    set currentCandela as current candela value;
    for each new vertical angle do
        set current candela value to currentCandela;
        while new vertical angle > vertical angle do
            move to next vertical angle;
            move to next candela value;
        end
        if new horizontal angle equal to horizontal angle and new vertical angle equal to vertical angle then
            insert candela value into candela values list;
        else if new horizontal angle equal to horizontal angle and new vertical angle less than vertical angle then
            interpolate the candela value using the last and the next candela values;
            insert candela value into candela values list;
        else if new horizontal angle less than horizontal angle and new vertical angle equal to vertical angle then
            interpolate the candela value using the corresponding last and the corresponding next candela values;
            insert candela value into candela values list;
        else if new horizontal angle less than horizontal angle and new vertical angle less than vertical angle then
            interpolate the candela value using the last and the next candela values;
            interpolate the candela value using the corresponding last and the corresponding next candela values;
            interpolate the candela value using the two last interpolations;
            insert candela value into candela values list;
    end
end
```

The interpolation method used was the cosine interpolation. When compared to other interpolations methods this was the best combination of simplicity and performance, as it is possible to see in Figure 3.2. The candela values referred in the algorithm are the light intensity (Subsection 2.1.3.2) values.

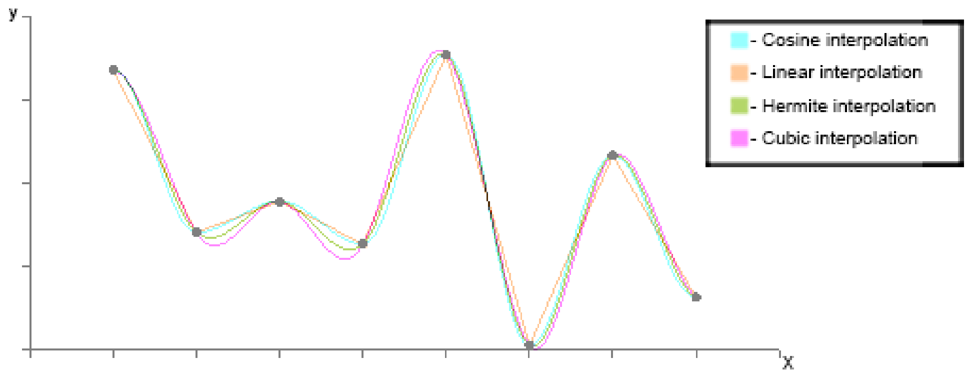


Figure 3.2: Comparison of different interpolation methods [44].

3.2.3 Ray tracing

To evaluate a lens it was necessary to simulate the light passing through the lens. The method chosen was based on the ray tracing algorithm (Section 2.4). Algorithm 5 describes the process to generate these rays.

Algorithm 5: Generate light rays.

```
read horizontal angles and vertical angles;
read candela values;
set rays origin point;
for each horizontal angle do
    for each vertical angle do
        calculate the direction X coordinate;
        calculate the direction Y coordinate;
        calculate the direction Z coordinate;
        save the ray direction;
        save the ray origin;
        save the ray intensity;
        insert ray into rays list;
        move to next candela value;
    end
end
```

Each ray generated, has a direction vector and an origin point, being the origin point the same of the position of the light source. The direction vector is calculated using a polar to cartesian conversion (3.2).

$$\begin{cases} x = \cos(\theta)\cos(\phi) \\ y = \cos(\theta)\sin(\phi) \\ z = \sin(\theta) \end{cases} \tag{3.2}$$

The Algorithm 6 implements a forward ray tracing method, where the rays depart from the light source, located above the lens, and pass through the lens. Due to the heavy computation associated to the algorithm it was decided to consider that the lens top surface had the shape of a half-sphere with the light source located at its centre, so that the rays don't suffer any kind of change in direction when passing through it. Therefore this surface is ignored for calculation.

Algorithm 6: Ray tracing.

```
read list of rays;
for each ray in the list of rays do
    set NReflections to 0;
    while ray didn't hit the ground and NReflections <10 do
        search for intersection;
        increment NReflections;
    end
end
```

When a ray is inside the lens it is verified if the ray intersects the bottom surface or the lateral surface. In both cases when the intersection is found the refraction or the reflection of the ray is calculated. In case reflection exists, the ray the algorithm continues to follow the ray until it exits the lens: additionally if the ray is reflected more than 10 times it will be ignored.
Algorithm 7 describes the method used to search for the intersection of a ray with the lens.

Algorithm 7: Search for the intersection of a ray with the lens surfaces.

```
for each triangle plane equation in the list of triangle plane equations of the bottom surface do
    if ray intersects bottom surface then
        if intersection point inside any of the triangles then
            set continue to 1;
end
if no intersection with bottom surface is found then
    call method findLateralIntersection;
if continue equal to 1 then
    calculate ray refraction;
    if refraction exists then
        calculate point of intersection of the ray with the ground;
    else
        calculate ray reflection;
```

The method to search for the intersections of rays with the lateral surface has some similarities with the method to search for intersections with the bottom surface and is described in Algorithm 8.

Algorithm 8: Search for the intersection of a ray with the lateral surface of the lens.

```
for each rectangle plane equation in the list of rectangle plane equations of the lateral surface do
    if ray intersects lateral surface then
        divide the rectangle into two identical triangles;
        if intersection point inside any of the triangles then
            calculate ray refraction;
            if refraction exists then
            calculate point of intersection of the ray with the ground;
            else
                calculate ray reflection;
end
```

3.2.4 Genetic algorithm

The genetic algorithm is an important component of the software, were all the components developed before interact to achieve the main objective. The genetic algorithm was implemented using the Open BEAGLE library (Subsection 2.5.2).
The main pieces of the genetic algorithm were the evaluation operator and the evolution strategy. To develop the evaluation operator it was decided that the optimization of the lens would be done by altering the form of the bottom surface. This way the genes are the Z coordinates of the points from the lens bottom surface, with X and Y coordinates being static, and only the Z coordinate is changed in order to achieve a shape that satisfies the requirements. To satisfy the requirements means that the light distribution resulting from the lens together with the light source is as close as possible to a light distribution provided as goal.

Algorithm 9: Evaluation operator.

```
read individual;
apply gaussian smoothing to the individual;
call generate rays method;
call ray tracing method;
call light distribution method;
create individual IES file;
call compare method with individual IES file, objective IES file;
set Fitness to $\frac{1}{1 + compare\ result}$;
return Fitness
```

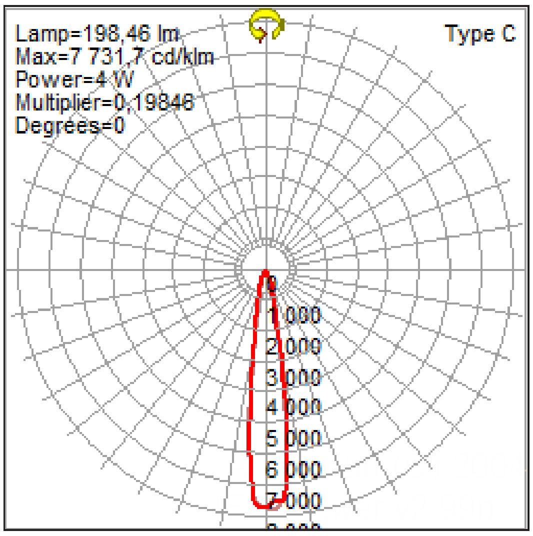
During the software test phase a problem became clear: the lenses generated by the algorithm had a too irregular surface, so it was necessary to introduce a gaussian smoothing filter to smooth the lens surface into the process. The evaluation operator developed is presented in Algorithm 9.

After the ray tracing was done it was necessary to compare the resulting light distribution with the desired light distribution. In order to compare the two IES files they must have the light intensity given by the same pairs of angles (Algorithm 4).
