Università
Ca'Foscari
Venezia

Masters's Degree
in Computer Science

Final Thesis

Multi-view Light Source Estimation for Automated Industrial Quality Control

Supervisor
Prof. Filippo Bergamasco

Assistant supervisor
Dott. Mara Pistellato

Graduand
Mauro Noris
Matricolation number
877632

Academic year
2019/2020

Acknowledgements

This work ends my journey at the University Ca'Foscari of Venice. I've moved in from Bergamo in order to engage myself in the Master degree in Computer Science, and during this period many people supported and encouraged me to be the best version of myself. I found myself obbliged to express my gratitude to all those great people. There's no need to say this, however I would like to thank my family and especially my parents; without their support and encouragement I would not have a degree today. I would like to express my admiration and gratitute to both Prof. Filippo Bergamasco and Drs. Mara Pistellato, who patiently guided me in the Computer Vision world and aided me during the developement and the drafting of this thesis. I want to send a big shout-out to all my friends who I encountered during these two years, which made the hours of studying less heavy. Moreover, I would like to thank all my friends back in my hometown, which helped me relax and enjoy life to the fullest when I was far away from the university. Finally, I would like to thank all the teaching staff members of Ca' Foscari, since helped me grow as a person and was always available to solve my doubts and answer my question.

Abstract

Nowadays, Computer Vision comprises a lot of different useful applications, ranging from real-time surveillance to series production.

Since most of the mass-production systems are automatized, there is a non-zero probability for an error to happen, which in some cases would make the final product useless. In that regard, many factories often integrate into their assembly line a specialized camera system for the purpose of identifying errors at the end or during the production. In particular, a device composed of a network of calibrated cameras is usually employed to identify metrological and visual defects.

Together with cameras, such device could comprise multiple light sources to easily spot defects caused by an imperfect structure of the object's surface.

This work explores the possibility of increasing the precision of such an inspection system by estimating the position of each light source. The estimation is carried out through the analysis of the shading observed on the surface of some spherical Lambertian objects in different poses and from different points of view.

Index

Chapter 1

Introduction

Illumination plays a big role in Computer Vision, since it can give useful information about the scene and the object which comprises. However in many cases and systems the user has no control or information over the illuminants and their geometric properties. That's why many researchers started to study techniques about inferring light sources' properties by photograms of a specific scene, however in many cases it resulted in an ill-posed problem. This is because if no specific assumption on the scene are made, no precise model can be used in order to correctly describe the light.

In [1], Langer and Zucker stated that the definition of a light source is not an easily defined concept: many models can be used with respect to what kind of illuminant it is considered. For example, when talking about an outdoor scene the most incisive light source is considered to be the sun most of the time, which can be modelled as a "point at infinity", while indoors the source can have a point-like source with coordinates in a specific reference frame.

Having information on the light source can lead to an improvement on an already functioning system and Computer Vision's techniques. As Powell states in [2], some previous work has been affected by the lack of knowledge on the light source.

Moreover, being able to estimate the properties of the illuminant can improve the applicability of a system, for example it can lead to achieve *colour constancy* so that colours perceived by a visual system remains stable, regardless the of illuminant which enlightens the scene. Furthermore, illuminant estimation is needed for the shape-from-shading problem. Shape-from-shading is an inverse problem which involves the reconstruction of a 3 object's surface by extrapolating its features from 2D images.

Many techniques have been designed and developed, in order to reach a good state-of-the-art solution, however no technique has been recognized as standard. Most of the times the objective is a good tradeoff between precision and quickness, because many of those techniques are implemented on small hardwares, which generally do not have large memory. This techniques are often implemented in Augmented Reality scenarios.

Such methods have found multiple applications, ranging from cultural heritage from industrial application. The main reason for light estimation nowadays regard surface reconstruction, in which an object is reconstructed to a certain degree of precision in a virtualized 3D space. For example, ambient light estimation is often

used to construct a realistic light model in videogames and 3D render as exploited in [3].

1.1 What is light in a scene?

When talking about illuminations and light source, the first thing that comes to mind is a point-like illuminant which irradiates the scene,which has a precise direction. However, in most cases the light source is not summarizable in a point, since can have different natures and different sources.

With regard to the initial data and scene, in order to estimate the light source geometrical properties it is mandatory to make some assumption on the light and the scene. Using a particular object in the scene, which respects some specific properties is a common choice, since light can be inferred using its reflection on objects.

Figure 1.1: Two pictures of the same sphere with different illuminant

A practical example of the diversity of light nature can be seen in the figure 1.1: The same scene is illuminated respectively by a led light, which has a linear form and include multiple leds, and a point-like illuminant. The led illuminant will have a stronger radiance and the sphere surface will present a greater brightness, however nothing can be infered about the structure of the illuminant looking at the shade casted on the sphere.

In many case, the light which illuminates the scene is always the same, which is a common scenario when talking about industrial machines where ad-hoc system with well-known illuminants are used. However, for other scenarios like AR applications where the illumination changes and even multiple illuminants are in play, the estimation of the light depends on multiple factors and the geometry of the illuminant cannot be easily defined.

1.1.1 Light source models

Illuminant estimation is not an easy problem and it is an ill-posed one if only images are available. A good standard solution for this problem has not been found yet. This is because a general model for light has not been devised yet and so each technique will strongly depend on the kind of illuminant used. Many techniques uses a precise and simplified model in order to maximize the precision of the estimation

or because the scenes on which these methods are applied have always the same type of illuminant, so a generalized model is not needed.

The two classical model used are the **directional source** model and the **point source**. The first one, used by a lot of techniques like [4] and [5], approximates really well with far away lights, placed at infinity like the natural sunlight. This works well even when multiple light sources are involved, and so the direction of every illuminant is estimated using different patches on the image as proposed in [6].

When talking about sources decently near the object, maybe in the boundaries of a small closed space such as a room, the directional model may not hold anymore. The point model is used for such situations, as can be seen in the method explained at [7]. However, these traditional methods suffered from the fact that the type of light must be assumed or known in order to be modelled properly. In complex systems, with different kinds of illuminants, these models do not hold. In these cases an *area light model* may be used, which approximates very big light sources which arevery close to the light, however the problem becomes more complicated problem and might not have a real solution.

Many researchers tried to develop a general model which does not depend on the specificity of the illuminant involved, however it is shown that this would increment the computational complexity of the whole estimation algorithm. In [1], the light was modelled as a 4D hypercube, where illumination was described by rays and so light source of different kinds were represented as rays of different dimensions. In order to do it, the idea is to place an imaginary plane between the illuminant and the scene. By doing so, a ray can be parametrized using its origin and the point of intersection with the plane. The type of light will change with respect to the dimensionality of the model and by the numer of rays emitted from the light source. Zhou in [8] expands the work done by Langer and Zucker in [1], using a 6 dimensional light source model. By adding two dimensions, the light source can be described in more detail, adding information on light source position, size and intensity. The idea is to consider the plane of [1] approximately coplanar with the light source plane. In this way, many types of light sources can be represented and the framework will be effective even without assumptions or prior knowledge on the illuminant structure and properties. It is assumed that the rays of the light source cover the whole range of the scene, since some light source might be non-isotropic, which means it does not expand in all direction.

1.2 Related work

In this section the work of researchers on illuminant estimation is presented, where different light models and assumptions are presented.

1.2.1 Light estimation methods

Throughout the years, many techniques for illuminant estimation were developed, using different assumptions and features to exploit. Each method was designed based on prior knowledge on the illuminant and how much the scene was manipulable.

Light source can have different forms, nature and distance with respect to the scene. In many cases the illuminant taken into account is the sun, while in other

realities artificial illumination is considered. Moreover, the use case of the system whose light sources need to be estimated as to be taken into account. Industrial quality control system usually require a fast computation and uses images captured at the moment, maybe at the end of a conveyor to perform an automated quality control on products. This system usually do not have great memory or vast computational power, so a technique with a good tradeoff between precision and time is preferred where hardware implementation is needed, as stated in [9].

In other circumstances the illuminant involved in the scene may not be modelled in a proper way or may be unknown. When a higher precision is needed and a lot of previous data is available, regarding the type of illuminant and its position, a different approach might be used, which exploits the prior knowledge to solve the problem. An unsupervised learning approach is preferrable in those cases, as stated in [9], which sacrifices speed, at least for the training step, and requires a lot of memory, but outclasses the other methods in terms of applicability and results.

Another thing to keep into account in certain situation is the color of the light involved in the scene, because it has an influence over the pixel intensity. In many cases white illuminant are considerated (even sunlight is considered white) or some assumptions are made, like the **Gray-World** assumption, which assumes the average of an image to be gray. Moreover, usually invariant features are used and the effect of the coloured light source is removed using *white-balancing* techniques.

With regards to different assumptions, data and situations, the illuminant estimation methods can be classified in three categories, which is well explained in [9]:

- **Statistic-based** methods: Includes methods involving low-level image features which lead to an inference of light parameter. This category is suited and often applied for hardware implementation given their high speed and their relatively low computational complexity.

- **Learning-based** methods: These methods relies to training a specific model. This category proves to be the most precise and performs better if a sufficiently large dataset is available, but "due to training process and more complex structures, they often take longer to execute"[9]. This are often used for complex systems which as to perform calculations on scene where the illuminant is not always the same.

- **Gamut-based** techniques that exploits **gamut mapping**. Gamut mapping is the function which maps a pixel to the subset of possible colours. Most of those methods can be represented as a subclass of learning-based techniques, since they exploits classification in some cases. They use light's gamut consistency as the main exploit to estimate the light source.

The model used for the light depends on the situation in which the algorithm is applied, not by the method used. However, the choice of the method might be correlated with the illuminant type which is involved in the scene in some cases.

1.2.2 Statistic-based methods

The preferred category of methods for industrial quality control regarding light source estimation involve studying shades casted on particular (or not) surfaces. Many of them relies on the brightest pixel of the surface reflection, as explained in [10], since it represents the point where the light is perpendicular to the object In order to infer light from reflections, some assumptions must be done on the image regarding the color, illumination statistics or the scene geometry.

The first techniques implemented of illuminant estimation modelled the illuminant as a point "to infinity", since in most cases the light source considered was the sun. This particular concept was well explained by Pentland in [5], which has given an algorithm and a good starting point for light estimation methods from features extracted from the image. It assumed a infinitely distant light source and by working with Lambertian surfaces, it constructed a light model where the image irradiance has a well-defined form. From that form, if an hemisphere is considered as the surface, the tilt and the angle of the direction of the light can be retrieved. However, the approach designed bt Pentland had several flaws, which are corrected [11].

Powell in [2] worked with manmade light sources and so it modelled the light in a different way. Powell used two specular spheres, with known radius and position, to retrieve intensity and direction. The method uses the surface normal on particular highlight points on the sphere. Using the *law of sines* and the perspective projection of the highlight point, the point of intersection is calculated using a least-squares estimation. Another method which uses a known geometry object is explained in [12], which assumes an isotropic point illuminant, which is modelled as a radiance mapping to the points on the image. The two main assumptions used in order to estimate the light is the point structure of the illuminant and the existence of at least one rotation axis which does not affect the distribution of the radiated light. The objects used are two Lambertian cubes and the direction is estimated by first assuming a infinite distance between the light source and the object. The parameters are estimated using a *linear* least squares estimation, which is then used as input to a non-linear procedure, which is the textitGauss-Newton method in order to find the finite distance and direction.