Each ray has an horizontal and a vertical angle associated. The algorithm to get the light distribution starts by looking for the horizontal angle in the list of horizontal angles, and a certain distance is attributed to each angle of the list (e.g., any angle from 2.5^o to 7.5^o is attributed to the angle 5^o): if the ray horizontal angle belongs to any of the distances, that is the horizontal angle used when adding the light intensity value.
After the algorithm searches for the vertical angle in the list of vertical angles. This search follows the same logic as the previous, however, the distance is smaller in order to create a border zone. In this border zone a fuzzy logic is applied and, if the ray vertical angle is inside the border zone between two vertical angles of the list, a certain weight is given to the border limits, if the angle is closer to the left border angle it will have a weight greater than the right border angle, and vice versa. So, instead of adding the total light intensity value of the ray to one pair of angles, the value is divided by the two pairs.
After this distribution of the rays light intensity by the angles the list of intensity values is searched in order to look for angles that could have been wrongly left with intensity zero. If any angles are found, a cosine interpolation method is applied. The values given by the interpolation method are good approximations due to the continuous nature of the light distribution.
A special case needed to be considered. The vertical angle 0^o has the same intensity regardless of the horizontal angle. So, when generating the rays only one ray is generated with a vertical angle of 0^o and in this last step the intensity of that ray is added to all the pairs of angles with vertical angle 0^o.
At last the IES file is created and the two files are compared. The comparison returns the total absolute error that is then normalized so that the fitness value is in the distance $[0, 1]$.
After the evaluation operator was developed the evolution strategy needed to be defined. The Open BEAGLE provides a set of operators that can be used in the evolution, like mutation and crossover operators, termination operators and some others. The Open BEAGLE as a feature that allows the evolutionary system to be configured using a configuration file. This file is formatted in XML and it can be used to set or modify parameters and to configure the evolution strategy.

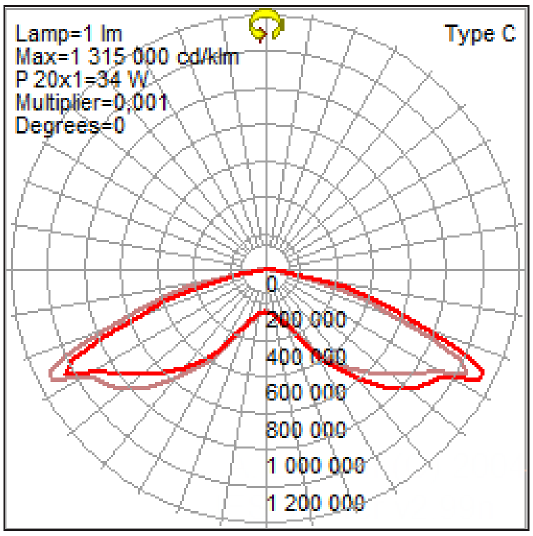
Chapter 4

Tests and Results

Throughout the development of the software many tests were made to the software components. However, these tests were only to ensure the validation of the individual components, rather than the validity of the software. This chapter describes the tests made to validate the correct functioning of the software and of the approach used. In addition other tests were conducted to improve the software and find the genetic algorithms parameters best configuration.



(a) Luminaire LED Energy Smart^{TM} MR16, manufactured by General Electric.



(b) Luminaire HomeMadeObjective, not commercialized.

Figure 4.1: Polar intensity diagrams of the two luminaires used, as goals, in the tests made to the software.

4.1 Starting and goal IES files

In order to maintain some consistency all the tests were preformed using two different types of luminaires. Figure 4.1 shows the polar intensity diagrams of the luminaires used. The luminaire represented in Figure 4.1a focus the light. The other luminaire, Figure 4.1b, does the opposite and floods the light. Also, for all the tests conducted with the software, the same light source was used, an LED manufactured by Cree, Inc. This light source had an uniform light distribution, as it is shown in Figure 4.2.

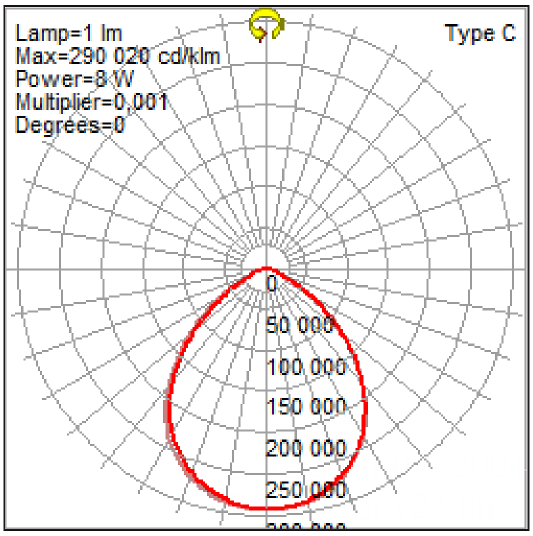


Figure 4.2: Polar intensity diagram of the LED used as light source in the tests made to the software. The LED is manufactured by Cree, Inc.

Other experiments, with these and other IES files, were also performed to debug the software and confirm the results. However, for succinctness, only the most relevant tests and results are reported in this chapter.

4.2 Software validation

Although each component had already been tested the interactions between them could reveal faults in the software. So the first tests assessed the validity of the software. Also, using the software modularity, a different solution to the problem was implemented and tested.

4.2.1 Alternatives

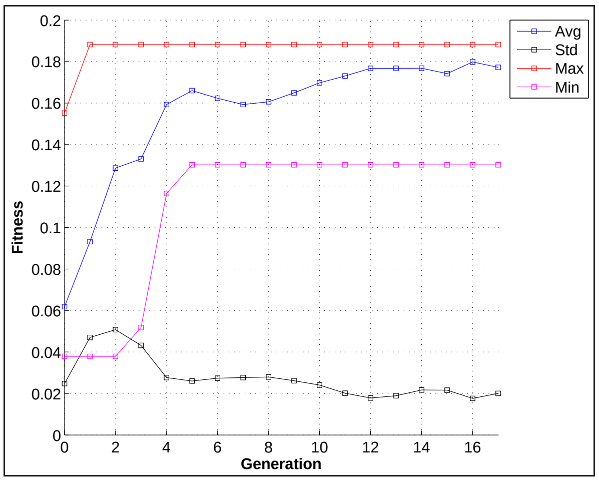
Even though the software was developed to approach the problem in a specific way, it is versatile enough to allow other approaches to be tested with minimal changes to the source code.
So a different approach was tested. Instead of working with free form lenses, searching for the form that would create a light distribution as close as possible to the intended one, the software was modified to work with a pre-defined form. The chosen form was a paraboloid. With this modification the software goal changes from searching for the lens form, to searching for the best combination of thickness and bending, which are used to generate the paraboloid-shaped lens that creates a light distribution as close as possible to the intended one.

		FOCUS	FLOOD
LENS	**Radius** [m]	0.01	
	Aspect Ratio	1	
	Points Interval [m]	0.0001	
	# Points	4115	
	# Triangles	7949	
LED	**IES**	CR Series 4” Retrofit	
	Position [m]	(0,0,10.000)	
	# Rays	538	
Luminaire	**IES**	LED Energy Smart^{TM} MR16	HomeMadeObjective
Genetic Algorithm	**Conf. File**	conf.conf	
	Seeds File	seeds.conf	
	Population	100	
	Max Generations	100	
	Mutation Prob.	0.30	
	Crossover Prob.	0.65	
	Crossover Type	One Point	
	Reproduction Prob.	0.05	
	Max Fitness	0.188194	0.049237
	Best Individual	G:1 I:24	G:8 I:65
Duration [h]		0.2723	0.6205

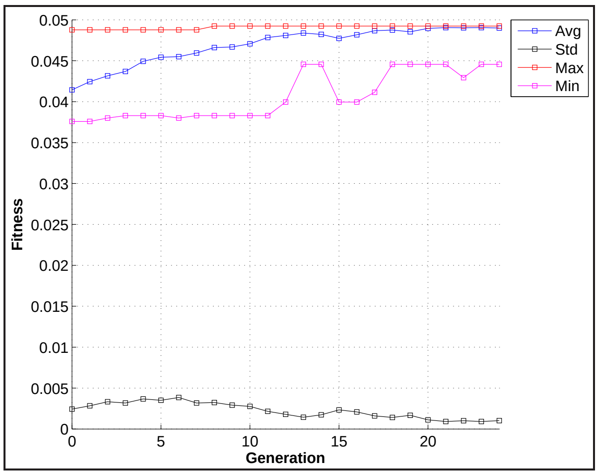
Table 4.1: Configurations used in tests to the 2 parameters gene variation of the software and summary of results obtained.