Zang and Chellappa in [4] changed completely the approach and uses surface reflectance and statistics to infer surface albedo and to solve the shape-from-shading problem. What is estimated in [4] is the azimuth between the illuminant and the camera. Since the objective was surface reconstruction, the exact geometry of the illuminant was not needed. Another method for light direction estimation whose does not need a known object is presented in [13], where no specific object is used in the scene. The assumption use is that "there exists a segment of an occluding contour of an object with locally Lambertian surface reflectance in the image"[1], which has to be found. Nillius found candidates for such a contour using an heuristic algorithm and for such edges the intial guess of light direction is estimated. Then, after retrieving the intensity on contours, a Bayesian network is used to improve the estimates and arrive to the most probable light direction. Since the intensity formula of pixel on a Lambertian surface is known, the light direction can be least-squares estimated by using occluding contours, which will have the third component normal equal to

[1] [13]

zero. By doing so, however, if there is not previous knowledge on surface albedo and light source intensity, as explained in [13], the z-component of the light vector cannot be estimated. Since the Lambertian model does not work well contours, so extrapolation is performed along a direction perpendicular to the edge, which has an ellipse-like shape. This method, although does not use a known object in the scene, makes strong assumptions on the geometry of the image.

Althought these methods have greater error margins than the others, they are preferred on hardware application since they do not require a large memory to run.

Reflectance Transformation Imaging

Illuminants and light estimation is a fundamental step of the process for surface reconstruction, both for scene modelling like in AR application or simply for storage. There is the need to reconstruct a specific structure in a virtual enviroment in some application field, for many different reasons like the impossibility of moving the object. Of course, a image recording of a particular object is sufficient in most cases, however is not always like this.

This is particularly true for archeology, since many further information can be extracted by observing peculiarities and markings on many archeological remains. Moreover, the could be finds which cannot be carried to a lab or cannot be moved, and so having a good acquisition which can be examined carefully from a reliable reconstruction. So a peculiar acquisition technique was designed for aiding the archeologist.

Reflectance Transformation Imaging(RTI) is used in this case, which exploits multiple images of the same object and with a particular artificial illuminant. RTI is a photographic computational method, which permits a re-lighting of the saved object from multiple angles and enhances the object's surface features and colors Is a shape-from-shaping solving algorithm, which reconstruct the surface of an object by using light direction and surface normal.

The basic idea around RTI, as explained in [14], is to create a composite image from a series of pictures of the same historical find, illuminating the object with different lights, each coming from a well-known direction. Then the surface is reconstructed by a calculating the normal of a surface starting from the reflection of the light.

However, some constraints have to be placed into the scene in order to take valid acquisition and ensuring them in an archeological dig or any open area is not an easy task. In order to function properly, the light direction must be known, but is difficult to set a light in a precise direction by hand. It can be calculated, but the algorithm should be fast and must be easily applicable, with the least number of elements involved.

This can be done by using two reflective targets, more specifically two spheres, the software used for the computation of RTI is capable to estimate the light direction.

By calculating the light direction, RTI can reconstruct the 3D image, with each feature's visibility improved. Of course, this algorithm is applicable only to a subset of all possible artifacts, with respect to surface material, and in order to function properly the artifact must have a certain size with respect to the light involved. Since those acquisition are often made "in site" and no previous data are available,

a statistic-based method based on [2] is used, althought is faster since only the direction is needed in this case.

Figure 1.2: Practical application of RTI on the field on an Irish gravestone, photo by Terry Collins, 2015, taken from [14]. Note that in this case the sphere are located at the base of the grave.

1.2.3 Learning-based methods

One drawbacks of the statistic-based methods is that the assumptions and the gimmicks used limit the applicability of such techniques. However, illuminant estimation remains an ill-posed problem if no constraint is given. In order to solve the problem, it is possible to extract the contraints directly from the image, "learning" how the illuminant acts on scenes and how the scene is affected by different illuminants.

When there are a lot of previous data available learning-based methods can be used. The assumption made in this case is that the information on the lighting can be learned from a *training phase* of an illumination estimation model. There are different types of learning, as stated in [9], but *deep learning* became the state-of-the-art for computer vision.

In [9] a method which tried to give a different approach for achieving *color constancy* was studied. Color constancy is a fundamental step performed on any digital camera's image formation pipeline, since it "removes the illumination's color on the colors of object in the observed image scene"[2].

This method has been developed by using three classes of light models, for each class a specialized estimator is used to estimate the light. The classes are:

1. Outdoor pictures with the sun as illuminant.
2. Outdoor pictures with both natural and artificial illumination.
3. Indoor pictures with only artificial light sources.

[2] [9]

The datasets present in the field of illuminant estimation are not sufficiently big, so a pre-trained network was used in [9].

After the image illuminant is classified, a specialized convolutional neural network is used to perform illumination estimation. It uses a classificator which can distinguish between informative and useless parts of the image. Three instance of the same convolutional network has to be trained on the three class of images in order to optimize the precision of the method.

Another application of a deep learning method is presented by Kafumann and Kàn in [15], which is an approach specific for the Augmented Reality(AR). It uses an RGB-D image to estimate the light position. A RGB-D picture is an image where, in addition to the standard RGB image, a depth image is added, which is a particular image channel where each pixel is related to the distance image plane-object in the RGB image. In other words,it adds information on the depth of an object in a scene.

In AR, as referenced in [15], an estimation of the world light source is essential in order to the rendering of objects in the space and also for visual coherence. The two main categories of methods for AR are *probe-based* and *probe-less*, the former uses a particolar object with well-known reflectance properties, while the latter eludes the usage of specific objects and estimate illumination from a main AR camera image.

The function to be learned represents the "relationship between input RGB-D image of the scene and a dominant light direction"[3].

The dataset comprehends a series of images with well-known illuminants working with the assumption that there is only one dominant light in a scene. The dataset is used to train a neural network. After the training, this network becomes capable of estimating light sources during rendering of an AR scene not seen by training.

The estimation process is done by regressing a dominant light direction using Euler angles, one for the light and one for the camera one in the AR. However, by doing this problem arises when dealing with various camera poses, and so the estimation has been made independent from camera pose. A transformation to the world reference frame of the light estimated must be done in order to have the precise 3D coordinates of the light.

Whilst these methods are very precise, they require a good deal of memory in order to process, and so they are often used in Augmented Reality applications on big hardware or as input for lighting modelling applications.

1.2.4 Gamut-based methods

Most of statistics-based methods relies on image features like pixel intensity and colour, however to use color is implicit that colour is assumed to be an inherent property of the color. As stated in [16], colour depends both on illumination and physical properties of the object. Many approaches using gamut mapping combines a learning phase with a study on the image colors. Gamut approaches, as explained in [16], do not require precise knowledge on surfaces or illuminant, minimizing assumption and simplify the training of the model.

Some researchers, such as Forsyth in [17], tried minimize the assumption on the scene and working with the gamut of the colours in an image. Given a set

[3][15]

of possible colours observable under a reference light, which is a *bounded convex set*. The estimation of scene illuminant was converted into a problem of finding the specific mapping which relate the image gamut to the "ideal" gamut.

Modelling the illumination change as a diagonal model, as explained in [17], so the relationship between different lights becomes a scale factors. The illumination estimation is performed in two steps first determining the mappings between image gamut and the canonical one, and then selecting the most prosimising with respect to a specific criteria. This algorithm was denominated **CRULE**.

This approach was then refined by Finlayson and Hordley [16], since modelling illumination change as a diagonal model is a strong implication, which when not satisfable would lead to no solution. This was solved by defining a priori set of possible light sources, which restrict the solution set and is more reliable.

1.2.5 Novel approach based on level curves

The purpose of this work is explore the possibility of improving the precision of an already functioning stereo system used for quality control, by estimating the position of the artificial illuminant used. Illuminant estimation is performed using a novel approach for light direction estimation. The particular feauture exploited in this method are the curves defined of a particular intensity extracted onto the reflection of the light on a Lambertian spherical object with known position and radius.

The illumination estimation technique retrieves the illuminant geometric parameters, not for shape-from-shading purposes nor for making the system achieve colour constancy, but in order to inspect if the precision of the triangulation of particular painting defects can be improved by knowing the precise light position and the illuminant vector.

For this approach, light is modelled as a point in space, which irradiates the scene in a cone-like structure. Light directions are estimated by using two different approaches, both exploiting "fitting" of a particular geometric structure using a set of points.The precision of both approaches is explored.

The main idea is to consider points which are placed on a particular intensity *isocurve* on the light reflected by the sphere, using a reasoning similar to level curves used in topography.

This techinque is similar to [6], [18], where only a single sphere is considered.

By using multiple sphere positions and images on a specific camera, a series of illumination vectors is obtained, and those "rays" intersects near the real light position.

1.3 Theoretical background

Computer vision's objective is to recontruct and understand a 3D scene by its imaging in 2D, using some properties of the structure present in scene. While in computer graphics light is needed in order to correctly image an object, what is wanted here is to estimate the light position from a 2D image, so it is basically an inverse problem.

This can be seen in 1.4, the point P is imaged to point p', which is represented by the intensity of the pixel. That intensity if certain conditions are met depends

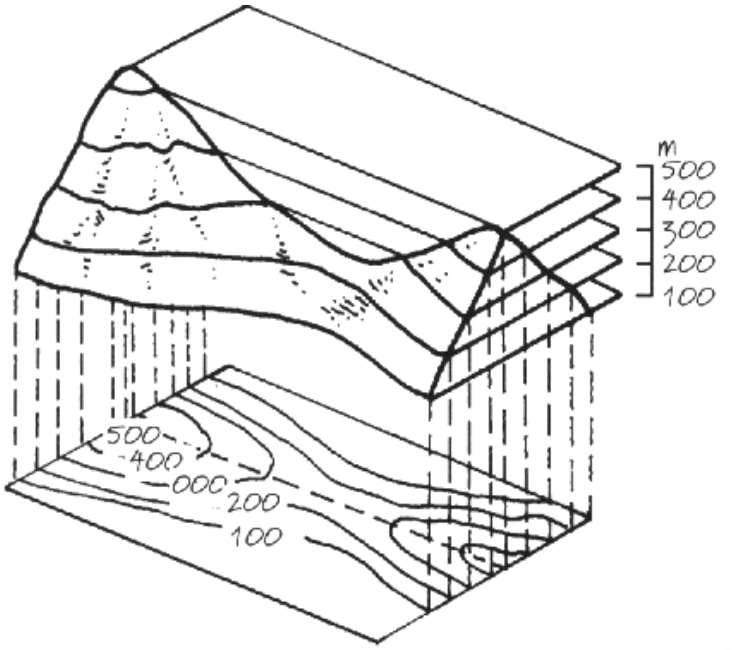


Figure 1.3: Example of level curves in topography

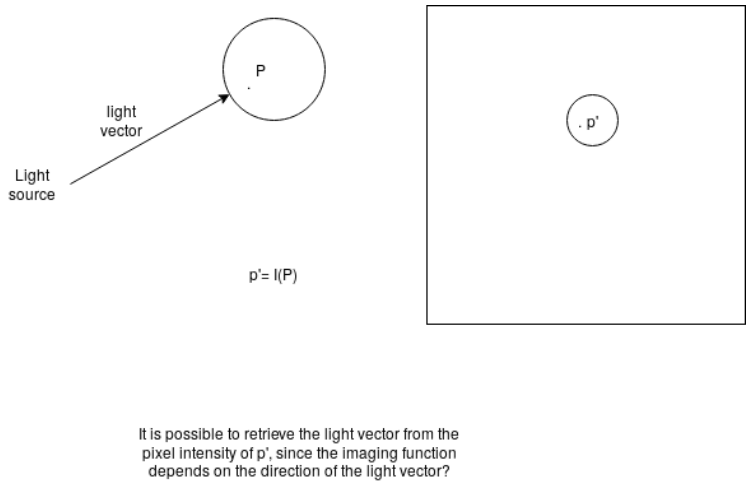


Figure 1.4: A visual representation of the problem of light estimation

only on the object's surface and the light vector direction and magnitude. So the problem is retrieving the light direction which brought the pixel to have its intensity.

One of the most used assumption made for illuminant estimation that makes the problem solvable is the *Lambertian property* of a specific surface.

1.3.1 Lambertian surfaces

A Lambertian object has a particular surface, which respects one specific property:the perceived brightness of that surface does not change with respect to the

angle of view(camera pose) of an observer. This is an important assumption, since it is known that a particular scene is imaged by camera by capturing the light reflections on the object.

By assuming that the object has a Lambertian surface, it will clearly define how that object is imaged, since the intensity of pixels belonging to a Lambertian object will depend only on two thing: the normal of the surface and the illumination vector. In particular, the intensity I_d of a pixel is defined as

$$I_d = L * NCI_l \tag{1.1}$$

where L is the normalized light-direction vector, N is the surface's normal vector, C is the color and I_l is the intensity of the incoming light.

The normal of the surface is strongly related to the light vector by the **Lambert's cosine law**:

$$L * N = \|N\|\|L\|cos\alpha = cos\alpha \tag{1.2}$$

where α represents the angle between the two vectors. That formula basically states that the intensity will be maximize when $cos(\alpha)$ in Eq. 1.2 will be equal to zero, hence when both L and N have the same directions.

In other words, the regions of pixels which will have the greatest intensity are the one where the illumination vector is perpendicular to the surface of the object imaged. By assuming an object to be Lambertian features of the object can be exploited in order to infer some properties of the illuminant.

However, by only making assumptions on the scene's object there aren't many things that can be done to make assumption and estimation on the scene. There is a need to know the properties and mechanism of the object which projected the surface on the image: the camera. However, there are many cameras with different parameters, such as different resolution or different focal length. So it is needed to know how a camera images the actual objects onto the image plane. Most of the time Computer graphics image are produced using the same model, which *pinhole camera*'s one.

1.3.2 Pinhole camera model

The pinhole camera model expresses the relationship between a point in the scene, which has 3D coordinates, and its image on the picture in 2D dimension.

Since the imaging device takes light from all the point in the scene, it is impossible to have a perfect reconstruction. In order to have a better image, the idea is the same as Leonardo's camera obscura experiment, that is putting a barrier with a hole between objects and sensor. Of course, the size of the hole is relevant, since a big hole will let pass more light and the image will become blurred, whilist a small hole will force the object to be exposed for longer periods of time.

In order to avoid the hole problem, lenses are used, which will take all the parallel light reflections of the objects and will converge them into a single points. However, the lens will create a distortion effect, since the displacement of the projected point will become non-linear transformation. Since the distortion can be modelled, using radial distance from the lens center, this is not a big problem, but those parameters needs to be estimated in order to perform estimation from 2D to 3D worlds.

A digital camera can be assumed to work like a pinhole camera, and so the relation between 2D and 3D points can be modelled. However, most of the time this relation is not easily described, since most transformations are not linear. By using **projective geometry**, which is an elementary form of geometry which "expands" the Euclidian one it is possible to describe the pinhole camera model with linear relations.

1.3.3 Projective geometry

The 2D Euclidean space can be expanded by adding a dimension to all of its points, which creates a new space which is called *projective space*

$$\mathbb{P}^2 = \mathbb{R}^3 - \begin{bmatrix}0\\0\\0\end{bmatrix} \quad p = \begin{bmatrix}wx\\wy\\w\end{bmatrix} \in \mathbb{P}^2 \; with \; w \in \mathbb{R} - \{0\} \tag{1.3}$$

This allows to convert operation of points into linear operations because of the higher dimensionality. A point p expressed like in equation 1.3 which belongs to $\mathbb{P}^{\not=}$ has *homogeneous coordinates*. An homogeneous points correspond to only one Euclidean point, which is obtained by dividing all its coordinates by the last one(w) and considering only the first two coordinates. An Euclidean point however is described by infinite points in the projective space. Those points are described as an *equivalence class* of points precisely, where only the last dimension differs.

By using this peculiar geometry, a planar projective transformation can be represented as a linear transformation involving a 3x3 matrix H:

$$\begin{bmatrix}X'\\Y'\\W'\end{bmatrix} = Hx = \begin{bmatrix}h_{11} & h_{12} & h_{13}\\h_{21} & h_{22} & h_{23}\\h_{31} & h_{32} & h_{33}\end{bmatrix} \tag{1.4}$$

This geometry can also be applied to the 3D space,with the difference that the vector will be four-dimensional. In this particular case, the linear projective transformation will be represented by 4x4 matrices in the form:

$$\begin{pmatrix}A & t\\V^T & v\end{pmatrix} \tag{1.5}$$

where V^T and t are vectors, A is a 3x3 invertible matrix and v is a scalar.

Both geometries are very important, because they permit to describe the pinhole camera model in a simple way and expresses and using linear transformation.

All modern imaging models can be approximated to the pinhole camera model. By doing so, the relation between 2D and 3D points is known and inference of 3D properties from 2D images becomes possible.

In particular, what is done by modern cameras is consider a virtual image plane in front of the center of projection, as can be seen from figure 1.5. The point p' represents the projection on the image plane of the top of the sphere. The point c is called *principal point* and all coordinates of the points in the image plane are expressed with respect to the point. The actual relation between a point in $\mathbb{P}^3$ which

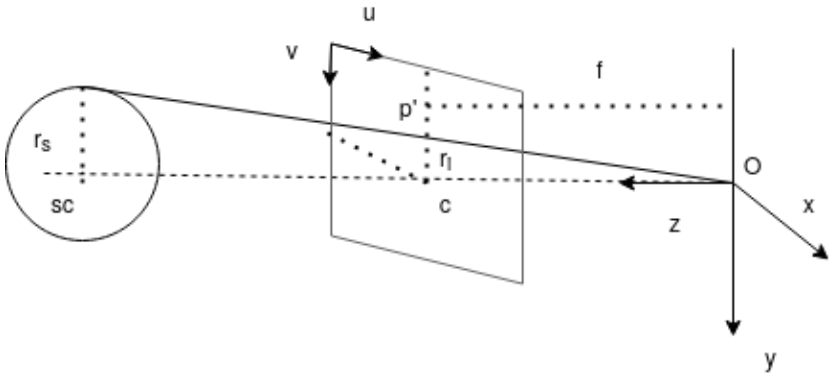


Figure 1.5: An image which shows how a pinhole camera model image an object on the image.

has been projected in $\mathbb{P}^3$ is:

$$p' = \begin{pmatrix} fx_p + c_x z_p \\ fy_p + c_y z_p \\ z_p \end{pmatrix} = \begin{pmatrix} f & 0 & c_x & 0 \\ 0 & f & c_y & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} x_p \\ y_p \\ z_p \\ 1 \end{pmatrix} \tag{1.6}$$

In order to exploit projective geometry on an image, however, there is a need to know the parameters of the camera which has imaged the scene. The camera is considered to work like a perfect pinhole camera, which uses known parameters in order to project a point in the image plane. Those parameters can be divided into two categories:

- **Intrinsic parameters**, which depends only on the camera's specific characterstics.
- **Extrinsic parameters**, which are related to the camera's position and orientation.

Intrinsic parameters

The intrinsic parameters usually are represented by a 3x3 matrix K, and a five-dimensional vector. The matrix K contains the principal point coordinates, which is the last column vector of the matrix and the focal length of the lens.

$$\begin{pmatrix} f & 0 & c_x \\ 0 & f & c_y \\ 0 & 0 & 1 \end{pmatrix} \tag{1.7}$$

The 5D vector represents the polynomial model which represent distortion applied by the lens. By using these parameters, an image can be undistorted as it would be taken with a perfect pinhole camera.

Extrinsic parameters

In order to have a precise estimation of the projected point and its equivalence class, there is a need of knowning the position of the camera with respect to the object. Moreover in complex systems, multiple cameras have to be present, since **stereo vision** is applicable only when multiple cameras are present. There is the problem of considering a common reference system, known as *world reference frame*, where all points of the scene can be reconstructed.

This world reference system must be common to all cameras, which will have a specific pose. The extrinsic parameters are none other than the elements of a **rigid motion** which translates points from the camera's reference system to the world one. So if a camera A is selected as center of the world reference frame, $O = -RT$ is the position of the camera B's center with respect to camera A. R is referred to as *rotation matrix*, while T is denoted as *translation vector*.

By knowing all the parameters of the camera, the projection of a point p to the image point p' is expressed as

$$p' = K(RT) \begin{pmatrix} x_w \\ y_w \\ z_w \\ 1 \end{pmatrix} \tag{1.8}$$

Since the projection of a point is a linear function, it is also invertible, and so we can back-project a point on the image onto the world reference frame.

A 2D point is the imaged version of a 3D point, which has been projected on the image using the projection matrix. Consider that point, an equivalence class of points are mapped to the same point of the image plane. The collection of these points is represented as a ray which is projected in space by the camera center and the specific point p.

This can be done by considering the pseudo inverse of the projective matrix, denoted as P^+,which will define the equation of the ray:

$$C + \lambda P^+ x \tag{1.9}$$

whew λ is a real coefficient which represent the exact position of the considered point on the line and C is the camera center. The back-projection is very important when trying to perform stereo vision, since it can add information on depth of object and when rendering. In order to use projective geometry to retrieve 3D properties, the camera's parameters must be estimated first.

1.3.4 Camera Calibration

In order to exploit the a pinhole camera model on the imaging device, all camera's intrinsic and extrinsic parameters has to be estimated. This can be done by a process known as **calibration**, which can be done in many different ways, considering what type model is used, what is exploited and the linearity of the such procedure. The most general classification which can be done for calibration techniques is:

- **Photogrammetric calibration**: It implies using a known 3D object, that must have easily detectable feature. This involves a planar object and a "elaborate setup"
- **Self-calibration**: Unlike photogrammteric techniques, they do not require a *calibration target*, just a static scene recorded by the same camera from multiple position.

In [19], all different models are explained and compared, however, as stated, "there is at yet no accepted one step method that is either reasonably universal or amenable to full automation".