As Table 4.1 shows, two tests were made. The two tests differed only on the luminaire used as goal. The probabilities of the genetic operators (mutation, crossover) were chosen based on the results obtained in the preliminary tests, Subsection 4.2.2, and on literature. The initial population used in the tests was the same and it was composed of randomly generated values of thickness and bending.

Table 4.1 also shows some results of the tests. As expected, the FOCUS test achieved a better result than the FLOOD test, because the software was using only paraboloid-shaped lens. This form tends to concentrate the light rather than spreading it, so the result is the expected. Despite this, both results are far from the goal. A perfect match would have a maximum fitness of 1 and the best solutions found weren't even close to that value.



(a) FOCUS test.



(b) FLOOD test.

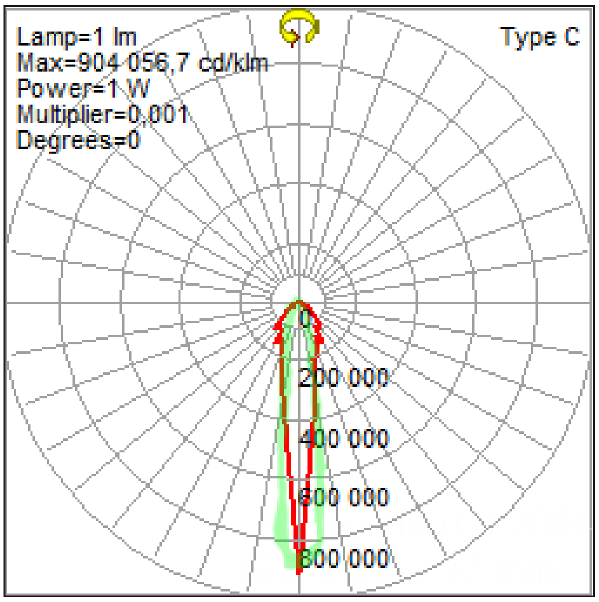
Figure 4.3: Evolution results of the tests made.

Figure 4.3 shows the evolution of the solutions through the software execution. It shows the fitness of the best individual in the population (red line), the fitness of the worst individual (purple line), the average fitness in the population (blue line) and the standard deviation in the population (black line).

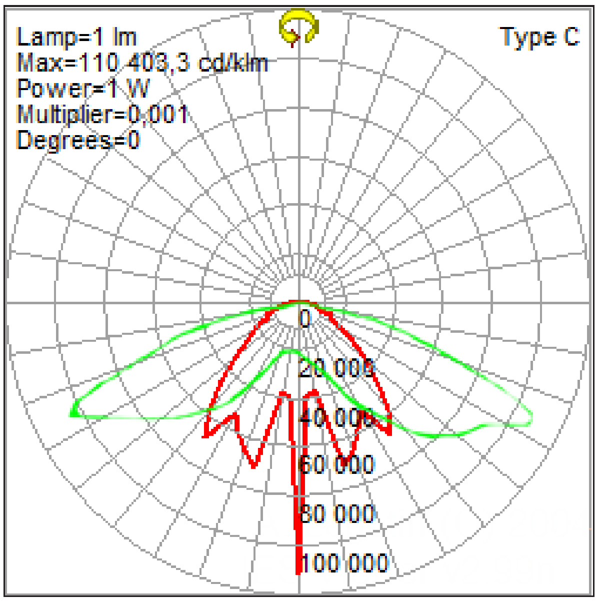
In the FOCUS test the best solution was obtained just after one generation. The fitness of the best individual improved quickly but then the software was unable to improve the solution in the next generations. In the FLOOD test the population improved slowly and steadily as shows the average fitness of the population. However, analysing the standard deviation it is clear that as the population improved the genetic diversity was lost and the individuals became all alike, causing the population to stop evolving.

Figure 4.4 shows the comparison of the light distributions of the luminaire and the best solution found in each test. Both images show that the solutions obtained were still far from the goal.

When analysing these results it is necessary to take into consideration the physic limitations of the lenses. In the FOCUS test the results, Figure 4.4a, were better because the goal was to create a light distribution where the light was more focused. Even so, the goal distribution didn't focused the light in one point as paraboloids do, so the result could never be perfect. This physic limitation is even clearer in the FLOOD test where the goal was to flood the light, rather than focusing it. As Figure 4.4b shows, the solution obtained is far from the goal and due to the physic limitations it is impossible to ever reach the goal.



(a) Comparison between the best light distribution resulting from the FOCUS test and the luminaire used as goal.



(b) Comparison between the best light distribution resulting from the FLOOD test and the luminaire used as goal.

Figure 4.4: Polar intensity diagrams of the best individuals resulting from the tests made.

From the results of the tests it was clear that this approach had several limitations and wasn't worth pursuing it. This doesn't mean that with more research and testing this type of approach wouldn't work, it just wasn't priority to pursue this route. Despite that, these tests proved the versatility of the software.

4.2.2 Preliminary tests

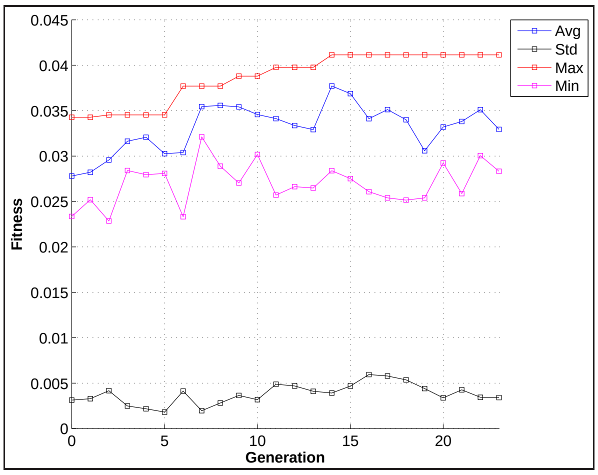
Table 4.2 shows the configuration used in the first tests made to the system. The first test, FOCUS Proof, had the goal of verifying the software correct functioning. The other two tests, FOCUS MutPlus and FOCUS CxPlus, were made to assess the effects of each genetic operator (mutation, crossover) in the results. As the goal light distribution was symmetric the three tests designed only half of a lens, instead of designing the complete lens. This improved the performance of the software. Also, for all the tests the same randomly generated population was used.

		FOCUS Proof	**FOCUS MutPlus**	**FOCUS CxPlus**
LENS	**Radius** [m]	0.01		
	Aspect Ratio	1		
	Points Interval [m]	0.0001		
	# Points	16016		
	# Triangles	31515		
LED	**IES**	CR Series 4” Retrofit		
	Position [m]	(0,0,10.004)		
	# Rays	1612		
Luminaire	**IES**	LED Energy Smart^{TM} MR16		
Genetic Algorithm	**Conf. File**	parab_conf.conf		
	Seeds File	rand_seeds.conf		
	Population	10	50	
	Max Generations	50		
	Mutation Prob.	0.35	0.80	0.1
	Crossover Prob.	0.55	0.1	0.80
	Crossover Type	One Point		
	Reproduction Prob.	0.1	0.1	0.1
	Max Fitness	0.0411396	0.0461285	0.0517391
	Best Individual	G:14 I:9	G:5 I:49	G:19 I:6
Duration [h]		1.75	6.54	9.19

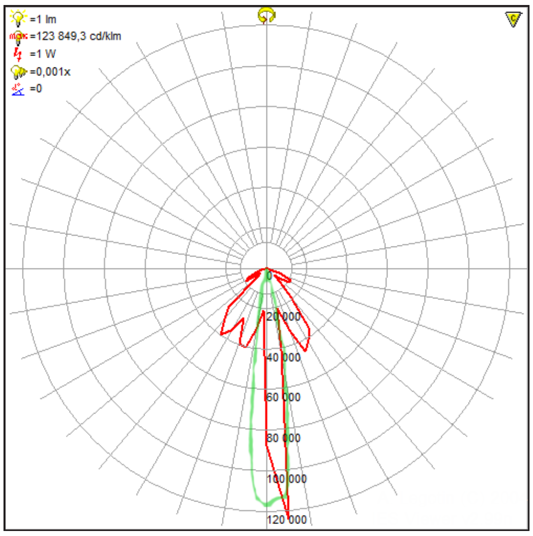
Table 4.2: Configurations used in the preliminary tests made to the framework and summary of results obtained.