However, the method explained by Zhang in [20] became the most reliable solution and it is considered a standard. This particular approach exploits a real chessboard as target, since its axes and corner can be easily detected. By having the same chessboard imaged from multiple position by the same camera, with known 3D coordinates of points, it is possible to infer the projection properties. This is because many 2D point- 3Dpoint correspondence will be available, where the matrix of the intrinsic K does never change.

Starting from a single plane, rewriting the modelling of a point in 3D M and its image projection m becomes possible. By denoting the rotation matrix R using its column vector[20]:

$$s \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = K \begin{bmatrix} r_1 & r_2 & r_3 & t \end{bmatrix} \begin{bmatrix} X \\ Y \\ 0 \\ 1 \end{bmatrix} = K \begin{bmatrix} r_1 & r_2 & t \end{bmatrix} \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} \tag{1.10}$$

The matrix which multiplies the 3D point can be interpreted as an **homography**, which is a planar transformation which preserves straight lines, which can be estimated by having a necessary amount of point-point correspondeces.

The target is a chessboard and the points used as exploit are it's angles, which are good feature and are easily detectable. Starting from these point-point correspondence, the instrinsic parameter of the camera can be retrieved. Also, the extrinsic parameters pf the camera can be derived, having as world reference system the chessboard one. Of course, the estimation of the extrinsic becomes possible after the intrinsic have been estimated.

By calibrating intrinsic and extrinsic parameters of each camera involved in a system, the position of object in the real scene can be guessed from just its image.

1.3.5 Triangulation

The triangulation of a particular point in a scene, that is estimating the 3D coordinates of a pixel which has been imaged in a picture, cannot be done with one single image. Starting from a 2D pixel, it is impossible reconstructing the point which has been imaged at that location, since no depth information are available, and so infinite points X can be imaged to a point x'. Moreover, the point of interests has to be extracted on the image, exploiting particular knowledge on objects or on the image characterestics.

Using the camera's parameters and the 2D points on multiple camera poses, the 3D position of the sphere center can be estimated. Naive triangulation, which involves finding the intersection of the back-projections of the 2D points from two images using *epipolar geometry*, is not a good choice, since the rays will be skew and will never intersect in reality.

The triangulation can be performed in many ways, but the most common are:

1. Find the midpoint of a line perpendicular to the two back-projections, which is called **Midpoint triangulation**.

2. Since both the points in 2D were projected on each image plane using the respective projection matrix P(if the point is not a point at infinity), it is possible to decompose each component and build a linear system, which using homogeneous coordinates is equivalent to solve:

$$min_x||AX||s.t.\ ||X|| = 1 \tag{1.11}$$

   By doing so, the point can be retrieved by solving this system.

If the image is undistorted, it is possible to use a linear triangulation method explained in [21] which exploits the Direct linear transformation(DLT) algorithm in order to solve the system. DLT is an algorithm which solves systems of linear equations described as a set of similarity relation [22]:

$$x_k \propto Ay_k\ for\ k = 1, ..., N \tag{1.12}$$

where A is the matrix which contains the unknowns to be solved. By doing this the mapping between a set of 2D points to 3D becomes linear. The eq. 1.11 derives from the fact that by having two points in the same image which represents the same point X in space, the two equation $x = PX$ and $x' = P'X$ can be related to each other. By expanding the relation as a system for each coordinates, four equation are involved which can be summarized as:

$$A = \begin{bmatrix} xp^3T - p^1T \\ yp^3T - p^2T \\ x'p'^3T - p'^1T \\ y'p'^3T - p'^2T \end{bmatrix} \tag{1.13}$$

where p^nT is the n-th column of the projection matrix.

By solving this system, a solution is found, however since it minimizes the algebraic error which has no geometrical meaning, the estimate is not very precise. A possibility of improving the estimate is to consider the DLT solution as a starting point for a non-linear optimization algorithm, which may minimize the reprojection error of the point on the image.

1.3.6 Feature points

Even if the points can be triangulated with a decent precision, in order to position objects in the space and make solid assumptions, there's the need to undestand what object is imaged first. Recognizing a specific object in the image space is a

difficult task, which cannot be done on the triangulated point since it will increase the computational burder.

Usually, the information extrapolation is done on the image by localizing some meaninguful points, that are called *feature points*. A feature point is local part of the image which is invariant to viewpoint or illumination, resulting in a point which contains a lot of information. It can be said that a feature point is a locationof *sudden change* in the image.

A good example of feature are **edge pixels**, which reduces the information required for the inspection(few pixel in a binary image are more than enough) and their causes can define the structure of an object.

Edges can be detected by looking at the pixel intensity. Since an edge can be derived from a reflectance discontinuity or another interesting causes, in the proximity of an edge the pixel intensity should change a non-trivial amount. So the edges are detected by locating those areas where the intensity rises or drops, given a certain threshold. However, the threshold must be setted wisely. If an incorrect threshold is setted, *false edges* might be detected. Another thing to keep in mind is the actual change that is considered as "abrupt". If the intensity of an object changes gradually, the edges might be measured incorrectly and result thick, so a proper heuristic in those cases has to be setted. With regard to the different type of changes, three types of edges can be identified: the *step-edge*, the ramp edge and the roof edge.

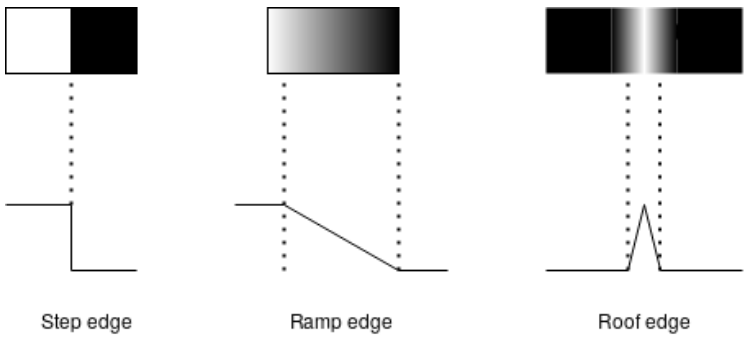


Figure 1.6: Visual example of the three kinds of edges.

A good idea is to use the derivatives of the image, since it highlights very well these zone of change. In particular, if we use the *gradient* of the image at each pixel as indicator of a possible edges, which is equal to the first derivative in 1D scenarios, it will be greater than zero along all the slow increase/decrease in intensity. Using the second derivative, i.e. the *Laplacian in 2D*, will avoid this problem, since it gives a better response if the change is gradual, however the more the function is derived the more impactful noise will be on the result. Moreover by using the second derivative there will be two responses which must be deambiguated using *zero-crossing*.

In order to solve all this problems, many heuristics and derivatives approximations where designed, until an algorithm with a solid mathematical background became standard.

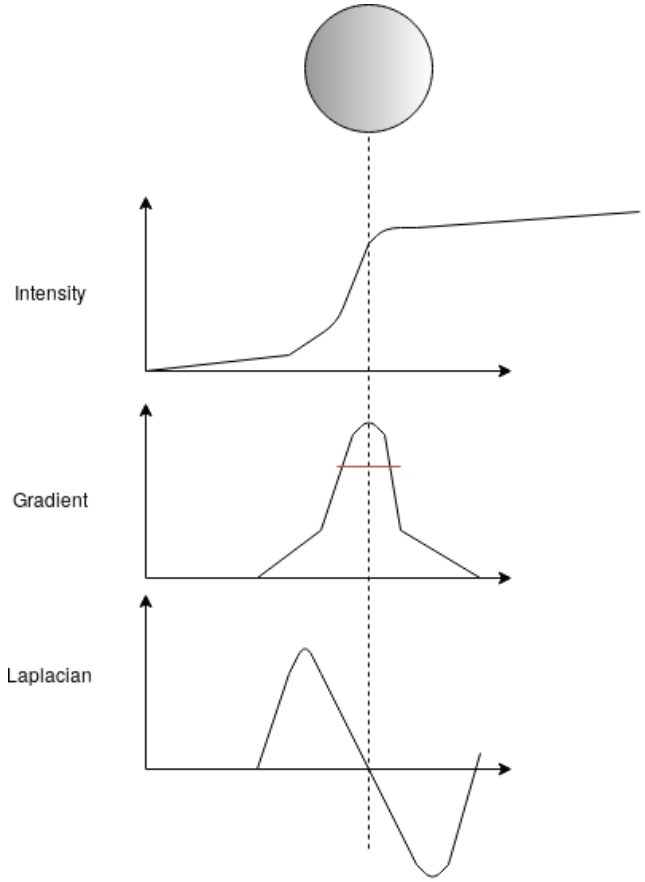


Figure 1.7: Example of the difference of edge detection using first and second derivatives. In the gradient case a threshold is setted in order to extract the edge, and the actual edge becomes thick, while using the laplacian the edge is identified at the zero-crossing.

Canny Edge Detector

Canny's edge detector became the standard solution for edge detection problems. This techniques relies on a strong mathematical background and the good heuristics implemented for thick edge trimming. Canny in [23] defines two objective function, which will be called for Λ and Σ, given a filter f.

- Λ(f) will be large if f produces a good localization for the edge point.
- Σ(f) will be large if f produces good detection.

The goal here is to maximize the product $\Lambda(f)\Sigma(f)$, with the constraint that only one peak peak should be generated at a step edge.

Canny used as indicator of possible edges the *Derivative of a Gaussian*(DoG), since it is a good approximation of the optimal filter and it's a linear operator, so each gaussian and derivative can be applied separately on the image as **convolutions**[4],

[4]mathematical operation between two function which expresses how the shape of the first operand is altered by the second

image smoothing using a 2D Gaussian filter and afterward computing the gradient on the smoothed image. The DoG is so good because it suppress the noise before applying the derivative, because the noise impact on the result will increase the more the function is derived.

This step will generate pretty good results but will have thick ridges around a peak, since first derivative is used. In order to trim the edges and eliminate false positives, two particular heuristics are used, namely *non-maxima suppression* and *hysteresis thresholding*, These techniques are performed on the result of the DoG, non-maxima suppression selects a single maximum point across the width of an edge, while hysteresis thresholding uses two limit values for separating "strong" and "weak" edges: only weak edges connected with strong ones will be considered, eliminating false edges.

By performing the edge detection on a picture, a binary image returns, where at each pixel position 1 indicates the presence of an edge pixel. If the object has a specific geometrical structure which can be generated by a mathematical model, the model parameters can be estimated, since it is possible to generate a curve which explains certain edges.

1.3.7 Model fitting

Fitting a mathematical model into an image implies solving some subproblems, with respect to what information is available:

1. **Parameter estimation**, which is the recovery of the parameter of a specific model which generated a specific segment in the scene.

2. **Token-curve association** which is the relation between a specific curve whose parameters needs to estimated and the edge segments.

3. **Counting** of the curves in a scene. It basically resolves the problem of how many object which are describable by the model needed to be fit.

If no prior knowledge on the data is available, all three of this problem has to be solved. Counting is solvable by running the fitting on all the connected edges on the image. However, if the model is not threshold properly, false positive might be present, so generally either the shape are clearly recognizable or the number is known beforehand.