In the first test, to prove that the software was working, the evolution results were more important than the solutions found. Analysing the results it was possible to see if the software was working correctly. Figure 4.5a shows the results of the FOCUS Proof test. The figure shows that the solutions were evolving. Analysing the maximum fitness of the population it is possible to see that the fitness improved at a steady pace. Also the population average fitness fluctuated through the execution, showing that indeed the solutions were being exposed to the genetic operators.
It is worth to verify that the best solution found was far from the desired goal. Figure 4.5b shows the comparison between the luminaire used as goal (green) and the best solution found (red), the differences are clear.

The FOCUS MutPlus test and the FOCUS CxPlus test had different goals, in relation to the FOCUS Proof test. With these tests the goal was to assess the influence of each genetic operator



(a) Fitness evolution in FOCUS Proof test.



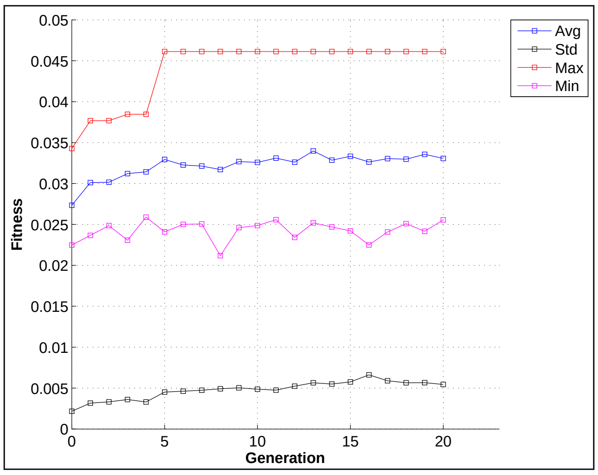
(b) Comparison between the best light distribution resulting from the FOCUS Proof test (red) and the luminaire used as goal (gree).

Figure 4.5: Results from the FOCUS Proof testing.

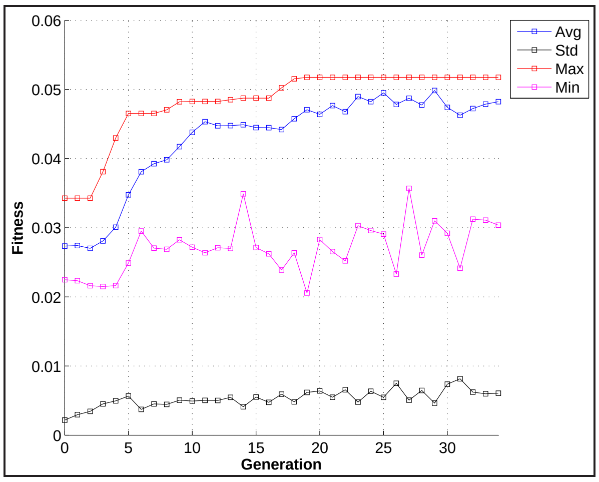
(mutation and crossover) over the evolution. So, as described in the Table 4.2, the two tests only differed in two parameters, the mutation probability and the crossover probability. To check the effects of each one of these parameters, each parameter was given a much higher probability than the other. So in the FOCUS MutPlus test the mutation probability was the higher one and in the FOCUS CxPlus test it was the crossover which had the highest probability.

Figure 4.6 shows the evolution results of each test. In the FOCUS MutPlus test the evolution was fast and sudden, stagnating just after the fifth generation. This result showed that such a high rate of mutation wasn't a good evolving strategy because there were no more improvements. The FOCUS CxPlus test was the opposite, the evolution followed a steady pace evolving throughout eighteen generations, and only then the evolution stagnated and the software stoped. Despite that, this evolution strategy seemed more promising.
The evolutions of each test followed different routes, even so the best solutions found had a similar fitness value. Both solutions were far from the goal and it is possible to see the comparison between the solutions and the luminaire used as goal in Figure 4.7.

The software version used in these tests didn't include the gaussian smoothing filter. So, the generated lenses had such irregular surfaces that they would not be suitable for production.



(a) Fitness evolution in FOCUS MutPlus test.



(b) Fitness evolution in FOCUS CxPlus test.

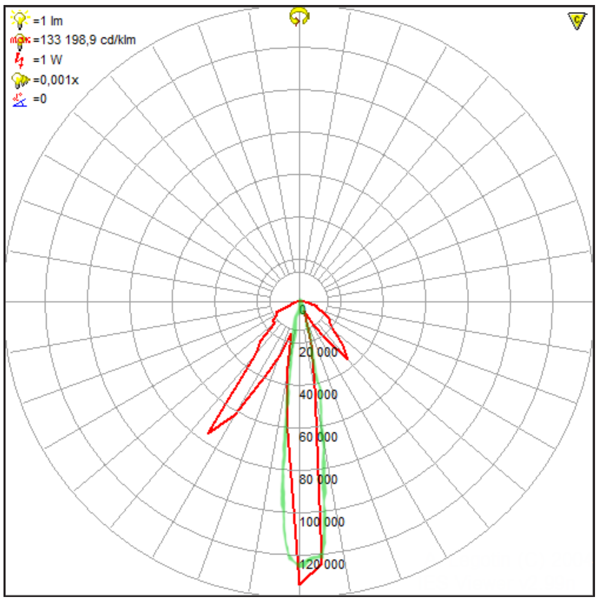
Figure 4.6: Evolution results of the tests made.

4.3 Genetic algorithm tuning

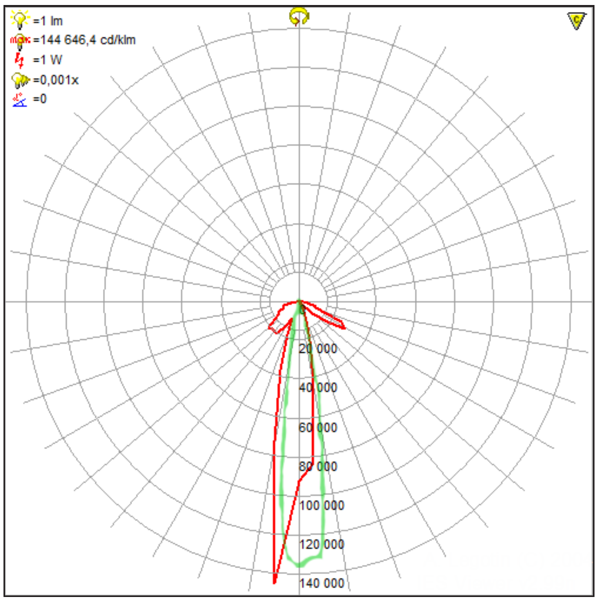
Open BEAGLE offers a variety of configurable properties and among those properties are the genetic operators. For the software to achieve the best performance possible it was necessary to test and analyse the options available. Among the options available for the genetic operators there was one type of mutation and five types of crossover. It was decided to test all the crossover types to determine which would preform better. Using the symmetry property the tests use only an eighth of a lens in order to increase the software performance.