For the token-curve association problem, two of the main approaches are:

1. Consensus-based approaches, such as **RANSAC**.

2. Voting schemes using accumulators, like **Hough transforms**.

RANSAC algorithm

RANSAC(RANdom SAmple Consensus), first explored in [24], is an iterative probabilistic algorithm for parameter estimation, starting from a set of data. Being a probabilistic algorithm, it does not guarantee to return the correct results.

The main idea is to random sample the minimum number of points which defines the model which is wanted to be fitted. Then, according to this generated fitting all

the points are tested according to a loss function, and will marked as *inliers* if they accept the current model, *outliers* otherwise. The model chosen is the one which will have the biggest number of inliers.

In order to maximize the probability of choosing the correct model, the maximum number of iteration can be estimated. Assuming that e is the probability of a point to be an outlier, $(1-e)$ will be the probability to be inlier. Given the number of minimum points to fit the model s, $(1-e)^s$ is the probability that all the points contained in s are inliers and e^s indicates the probability of having an outlier in s. Varying that probability with the number of iteration N will give the probability that the algorithm won't ever choose a set of points which will be all inliers, which is $1-p$. More precisely:

$$\begin{aligned} 1-p &= (1-(1-e)^s)^N = p \\ 1-(1-(1-e)^s)^N &= p \end{aligned} \tag{1.14}$$

Using the logarithm and rearranging the elements of the equation 1.14:

$$N = \frac{log(1-p)}{log(1-(1-e)^s)} \tag{1.15}$$

Hough transform

The main idea of Hough transform is to map points from the image space to the parameter space points.

Every point votes for a specific curve by filling an accumulator in the parameter space. The most votes curve will result in the biggest accumulator. This approach is fast in computation and it is decent precise, however it has some problems.

This method will become incovenient to be applied if the parameters of the model are more than three , because the space will become too high-dimensional and sparse, so the correct solution is not easy to be found and noise will be amplified.

The dimension of the parameter space is discretized by using the range of values that the parameter can assume.

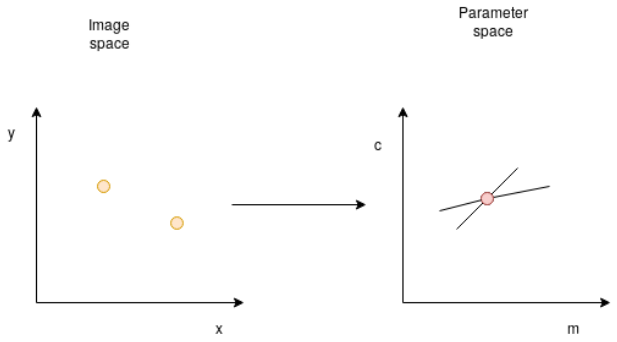


Figure 1.8: Visual example on how hough represents point in the parameter space.

In figure 1.8, it can be seen how Hough works. Consider two points a and b with coordinates (x_1, y_1) and (x_2, y_2), their representation in the parameter space is a line expressed with respect to the constant c and the the slope of the line m. The

point of intersection of those two lines in the parameter space its actually a line in the image space, which is the best fitting line for those points. The parameter space is actually discretized into a matrix of "bins", which works like an accumulator, and the best line will be the one which accumulated the most votes.

1.3.8 Structure of the dissertation

This thesis is divided as follows: in Chapter 2 an overview of the method is presented, in the third chapter the methods exploited for the triangulation of the spheres are explained. Chapter 4 exposes the technique used for the extraction of the level curve at a certain intensity, while in Chapther 5 the method of estimation of the normal of the plane described by the isocurve is explained. Finally, in Chapter 6 the results of the method are presented.

Chapter 2

System setup and method overview

The goal of this work is to estimate the light position from a large dataset of pictures of a Lambertian spherical object, which is illuminated by twenty different lights. The system considered is an already functioning industrial quality control camera system, used by Electrolux company at the end of their assembly line.

The machine inspects ovens which underwent a painting process. By taking multiple pictures of the oven using a camera system, painting errors and defects can easily be detected and triangulated.

The stereo system is composed of five different cameras placed in a rectangular shape, four at the side and one at the center. The side cameras reside on the same plane, while the center one is deeper than the others. The system comprehends a illumination system composed of twenty lights and a real sense. The real sense is an object which implements a particular technology which adds depth information to images. To do this, it uses a hybrid sensor which mix CMOS with infrared, a Micro-Electro-Technical(MEM) devices, which projects a invisible light ray in order to perceive profondity and a CPU. The real sense, however, has not been used for this work and its presence is irrelevant.

Four lights, which are encoded from zero to three, are lines structures composed by led light, while the other sixteen are flashlights, which are the ones whose position will be estimated.

The system operates in a closed room and the ovens passes over the cameras. Multiple acquisition of the same objects are made, in order to increase the robstness of the triangulation. The cameras are logically numerated from 0 to 4, as descripted in 2.2. This will be important when talking about triangulation and the results of the method.

2.1 Calibration

The intrinsic and extrinsic parameters of the camera had already been estimated, since they were necessary for the functioning on the system. Regarding the intrinsic parameters, a method very similar to [20] was used.

The extrinsic extracted using Zhang's algorithm, however, had to be perfected

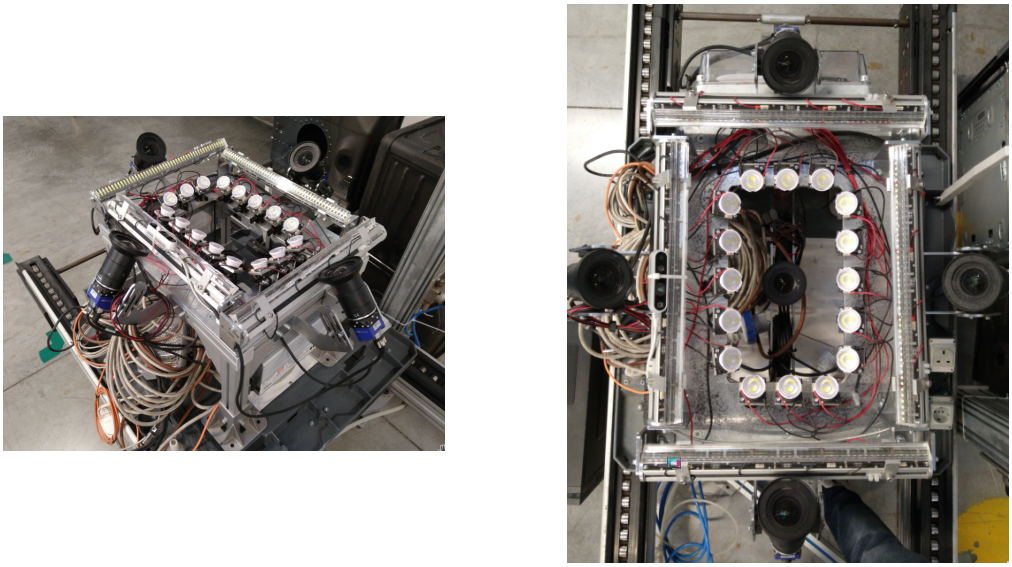


Figure 2.1: A couple of pictures representing the system

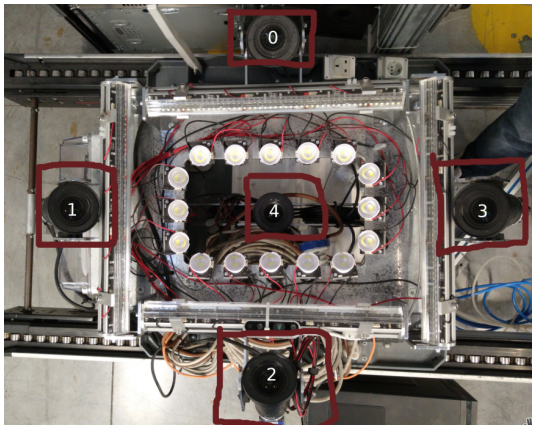


Figure 2.2: The coding of the cameras with respect to their position on the machine.

and mediated, since every projective transformation on the world reference frame had to make sense. In order to estimate the poses and mediate them, the technique explained in [25] have been used, which exploits *dual quaternions*, which are a special algebraic structure, and a specific planar target.

The method works in two steps:

1. A *view-graph* is build, with respect to the different poses of the camera on the scene.

2. By performing multiple acquisition of the marker, all transformations from marker's world to camera one are placed in one big graph which is then widespreaded.

2.1.1 View-graph construction

The view graph represents the various transformation needed to move from a reference system to the other. The nodes are the camera themselves with the real sense while the arches represents the rigid transformations in order to move from camera x reference system to camera y's. The poses of each camera are retrieved by taking a picture of the marker by all camers, which has its own reference system. The marker in this case is a chessboard, which is a good target since it has easily detectable corners. By detecting the corners of the chessboard, which has known 3D coordinates, a set of image point-real point correspondences as well as the intrinsic parameters of each camera are available, so the only thing remaining to estimate its the extrinsic parameters.

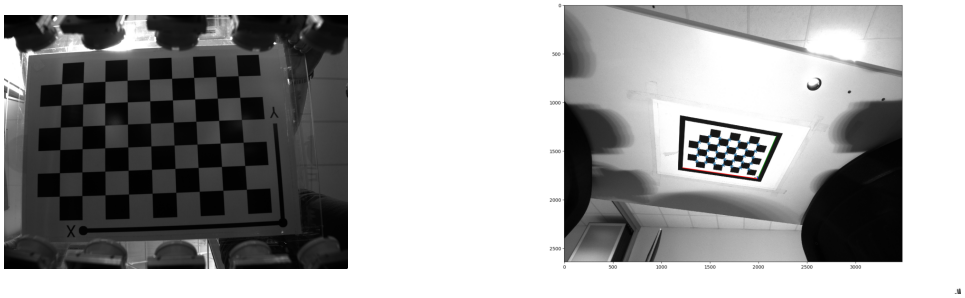


Figure 2.3: Two of the images used for, respectively, instrinsic and extrinsic parameter estimation. For the extrinsic, the points indicated on the cameras represents the corners which have known coordinates, which are used as correspondeces.

The problem of estimating the rotation and translation of a specific camera by a set of correspondences and with known intrinsic is known as **Perspective-n-Point**(PnP) problem.

In order to solve it, many methods have been developed, for example Gao in [26] gave an optimal and reasonably precise solution with only three correspondences, while in [27] an optimal solution to the general problem is given. As can be seen seen in figure 2.3, the correspondences are more than three, so Gao's approach is not suitable.

By solving PnP the pose of each camera is retrieved, and the relative transformations deduced by the pictures are composed in order to build the view graph.

2.1.2 Graph diffusion

In order to have the most robust estimate of the poses, multiple acquisition of the marker were made, and a big view-graph was built. In order to retrieve the best estimate of the extrinsic for each camera, a process of diffusion was made on that graph.

The view graph considers the edges as transformation from a reference frame to the other. As stated in [25], V_i a transformation from camera N to the common world reference frame, and as T_{ij} the transformation for camera N reference frame to camera M. For all adjacent view, it must hold the equation $T_{ij} * V_i = V_J$ in order for the transformation of the reference frame to be coherent. Since in reality this

equation is never exactly fullfilled, the goal is to minimize the distortion of that operation, that is:

$$D = \sum_{i} \sum_{j \in N(i)} d(T_{ij}, V_j, V_i) \tag{2.1}$$

The measure of distortion considered in [25] refers to the **Chasles' theorem**, which states that any rigid motion is equivalent to a rotation around a *screw axis* and a translation along it.