4.3.1 Simple crossover

To assess the impact on the evolution, each crossover type was individually tested. The five types of crossover available at Open BEAGLE are: one point crossover, two points crossover, blend crossover, simulated binary crossover (sbx) and uniform crossover. All of them were tested using the same configurations and all used the same initial population. This population was composed by three different types of lenses: semi-spheres, paraboloids facing up and paraboloids facing down. For each type of lens generated both a thickness value and a bending value were randomly selected. As well as in the preliminary tests, Subsection 4.2.2, this software version still didn't include the gaussian smoothing filter and the generated lenses continued too irregular to be doable. However, these results led to the addition of the gaussian smoothing filter to the software.



(a) Comparison between the best light distribution resulting from the FOCUS MutPlus test and the luminaire used as goal.

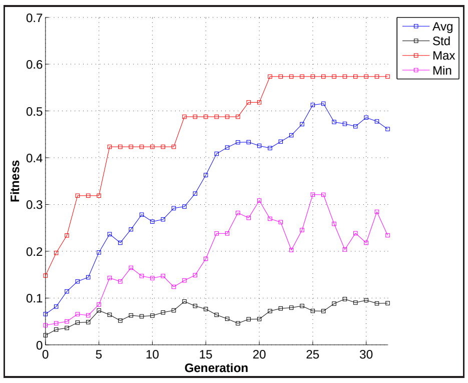


(b) Comparison between the best light distribution resulting from the FOCUS CxPlus test and the luminaire used as goal.

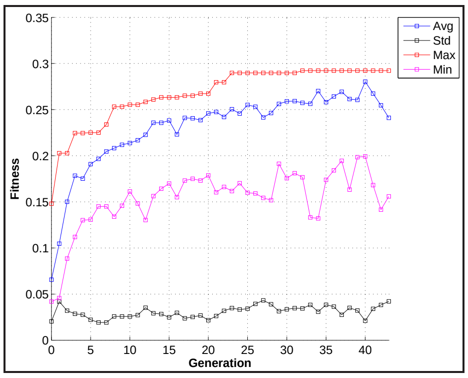
Figure 4.7: Polar intensity diagrams of the best individuals resulting from the tests made.

4.3.1.1 Emphasis on crossover

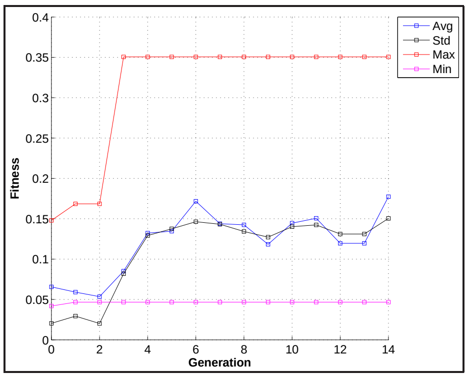
Table 4.3 shows the configurations used in the tests, where CX1P represents the test using the one point crossover operator, CX2P represents the test using the two points crossover operator, BLEND represents the test using the blend crossover operator, SBX represents the test using the simulated binary crossover operator and UNIF represents the test using the uniform crossover operator. In all the tests the light distribution used as goal was the one shown in Figure 4.1a. Figure 4.8 shows the results of the evolutions using different crossover operators. Comparing all the five results, the blend crossover, Figure 4.8c and the SBX crossover, Figure 4.8d. exhibit the worst results. In both cases the population average fitness didn't follow a clear improving tendency. The SBX crossover could possibly work if some adjustments were made to the software, however the blend crossover was clearly not appropriated for the problem.
On the contrary, the one point crossover, the two points crossover and the uniform crossover followed a steady evolution, improving the maximum fitness as well as the average fitness of the population. The best solution was found using the one point crossover. The polar intensity diagram of the best individual is shown in Figure 4.9 and even though the fitness value was just over 0.5, the light distribution was close to the goal.



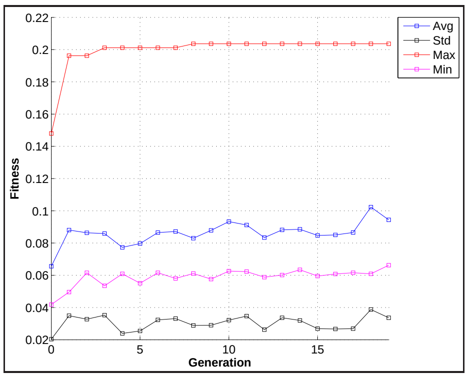
(a) One point crossover.



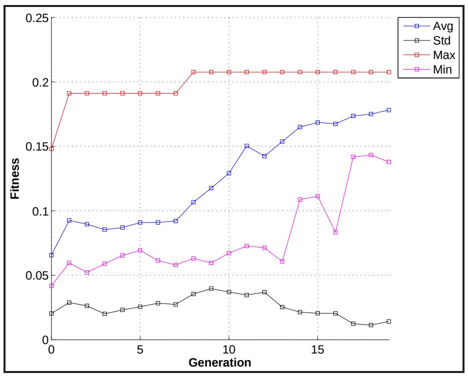
(b) Two points crossover.



(c) Blend crossover.



(d) SBX crossover.



(e) Uniform crossover.

Figure 4.8: Evolution results using different crossover types.

Although the two points crossover and the uniform crossover didn't achieve the same values of fitness they both had a steady evolution, with the average fitness of the population continuously improving. These results showed that the use of the one point crossover, the two points crossover and the uniform crossover created the best evolutions and had the best chances of finding the best solution, being worthy of further testing.

		CX1P	CXP2	BLEND	SBX	UNIF
LENS	**Radius** [m]	0.01				
	Aspect Ratio	1				
	Points Interval [m]	0.0001				
	# Points	4115				
	# Triangles	7949				
LED	**IES**	CR Series 4” Retrofit				
	Position [m]	(0,0,10.000)				
	# Rays	538				
Luminaire	**IES**	LED Energy Smart^{TM} MR16				
Genetic Algorithm	**Conf. File**	conf.conf				
	Seeds File	seeds.conf				
	Population	50				
	Max Generations	50				
	Mutation Prob.	0.30				
	Crossover Prob.	0.65				
	Crossover Type	One Point	Two Points	Blend	SBX	Uniform
	Reproduction Prob.	0.05				
	Max Fitness	0.573522	0.292224	0.350581	0.203628	0.207540
	Best Individual	G:21 I:24	G:32 I:3	G:3 I:47	G:8 I:49	G:8 I:4
Duration [h]		0.7475	1.7311	0.2528	0.5461	0.9769

Table 4.3: Configurations used in the tests made to the different crossover operators and initial population composed of parabolic and spherical lenses and summary of results obtained.

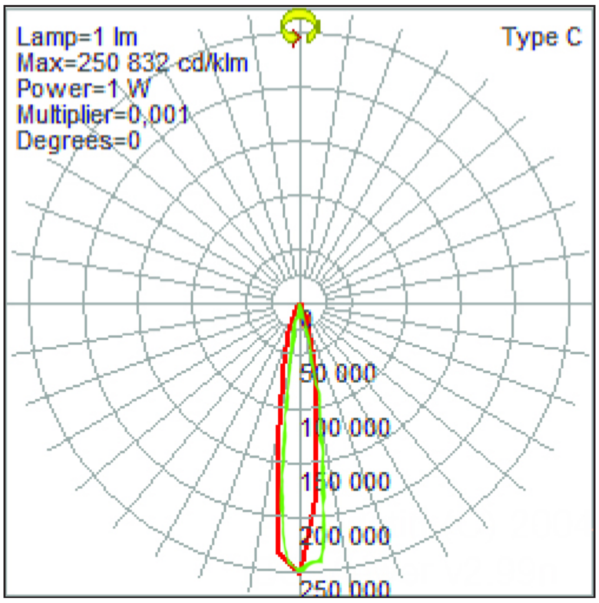


Figure 4.9: Comparison between the goal light distribution and the best solution, found using the one point crossover, light distribution.

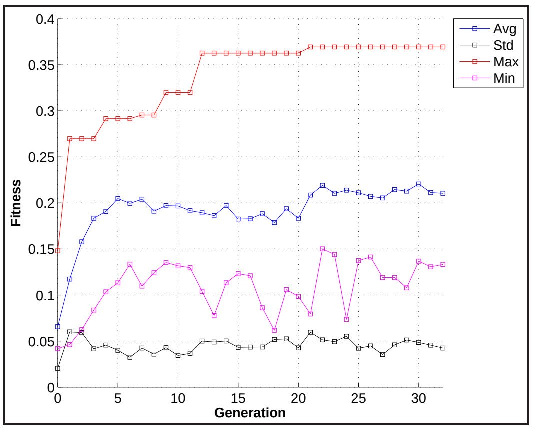
4.3.1.2 Emphasis on mutation

As stated in (4.3.1.1) the one point crossover, the two points crossover and the uniform crossover operators deserved to be further tested, so more tests were made using each one of these three operators. However, unlike the previous tests, where the greater emphasis was put into the crossover in order to observe its effects in the evolutions, in these tests the greater emphasis was given to the mutation. Table 4.4 shows the configurations used in the tests, which, essentially, are the same presented in Table 4.3, but with switched crossover and mutation probabilities, in order to change the genetic operator with greater impact. This switch allowed for a better understanding of the influence of each genetic operator in the evolution process.

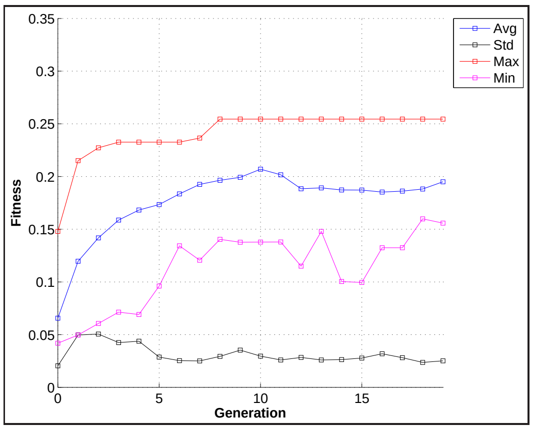
		CX1P	CXP2	UNIF
LENS	**Radius** [m]	0.01		
	Aspect Ratio	1		
	Points Interval [m]	0.0001		
	# Points	4115		
	# Triangles	7949		
LED	**IES**	CR Series 4” Retrofit		
	Position [m]	(0,0,10.000)		
	# Rays	538		
Luminaire	**IES**	LED Energy Smart^{TM} MR16		
Genetic Algorithm	**Conf. File**	conf.conf		
	Seeds File	seeds.conf		
	Population	50		
	Max Generations	50		
	Mutation Prob.	0.65		
	Crossover Prob.	0.30		
	Crossover Type	One Point	Two Points	Uniform
	Reproduction Prob.	0.05		
	Max Fitness	0.369304	0.254519	0.227777
	Best Individual	G:21 I:49	G:8 I:5	G:32 I:11
Duration [s]		1.1053	0.9736	2.2972

Table 4.4: Configurations used in the tests made to the different crossover operators with greater emphasis on mutation and initial population composed of parabolic and spherical lenses and summary of results obtained.

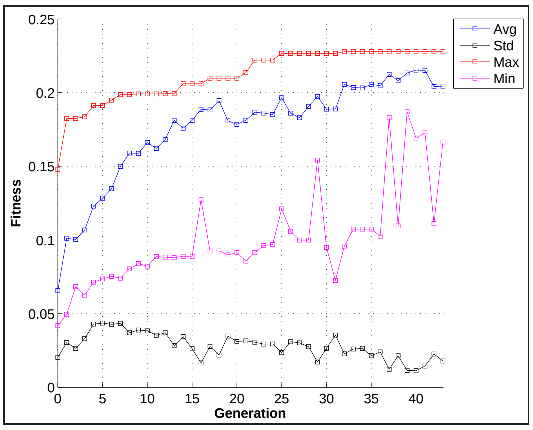
The resulting evolutions can be seen in Figure 4.10. As seen by the previous tests, the evolutions followed a steady pace. The test using one point crossover had a similar performance to the test made with the one point crossover in the Subsection 4.3.1.1, with the maximum fitness evolving gradually. Even so, the population average fitness didn't improve the same as before. The reduction in the average fitness is due to the increase of mutations which can improve or, more likely, worsen the individual. Also the new two points crossover test achieved a similar result to the test made with the two points crossover in the Subsection 4.3.1.1, but the evolution stalled just after eight generations. In this case the increase of the mutations led to an increase in performance, since both tests achieved a similar result.



(a) One point crossover.



(b) Two points crossover.



(c) Uniform crossover.

Figure 4.10: Evolution results using different crossover types and giving greater emphasis to the mutation.

Last, the test conducted with the uniform crossover showed the best performance. The evolution occurred at a steady pace with the maximum fitness and the average fitness having a similar improvement. If different termination parameters were set, the evolution could have improved indefinitely. The uniform crossover tends to create a homogeneous population, leading to the decrease of the genetic diversity. However the high rate of mutation in the population creates the necessary genetic diversity for the population to continue to evolve. These results showed that the configuration used in this test, was a good configuration and worth exploring.

4.3.2 Multiple crossover

Open BEAGLE allows the use of various crossover operators in the same evolution strategy. From all the five crossover operators provided, the one point crossover, the two points crossover and the uniform crossover got the best results in the previous tests. So, the tests in this section use combinations of those three crossover operators.

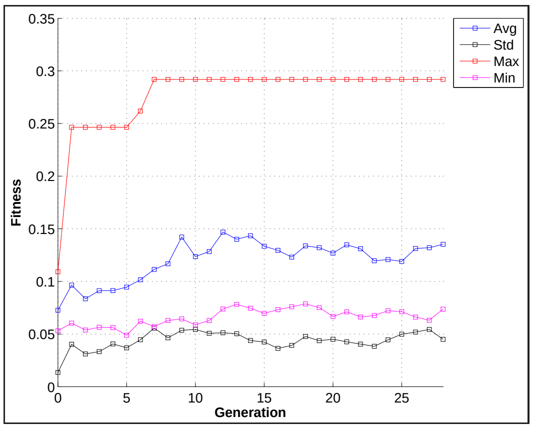
All the tests had as goal the light distribution shown in Figure 4.1a and all of them used the same initial population. This population was composed by three different types of lenses: semi-spheres, paraboloids facing up and paraboloids facing down. For each type of lens generated both a thickness value and a bending value were randomly selected.

Table 4.5 shows the configurations used in the four tests.

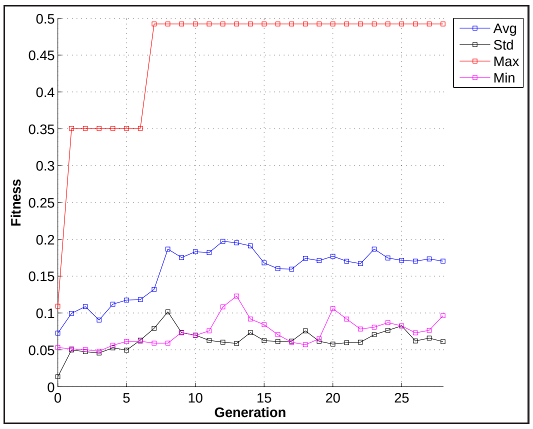
		EQUAL	CX1P	CX2P	UNIFORM
LENS	**Radius** [m]	0.01			
	Aspect Ratio	1			
	Points Interval [m]	0.0001			
	# Points	4115			
	# Triangles	7949			
LED	**IES**	CR Series 4” Retrofit			
	Position [m]	(0,0,10.000)			
	# Rays	538			
Luminaire	**IES**	LED Energy Smart^{TM} MR16			
Genetic Algorithm	**Conf. File**	conf.conf			
	Seeds File	seeds.conf			
	Population	50			
	Max Generations	50			
	Mutation Prob.	0.05			
	Crossover Prob.	0.3;0.3;0.3	0.5;0.2;0.2	0.2;0.5;0.2	0.2;0.2;0.5
	Crossover Type	One Point;Two Points;Uniform			
	Reproduction Prob.	0.05			
	Max Fitness	0.291890	0.492220	0.353427	0.210984
	Best Individual	G:7 I:49	G:7 I:1	G:5 I:0	G:3 I:28
Duration [h]		0.5729	0.5755	0.3534	0.2109

Table 4.5: Configurations used in the tests using combinations of different crossover operators and initial population composed of parabolic and spherical lenses and summary of results obtained.