The rigid motions are not expresses as the classical rotation matrix and a translation, but by using a particular algebraic construct known as *dual quaternions*.

By diffusing the dual quaternion in the view-graph with respect to the minimization of the screw distortion, the optimal extrinsics of the camera are retrieved. Of course, the resulting network of the cameras will have a star-like topology, which center corresponds to the world reference frame chosen. In this case, the real sense reference frame is selected as world, and all other cameras' extrinsics consider the real sense as origin.

2.2 Method overview

Since the objective is retrieving the light starting from the image, it can be seen as an inverse problem with the same idea of shape-from-shading.

The light assumed in this model is point light source, since it is a good approximation for the real scene illuminant given the small distance between light sources and illuminated objects. which irradiates the scene starting from a point in the world but outside the scene visualized. Moreover, the Lambertian surface of the sphere is assumed, in that way the intensity of a pixel will depend only on the surface's normal and the illumination vector.

The main idea is to retrieve the light position by estimating multiple light vectors of the same light source. Since they all came from the same source, their approximated intersection should be a good estimation of the light position. It is approximated because since there might be noise on the image the rays will result skew in the 3D space and might never intersect. Instead, the closest point between them is selected. The multiple directions are collected by changing the position of the sphere in space during the acquisitions.

The sphere parameters are not necessarily known, since the position of the sphere can be triangulate using the camera's extrinsic parameters, which are known, and a couple of images from different cameras. The radius can be calculated by geometry relations between the projected "sphere" radius and the real one, since the camera work following the pinhole model.

The method described in this work can be divided into various steps. For starters,the sphere is located in the image, extracting the edges from the image and fitting a particular curve on the edge pixels. Since a curve is invariant to the projective transformation, it will be projected on the image as an ellipse, however an ellipse is difficult to fit, since the distance point-curve is not easily defined. The ellipse is approximated to a circle, since the error which derives from the approximation can be trascurated in the estimation of the sphere center. A circle is fitted on the edges detected by Canny's algorithm, using two different method:

- The consensus-based approach RANSAC.
- A voting scheme using Hough transform.

For the latter implementation, the standard Hough transform is not suitable for the fitting of a circle. Since Hough uses a parameter space, the more parameter a specific geometrical model has, the bigger the parameter space used by Hough will become. This leads to a sparse, error-prone space and impacts the effectiveness and precision of the method. In order to solve this problem, a slightly different method is used, known as **Hough gradient**, which effectively fits the circle using the discretization of Hough.

From the center of the circle which has been fitted, by using a pair of images took from different cameras, the center of the sphere in the world reference frame is triangulated. The extrinsic of the cameras are known and can be manipulated to change reference frame as preferred.

Since the goal is to find an isocurve indicated by a pixel intensity level, a squared cut which contains only the sphere is extracted. This is a mandatory step, since all pixels on the imaged sphere must be inspected and instead of doing a logic partition on pixels, a separate image is considered in order to improve the speed of the algorithm. Given the **isovalue**, the level curve is found by using a particular algorithm, which is the 2D application of the famous Marching cubes algorithm, the *Marching square algorithm*. The main idea of Marching square is to transform the image in a binary one, where a pixel is labelled as one if its intensity above or equal the isovalue, 0 otherwise. Then, the binary image is structured into 2x2 cells, whose vertex indicates the general structure of the contour which passes through the cell. By inspecting all the cells, the imprecise contour is built, which is then defined using **linear interpolation**.

The level curve extracted from the sphere represent the intersection between a plane almost perpendicular to the light vector and the sphere. The next step is to find the plane which intersects the sphere at the points of the contour and find its normal, which will be a good estimation of the light direction. This is done following two different routes, the first implies the fitting of an ellipse which generates the contour, followed by the estimation of the normal on which the ellipse resides.

The alternative route implies the back projection of the contour points in space, which does not exactly converts to a 3D point, since the same point in an image might be generated by infinite 3D points. The back-projection is basically a ray in space, which however must intersect the sphere whose position is known. Since the camera who took the image can be approximated to a pinhole camera and the distortion coefficients are known, it is possible to calculated the radius of the sphere and the intersection ray-sphere can be calculated. The points are then used to fit a plane whose normal will almost correspond to the direction of the light vector.

By taking multiple pictures of the sphere with a different location, all the direction will almost converge to a single point, which denotes the light position on the world reference frame. Finally, the position of the light is then calculated as the closest point to all the normals projected into space.

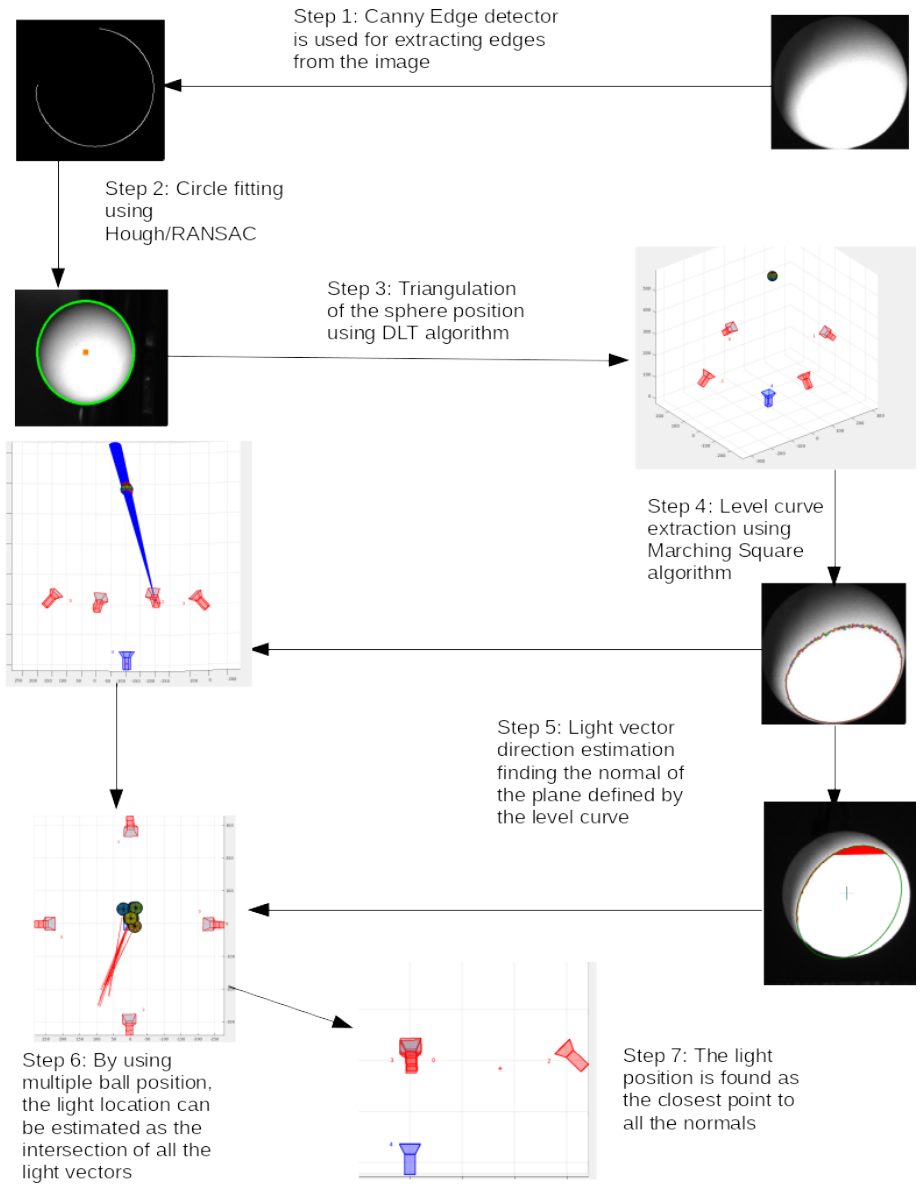


Figure 2.4: A schematical simplified representation of the method. Since two approached were used when dealing with the light direction estimation, there is a biforcation.

Chapter 3

Triangulation of the sphere's position

The triangulation of a particular point in a scene, that is estimating the 3D coordinates of a pixel which has been imaged in a picture, is not an easy task. The back-projection of the imaged point will not work, since no information on depth is given and infinite point can be imaged to that specific point of the picture. By considering the center of the sphere projected into the center of the circle fitted onto the image, a candidate feature point is found.

In order to perform this operation, it is needed to estimate the parameters which descripts the circle on the image, using a process known as **fitting**. In order to triangulate with a decent precision the center of the sphere, two points from different images are needed. By doing so, the triangulation can be performed using the DLT algorithm and the projection matrices of the cameras.

At first a circle model is robustely fitted into the image using two different approaches,then using the centers of the circles in multiple images the sphere center is triangulated in space.

3.1 Circle fitting for sphere center calculation

The most convenient way to triangulate the center of the sphere is to estimate the parameters of the "circle" projected on the image,if it's clearly visible and illuminated. The sphere captured by a pinhole camera model with no distortion will appear as an ellipse in the image,since conics are invariant under a general projective transformation.

However in the images given the ellipse can be approximated into a circle, for two reasons:

- The result of the triangulation of the sphere center is good enough, since the ellipse has so little tilt and "deformation".Moreover, the error derived from this approximation is trascurable.

- The fitting of an ellipse is much more complicated and computationally taxing than circle fitting, because the point-curve distance in case of ellipses is not

easily formulated and approximated, while for a circle that problem does not occur. It's a good tradeoff between precision and computational complexity.

3.1.1 Circle Fitting

The edges of the sphere are detected by Canny's edge detection algorithm and will be represented into the binary image as an arc. The objective is to find the parameters of a circle which fits the edge considered, basically a circle which as that edge as an arc. If the circle is fitted correctly, its radius and its center in the image will be retrieved, and using multiple views from different cameras the sphere center can be triangulated.

Circle fitting using RANSAC

In order to apply RANSAC, it must be known the minimum number of points needed to define the model. In the case of a circle, the minimum number of points needed to define it is three. The general equation for a circle is:

$$(x - x_c)^2 + (y - y_c)^2 - r^2 = 0 \tag{3.1}$$

where x_c and y_c are the coordinates of the center of the circle and r its radius. By putting all three points into the equation, since it is known they rest on the circumference of the circle, a system of three equation in three variables (x_c, y_c, r) is obtained, and so it is solvable with the classical linear resolution methods.

As loss function, the distance point-curve is used, since it is easily defined:

$$d = |\sqrt{(x_1^- x_2)^2 + (y_1 - y_2)^2 - r}| \tag{3.2}$$

The distance d is always positive, so it accepts as inlier points every point closer than r to the center. The smaller the value d is, the closer the point will be to the circle. The threshold applied in this case must be very small, since the edges are not so noisy and a big arc is detected in all the images by Canny's algorithm, in order to have the most correct estimate possible.

Fitting using Hough transform

The other common approach when it comes to fitting models is the Hough transform method. However, standard Hough transform is not suggested for fitting circle, since the parameter space will be three-dimensional as the radius of the circle has to be considered. Instead of using the standard Hough transform, a different method is used in this thesis, called **Hough gradient method**, described in [28].

Usually the standard Hough transform method quantizes and discretize the parameter space, and then a n-dimensional accumulator is used in order to inspect all points.This works very well in case of lines modelled using *Hesse normal form*. In the case of circle fitting, however, it is works less efficiently because of the three-dimensionality of parameter space. The Hough gradient method instead calculates the first-order derivative using *Sobel* operator,which is a linear operator which calculates the gradient using convolution on an image, for every edge point. The points on

parameters space along the line which has as slope the gradient, will be incremented in the accumulators.

Candidate centers will be chosen among the largest one in the parameters space and must be larger than all immediate neighbors. Even by using this exploit, the computation of different radiuses is quite taxing, since all possible numbers need to be considere. In order to limit the computation and bound the set, the maximum radius wanted to be found has to be noted. Moreover, it tells the algorithm the minimum distance accepted between two centers for which the circle is fitted.

The radius selected for each center is the best supported by the other non-zero pixels. The best center selected is the one which has sufficient distance between any previously selected center and the most number of supporting "white" pixels.

3.1.2 World reference frame

In order to have a more coherent reference frame and coordinates, the origin was moved the camera with code four, which is placed at the center of the system. The world reference frame used by the system had as origin the real sense, since it is an important piece of the machine.

A change of reference was needed, since the real sense its place upside down and the coordinates using the starting reference system would have the negative z. So the world reference frame used here is the camera four's one.

If a camera is the origin of the reference frame, its rotation matrix will be equal to the identity matrix while the transition vector will be null. The projection of a point to the image plane in that particular camera will depend only on the intrinsic parameters. For the other cameras, the extrinsic parameters as well as the projection matrix needs to be changed using the extrinsic parameters of the camera four in the real sense reference frame. First, it was needed to calculate the RT which projects point from the real sense reference frame to the camera which had to become the new origin(camera four) reference frame, so the inverse of R, T and the inverse of RT was needed.

$$\begin{aligned} R^{-1} &= R^T \\ T^{-1} &= -R^T * T \\ RT^{-1} &= \begin{bmatrix} R^-1 & T^-1 \\ 0 & 1 \end{bmatrix} \end{aligned} \tag{3.3}$$

The inverse of RT is padded, becoming a 4x4 matrix, because the transformation happens in the projective space $\mathbb{P}^3$ and homogeneous points have four coordinates. Then, for each camera R and T are extracted by the result of the multiplication of the inverse RT of camera four with each camera's RT matrix. This multiplication expresses the transformation from camera four's reference frame to the camera N's one. Then, the projection matrix for each camera is recomputerd using the new extrinsic parameter.

By doing this, a new world reference frame is created, where all successive calculation will be performed.

By considering the extrinsic parameters and the centers retrieved from the images, the 3D point can be triangulated using DLT algorithm.

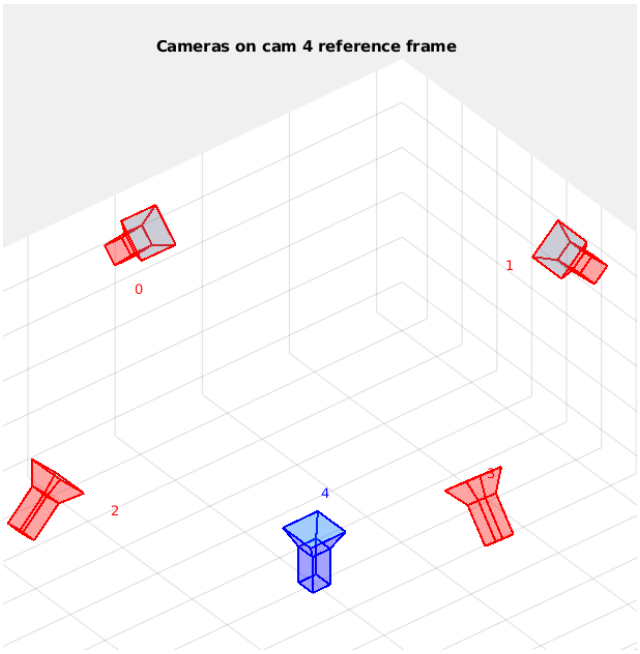


Figure 3.1: A visual example of the world reference system constructed using the camera's extrinsic parameters wrt camera four

Chapter 4

Level-curves detection

In order to estimate the light source, it is mandatory to have an informative region on the image regarding the illuminant. What is usually done is considering the reflection of known objects in the scene, but is not the only way.

One of the possible approach is to use the gradient mapping for light direction estimation, like [29], however this approach is very sensible to noise When using the gradient, the noise will be amplified, so this approach is not suggested if the images are not optimal. In fact, is mostly effective on close ups of a particular sphere with high resolution.

In this case, the sphere is far from the camera and presents a decent amount of noise, so even if the method is still applicable, it is not the best way to go. In order to reduce the noise is it possible to use *spatial filters*, which are linear operators that act on all the pixels in a specific *neighborhood*. Noise reduction is performed by shifting a filter of size MxN, which is usually a square matrix, in all vertical and horizontal possibility in an image, averaging the pixel contained in the neighborhood of the filter mask.

$$\frac{1}{273} \times \begin{bmatrix} 1 & 4 & 7 & 4 & 1 \\ 4 & 16 & 26 & 16 & 4 \\ 7 & 26 & 41 & 26 & 7 \\ 4 & 16 & 26 & 16 & 4 \\ 1 & 4 & 7 & 4 & 1 \end{bmatrix}$$

Figure 4.1: Example of fixed size convolution kernel approximating a Gaussian function with standard deviation of 1.

The filter used is an approximation of a *Gaussian function*, as explained in [30], which uses a two dimensional function in order to extract the Gaussian distribution values:

$$G(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{4.1}$$

The kernel size depends on the standard deviation a of the function, for example in figure 4.1, a standard deviation of $\sigma = 1$ was assumed. The values have been approximated since they are not really integers.

By smoothing the image using Gaussian filter noise is reduced and the image becomes blurred, so its details are suppressed. Moreover, if pixel intensities are considered in a specific method, the result might be altered since all pixel values will be mediated with respect to the filter.

The main idea is the following: if we consider light as a point-like structure, it will illuminate the scene in a cone-like fashion. When the light illuminates the sphere, this cone will be sliced by a non-planar object(the sphere). There has to be a specific region planar region which is orthogonal to the cone vertex. By forcing the normal of the light to be passing through the center of the sphere, an approximation of the light direction sis retrieved.

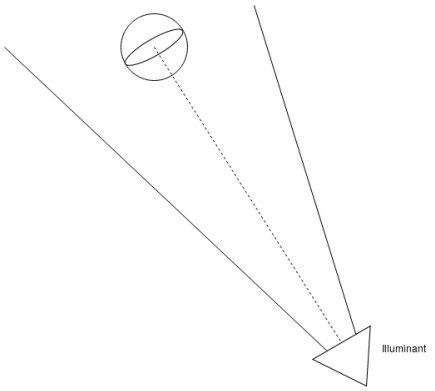


Figure 4.2: Simplification of how the illuminant light the sphere and the reflection of the light.

In figure 4.2, there will be a specific point on the sphere where the light vector will be perpendicular to the actual sphere surface. The region which describes where the light is is of course the one with the highest pixel intensity. However, using intensity pixel alone to find this region is not plausible, since a big neighborhood of that pixel will be "burned". The objective is to find the approximated plane which slices of the sphere and is perpendicular to the illuminant direction. By doing this, the illuminant direction will have the same direction as the plane's normal.

However, the planes which slices the sphere must be retrieved in some way, using some indications. The approach selected in this works the following: by using a selected level of pixel intensity, called **isovalue**, a **isocurve** is extracted from the image sphere. The isocurve extracted will represent a set of points which represents the intersection of a specific plane with the sphere. If the correct plane is selected with respect to the level curve, the light direction is retrieved.

The method used to retrieve the contours at a specific intensity uses an approach derived from the most famous *Marching Cubes Algorithm* for 3D meshing: the **Marching Squares Algorithm**.

4.1 Marching Square Algorithm

Marching squares is a computer vision algorithm, explaned in [31], that extracts a contour(isoline) with respect to a specific intensity values called isovalue. It can be divided into four steps:

- Divide the starting set of image points into 4 pixel cells.
- For each cell, an index of it is calculated.
- The geometry of the cell is retrieved using a lookup table.
- The contour position is refined using *linear interpolation*.

The first step implies a iterative scanning of the image, where 2x2 blocks of pixel are examined at a time. Each block of pixel is then simplified in a binary structure, where at each position a binary value is placed, if the pixel value is greater than the isovalue, 1 is placed, 0 otherwise.

4.1.1 Index calculation

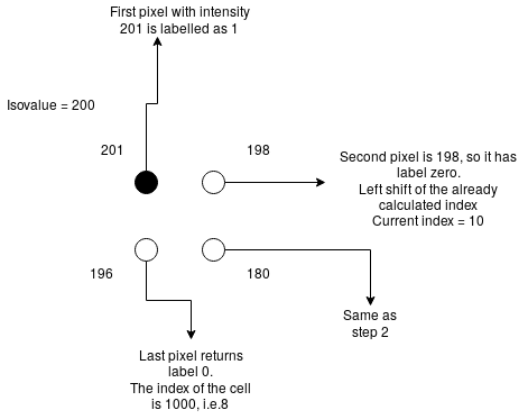


Figure 4.3: Example of the calculation of an index on a cell.

After each cell is built, an index is calculated by scanning each pixel belonging to a block in a clockwise direction. The index is calculated using *bitwise OR* operation as well as a left-shift, with the most significant bit being the top-left conrner of the block. The least significant bit on the other hand is the last one following the clockwise direction, that is the bottom left one. Since this process examine all pixels in an image, the complexity of this procedure depends on the dimension of the target. It should only be used for small regions and images.

4.1.2 Look-up table

After the index has been calculated for each cell, it is possible to outline the approximated shape of the contour in that particular block. Using the index, a lookup table is inspected, which contains the geometry of cells with a particular index. The

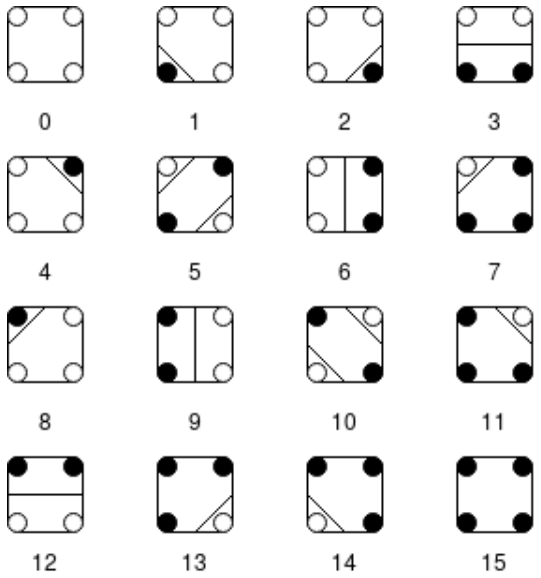


Figure 4.4: Lookup table utilized in Marching Squares algorithm.

geometry extracted from the table is only an idea of the countour, since the real position of the contour must take into consideration the real value of the pixel.