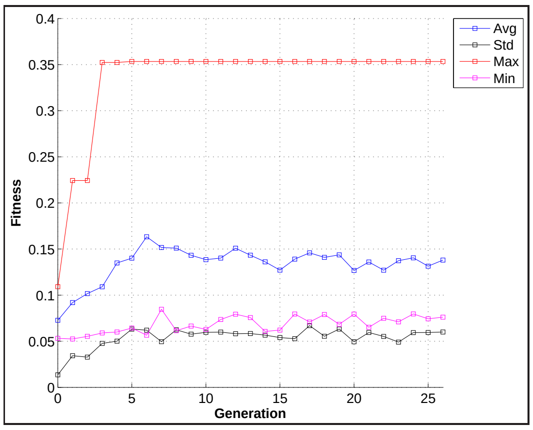
The evolution results shown in Figure 4.11, reveal that the evolutions followed a different path from the evolutions seen in previous tests. Instead of a continuous evolution, with several increases in the maximum fitness and a steady improvement of the population average fitness, this approach resulted in a fast rise of the maximum fitness while the population average fitness suffered only small improvements, remaining close to its initial value.



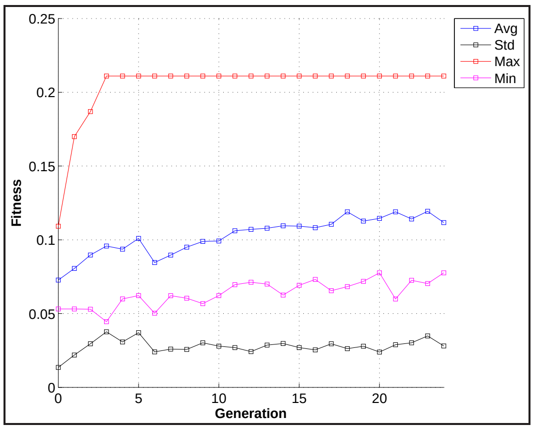
(a) Configuration with equal probabilities for the one point crossover, the two points crossover and the uniform crossover.



(b) Configuration with higher probability for the one point crossover and equal probability for the other two operators.



(c) Configuration with higher probability for the two points crossover and equal probability for the other two operators.



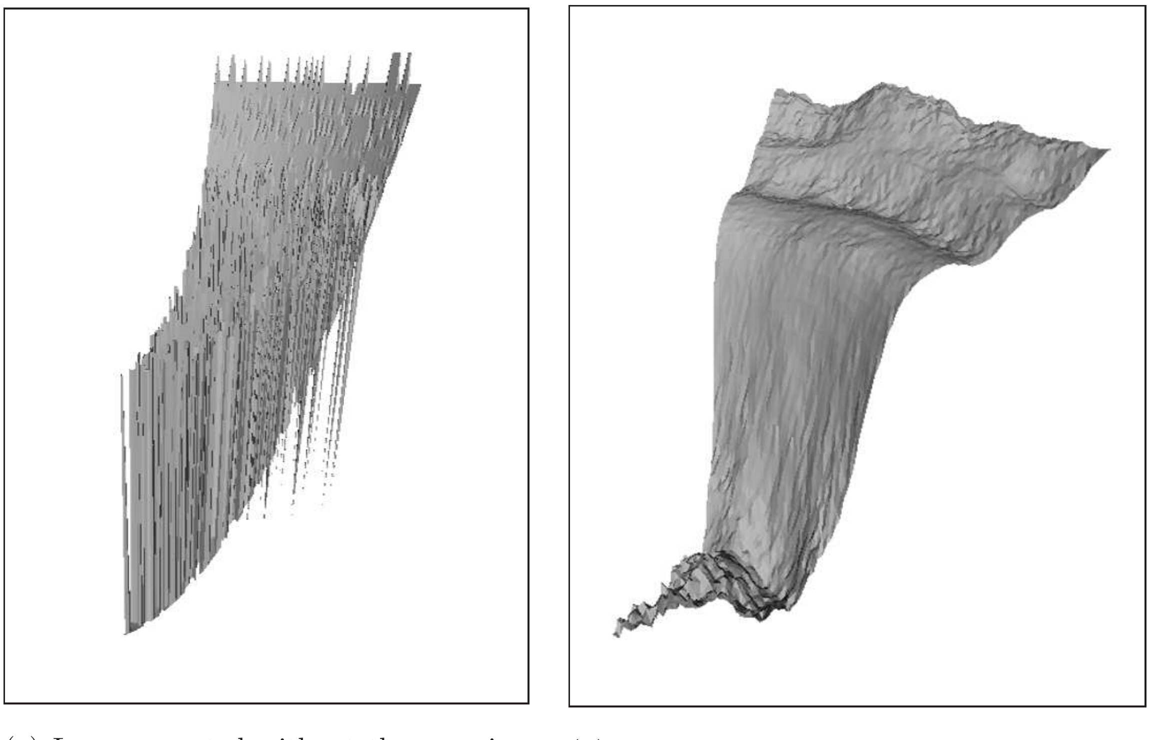
(d) Configuration with higher probability for the uniform crossover and equal probability for the other two operators.

Figure 4.11: Evolution results using different combinations different crossover operators.

The fitness results are very similar to the ones obtained in the tests shown in Subsection 4.3.1.1. However, due to the fast rise of the maximum fitness, the software performance had a huge improvement, reducing the mean time of execution of the software in more than 50% while maintaining a similar mean maximum fitness.

The software version used in these tests already had the gaussian smoothing filter integrated. This filter not only improved the quality of the lenses generated, making them doable, but it also had some effect in the evolution. Without the filter the changes made to the lens surface happened at isolated points not affecting the neighbouring area, resulting in extremely irregular surfaces. This led to constant variations of the fitness. With the filter included, the changes are smoothed and the fitness becomes more robust, which has impact in the evolution.

Figure 4.12 shows a comparison between two lenses, one generated without the filter and other generated with the filter. It is easy to see that with the filter almost all the spikes disappear leaving a smoother surface.


(a) Lens generated without the gaussian smoothing filter.

(b) Lens generated with the gaussian smoothing filter.

Figure 4.12: Comparison between two lenses, one generated with the gaussian smoothing filter and the other without it.

Chapter 5

Conclusions

This dissertation proposes a framework to automatically optimize the form of a secondary lens of an LED luminaire in order to optimize the light distribution towards a desired goal.
There has been some research on the subject but to the best of our knowledge, never using the approach used here. Given the partial information about the lens, the LED to use as light source and the desired light distribution output, the proposed framework requires three main steps:

1. Mesh and lens generation;
2. Simulation of light emission from the LED using ray tracing;
3. Search for optimized lens bottom surface using a genetic algorithm;

In the tests described it was used an LED, manufactured by *Cree, Inc*, with a symmetric light distribution. Also, two different luminaires were used as goal. One of the luminaires has a compact light distribution, focussing the light, while the second, with a more dispersed light distribution, floods the light. They were used in tests with different configurations of the genetic algorithm, i.e., different genetic operators with different properties.
The tests revealed which operators had a greater chance of succeeding, which were the one point crossover, the two points crossover, the uniform crossover and the mutation. The evolutionary strategies which used combinations of these crossover operators with the mutation operator achieved the best results. The lenses evolved in the available time showed a fitness of up to 0.49, the evolution results showed that this framework had the potential to achieve better results, using the correct evolutionary strategy. Its main drawback is the time required.
From a practical point of view, the framework was, indeed, capable of automatically improve the

form of a secondary lens of an LED luminaire. Notwithstanding the fact that improvements can be made to build a more robust and faster framework. The work developed is just the first step in an area that has huge growing potential. Nonetheless, it can be said that the main goal of the dissertation was achieved.

5.1 Future work

As stated before, the work developed is just the first step so, there are some improvements and further research that can be made to evolve the current framework.
The first improvement is related to the ray tracing and the performance of the framework. Most of the framework execution time is due to the ray tracing so, applying parallel computing in central processing units (CPU) or in graphics processing units (GPU) through CUDA or OpenCL to the ray tracing, it is possible to largely reduce the execution time, improving the framework's performance. Ultimately, all the framework can be adapted to take advantage of parallel computing. Another change that can improve the framework is, instead of computing the triangles normal vector statically, compute them linearly allowing a reduction on the number of points, therefore, a reduction on the number of triangles, needed to generate the mesh without creating lighting spots.
The framework can also benefit from the development of a graphical user interface (GUI). With a well designed GUI, the framework would be easier to interact for unfamiliar users and understand, allowing a faster manipulation of the configurable parameters. In addition to the functional features, a GUI has the aesthetics features that make the framework more appealing to the user.

References

[1] L.C. Brito A.J.Alves E.G. Marra A.P. Coimbra R. S. Ferraz E. G. Domingues J.L.F. Barbosa, W.P.Calixto and D. Pinheiro Neto. Secondary lens optimization for led lamps. 1, 1.2

[2] Fei Chen Zongyuan Liu Kai Wang, Sheng Liu and Xiaobing Luo. Novel application-specific led packaging with compact freeform lens. 1, 1.1

[3] Zhen-rong Zheng Hai-feng Li Xu Liu Peng Liu, Reng-mao Wu. Optimized design of led freeform lens for uniform circular illumination. 1.1, 1.2

[4] Wikipedia. Lighting — Wikipedia, the free encyclopedia, 2014. [Online; Last accessed 26-June-2014]. 2.1

[5] Wikipedia. Lamp (electrical component) — Wikipedia, the free encyclopedia, 2014. [Online; Last accessed 27-June-2014]. 2.1.1

[6] Lighting Industry Federation Limited. *Lamp Guide 2001*, fourth edition. 2.1.1, 2.1

[7] Wikipedia. Light-emitting diode — Wikipedia, the free encyclopedia, 2014. [Online; Last accessed 27-June-2014]. 2.1.2, 2.1.2

[8] Inc. Cree. Cree components xlamp xp-e leds, 2014. [Online; Last accessed 27-June-2014]. 2.2

[9] Wikipedia. Led lamp — Wikipedia, the free encyclopedia, 2014. [Online; Last accessed 27-June-2014]. 2.1.2

[10] Eartheasy.com. Led light bulbs: Comparison charts - eartheasy.com solutions for sustainable living, 2012. [Online; Last accessed 27-June-2014]. 2.1.2, 2.1

[11] LED Hut. Led light bulbs comparison - save ££££s install led light bulbs now, 2014. [Online; Last accessed 27-June-2014]. 2.1.2, 2.1

[12] Wikipedia. Photometry (optics) — Wikipedia, the free encyclopedia, 2014. [Online; Last accessed 27-June-2014]. 2.1.3

[13] Inc. Gigahertz-Optik. Basic photometric quantities, 2014. [Online; Last accessed 27-June-2014]. 2.1.3, 2.1.3.1, 2.1.3.2, 2.1.3.3, 2.1.3.4

[14] Wikipedia. Luminous flux — Wikipedia, the free encyclopedia, 2014. [Online; Last accessed 27-June-2014]. 2.1.3.1

[15] Wikipedia. Luminous intensity — Wikipedia, the free encyclopedia, 2014. [Online; Last accessed 27-June-2014]. 2.1.3.2

[16] Wikipedia. Luminance — Wikipedia, the free encyclopedia, 2014. [Online; Last accessed 27-June-2014]. 2.1.3.3

[17] Wikipedia. Illuminance — Wikipedia, the free encyclopedia, 2014. [Online; Last accessed 27-June-2014]. 2.1.3.4

[18] Photometric Testing. Talking photometry - understanding photometric data files, 2014. [Online; Last accessed 27-June-2014]. 2.1.3.5, 2.1.3.6, 3.1.2

[19] McGraw-Hill. *McGraw-Hill Encyclopedia of Science & Technology.* Tenth edition. 2.2, 2.2.1, 2.5, 2.2.1, 2.2.1.1, 2.7, 2.2.1.2

[20] BBC News. World's oldest telescope? [Online; Last accessed 27-July-2014]. 2.2

[21] Wikipedia. Optics — Wikipedia, the free encyclopedia, 2014. [Online; Last accessed 11-July-2014]. 2.2

[22] Benjamin Crowell. *Light and Matter.* 2.4, 2.2, 2.6

[23] J. Peatross and M. Ware. *Physics of Light and Optics.* Avaiable at optics.byu.edu. 2.2.1

[24] Leno S. Pedrotti. *Fundamentals of Photonics - Module on Basic Geometrical Optics.* 2.2.1, 2.2.1.2, 2.2.1.2

[25] Seoung Soo Lee Si Hyung Park and Jong Hwa Kim. The delaunay triangulation by grid subdivision. 2.3

[26] Wikipedia. Delaunay triangulation — Wikipedia, the free encyclopedia, 2014. [Online; Last accessed 02-August-2014]. 2.8

[27] Mark; Otfried Cheong; Marc van Kreveld; Mark Overmars de Berg. *Computational Geometry: Algorithms and Applications.* 2.3

[28] H. Edelsbrunner. *Geometry and Topology for Mesh Generation.* 2.3

[29] Gary Bradski and Adrian Kaehler. *Learning OpenCV.* 2.3

[30] Itseez. Opencv, 2014. [Online; Last accessed 03-August-2014]. 2.3

[31] Itseez. About — opencv, 2014. [Online; Last accessed 03-August-2014]. 2.3

[32] Esteban Clua Diego Barboza. Gpu-based data structure for a parallel ray tracing illumination algorithm. 2.4, 2.4

[33] Brian J. Ross. Cosc 3p98: Ray tracing basics. [Online; Last accessed 06-February-2014]. 2.4, 2.4, 2.4.2

[34] Pete Smith. Offline archives - pete smith, 2012. [Online; Last accessed 13-August-2014]. 2.9

[35] Farrokh Hodjatkashani Chiya Saeidi and Azim Fard. New tube-based shooting and bouncing ray tracing method. 2.4

[36] Oyonale. Free 3d models - glasses, pitcher, ashtray and dice (pov-ray), 2014. [Online; Last accessed 13-August-2014]. 2.10

[37] João Manuel Brisson Lopes. Ray tracing. [Online; Last accessed 13-October-2013]. 2.4.1, 2.4.1.1, 2.4.2, 2.11, 2.12

[38] Darrel Whitley. A genetic algorithm tutorial. 2.5

[39] Mitchell Melanie. *An Introduction to Genetic Algorithms.* 2.5

[40] Christian Gagné and Marc Parizeau. *Open BEAGLE Manual.* 2.5, 2.5.2, 2.5.2

[41] Christian Gagné and Marc Parizeau. Genericity in evolutionary computation software tools: Principles and case study. *International Journal on Artificial Intelligence Tools*, 15(2):173–194, 2006. 2.13, 2.5.2, 2.2

[42] Shari Lawrence Pfleeger and Joanne M. Atlee. *Software Engineering - Theory and Practice.* Third edition. 3

[43] Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2000. 3.2.1

[44] Paul Bourke. Interpolation methods. [Online; Last accessed 13-August-2014]. 3.2