The fifteen entries of the lookup table are shown in figure 4.4, however not all cases are reliable. In case 5 and 10 in particular it is notable that the real geometry of the cell might differ, because those cases happens at *saddle points* and defines ambiguosly the contour. That is because the geometry of the center of each cell is not considered, so it is impossible to know, given two opposite vertices of the cell, if represents two different contours or there is a "bridge" between the two areas.

Here the image which is wanted to be scanned for contours is free of saddle points, so the implementation did not need to care about these cases, however the ambiguity can be easily solved by using an average of each corner value as a center cell value and choose different interpolated points.

4.1.3 Linear interpolation

The contour given by the first two step is good but not too accurate, since it is simple the mid-point between the cell's vertices. Starting from the geometry of the cell, the pixel value of the points considered in the cell are considered and the *interpolant* is calculated.

An *interpolant* is a point estimated within a range of a discrete set of data points. In this set, the points considerated are the ones which build the side of the cell where the contour is defined. The interpolant is the true point along the side of the square where the contour passes, which is calculated by looking at the intensity values of the pixel identified by the vertices.

For each cell, depending on its geometry, different number of vertices must be taken into account:

- For the contours at the angles, only the three vertices at that side of the cell

must be considered. Referring to the figure 4.4, if a cell has index 1, the vertices considered are top-left, bottom-right and bottom-left.

- If the contour passes through the cell's center, all four vertices must be taken into consideration. If case 6 in figure 4.4 happens, the two interpolant has to be calculated for opposite sides of the square.

What is wanted here is to calculated the interpolant between two known points using *linear interpolation*. The interpolant in detail is non other than a point on the line which passes through the two points. In this case, the linear interpolant is the side of the square which connects two vertices. The formulaa of that line can be expressed as:

$$y = y_1 + (x - x_1)\frac{y_2 - y_1}{x_2 - x_1} \tag{4.2}$$

By using 4.2 and considering that the variable in this case can regard both the x and the y position, two formulas can be derived:

$$\begin{aligned} y_i &= y_0 + (y_1 - y_0) * \frac{V - I(x_0, y_0)}{I(x_1, y_1) - I(x_0, y_0)} \\ x_i &= x_0 + (x_1 - x_0) * \frac{V - I(x_0, y_0)}{I(x_1, y_1) - I(x_0, y_0)} \end{aligned} \tag{4.3}$$

where $I(x, y)$ is the intensity level of the image at pixel with coordinates (x, y) and V is the isovalue given in input to the algorithm.

The result at each cell is a couple of points where the isocurve must go through, by connecting all points extracted the isocurve is retrieved.

By applying this algorithm to the region of the image where the sphere surface is, a curve at the isovalue's level is retrieved.

Chapter 5

Light direction estimation

The isocurve extracted using Marching square represents an estimate the intersection of an "ideal" plane, which is perpendicular to the light vector, and the sphere. By using that points as intersection, it is possible to find the plane and by retrieving his normal, a good estimate of the light direction can be calculated.

The plane in this work will be fitted following two different approaches:

- The reflection of the light by the sphere represented in the image by an arc; the contour can be used to fit a particular quadratic curve, an ellipse in this case. Since the ellipse represents the cut given to the sphere by the plane, what is wanted is to retrieve the normal of the plane on which the ellipse resides. This can be done by estimating the feasible orientation of the camera that could have captured the plane containing the circle from which the ellipse is generated, method defined in [32]

- Since the position of the sphere is known in space and the radius can be calculated, it is possible to back-project points from the image plane to the world coordinate frame.However, those points are none other than rays in $\mathbb{P}^{\not\parallel\not\in}$ projective space. The intersection rays-sphere must be calculated in order to retrieve the 3D coordinates of the contour's points. This is possible since the sphere position is known and it radius can be estimated using the image. Then, the best fitting plane is found, using as set the newly estimated contour points.

Here both approaches have been implemented and confronted. Moreover, some mediation techniqus are inspected, in order to see if they actually improves the precision of the normal estimated.

5.1 Fit a plane in space

In order to estimate the position of a specific point imaged in the plane in the world reference frame, triangulation is a possibility. However, it is only applicable when there is certainty that the point in both images is the same point imaged, otherwise the coordinates will be wrongly estimated.

Since the camera can be assumed to work as a pinhole camera model, which has known projective properties and mechanisms, the points' coordinates can be

estimated by exploiting the known rules of the imaging of a scene to an image plane by a pinhole camera model. However, this would require to know at least to know the objects in the scene and some coordinates, since an point in the image could have been derived from infinite points in the world reference frame.

In the first approach the exploitation of the projection mechanism of the pinhole camera model is performed, in order to place the isocurve points onto the sphere in the 3D world reference system.

5.1.1 Back-projection of the level curve's points

Using homogeneous coordinates and the projective space $\mathbb{P}^3$, it is possible to express the projection of a 3D point onto the camera's image plane by first "translating" the point into the camera's own reference system:

$$p = \begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix} \tag{5.1}$$

and subsequently, by using the camera's intrisic parameters, that homogeneous point is mapped in the $\mathbb{P}^2$ projective space. The complete projection is indicated as:

$$p' = \begin{bmatrix} f & 0 & c_x \\ 0 & f & c_y \\ 0 & 0 & 1 \end{bmatrix} (R|T) \begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix} \tag{5.2}$$

Since all parameters are given in this case, the back-projection of the imaged point is possible, however since no depth information are given, there is a need to disambiguate the possible points which could have been imaged to the starting point. In order to do this, for this spacific case there is a need to model the sphere in the world reference frame, which is camera four's one, and use it as a "depth indicator".

But first, the points need to be back-projected. The back-projection using homogeneous coordinate is a linear operation, which can be divided into smaller sub-operations. By multiplying an image point, projected into projective space $\mathbb{P}^2$, by the inverse of K, the point is placed on a virtual image plane, which resides in front of the camera. Then, by using the projection process of the pinhole camera, the rays which intersects the virtual plane and the camera center can be calculated.

The projective center of the camera, since the extrinsic parameters are known, is derivable in world reference frame coordinates:

$$O = -RT \tag{5.3}$$

What it is wanted to do is basically inverting the projection process, which includes a step where the point is translated into the reference frame of the camera which is taking the picture. In order to complete the inversion, it is needed to multiplicate the point for the inverse of the RT matrix of the world coordinate frame, which corresponds to the camera's four extrinsics. This operation is usually done by using the *pseudo-inverse* of the projection matrix, but since the projective space is used here, the projection considering the homogenous coordinates becomes a linear transformation and so the operation can be decomposed in smaller steps.

Since camera four coincides with the world reference frame, if the image considered is one took with this camera, only the intrinsic inversion is sufficient, since the point will already be represented in the world reference frame. For the other cameras, however, the points must be converted in the world reference frame, which is done by considering the inverse of the RT matrix of each camera, opportunely padded since homogenoeus coordinates are used, which brings back the point from camera frame to world frame.

To sum it all up, by multiplicating each contour point by the inverse of K, those are place onto the virtual image plane, and by using the inverse RT that plane is actually positioned in the world, which is extremely near to the camera N projective center.

By using those points, it is possible to geometrically define the ray which "pierces" the virtual plane exactly at the contour point, which starts from the camera's projective center By considered the algebraic difference between the camera's center and the calculated point and subsequently normalizing it, the direction d of the line passing through the virtual plane which originates from the camera center can be parametrized as:

$$x = o + \lambda d \tag{5.4}$$

where λ is a variable; the greater λ is, the farthest along the line the point x will be. Since λ is unknown, the line is infinite and coincides to the equivalence class of points whose image coincide with the starting 2D point.

By repeating this process for all contour points will result in series of rays which, starting from the camera N's projective center which will intersectate with the sphere. However, the radius of the sphere is still unknown, since at this point only the radius of the circle projectied on the image and the center of the sphere is known. It is possible, since the camera is approximated to a pinhole camera model, to get an estimate of the radius of the sphere approximating the radius of the circle by the similar triangles relationship.

In figure 1.5, it can be seen that the radius can be scaled, since there is a relation between image radius and the real one. The rays form two similar triangles and since some sides are known, the others can be derived as well. In detail, if a point and is image projection are given:

$$p = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \quad p' = \begin{bmatrix} u \\ v \end{bmatrix} \tag{5.5}$$

it is possible to define a function $T : R^3 \rightarrow R^2$ transforming each 3D point in $(x * \frac{f}{z}, y * \frac{f}{z}$, where f is the **focal length** of the camera. However this function is not linear, given the division by z, but using projective space and homogeneous coordinates can overcome this issue, since the division by the last coordinate is applied only when reverting to inhomogeneous coordinates.

This proportion is valid even for distances too, so given r_i, which is the radius of the circle in the image and r_s, the sphere radius, it holds that:

$$\frac{r_i}{f} = \frac{r_s}{z} \tag{5.6}$$

This applies only when considering the world reference system, and by using an images captured from camera four the radius is retrieved.

Since the parameters of the sphere are known and the rays' equations are defined, now the calculation of the intersections is possible. The general equation for a sphere is:

$$(x - x_0)^2 + (y - y_0)^2 + (z - z_0)^2 = r^2 \tag{5.7}$$

which rearranged in vectorial form becomes:

$$||x - c||^2 = r^2 \tag{5.8}$$

where c is the center of the sphere, r its radius and x a point belonging to it. In order to find the intersection between a sphere and the rays calculated before, the x in the equation 5.8 is substituted with the equation of the ray, that is 5.4[33]. The sphere equation becomes:

$$||o + dl - c||^2 = r^2 \tag{5.9}$$

By developing the norm, a second grade equation is derived, with d as the unknown variable. With respect to the value of the *delta* of the second order equation:

$$\nabla = l * (o - c)^2 - (||o - c||^2 - r^2) : \tag{5.10}$$

three distinct cases are possible:

1. $\nabla < 0$ means that no solution exists, so the line does not intersect the sphere
2. $\nabla = 0$ implies that one unique solution exists, meaning that the line is *tangent* to the surface of the sphere
3. $\nabla > 0$ is the most frequent case. The line is *secant* to the sphere, so two points are returned.

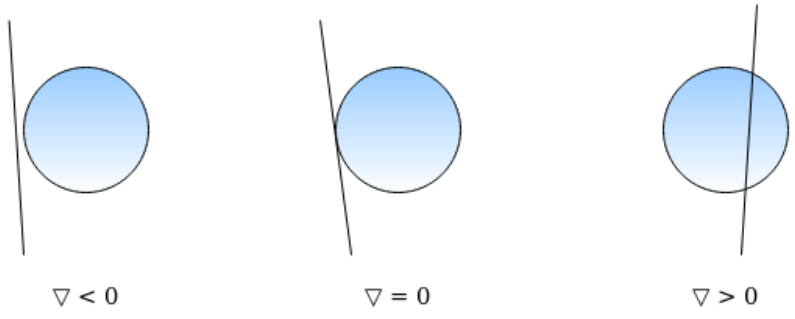


Figure 5.1: Visual examples on how a line can intersect a sphere

These actually represents the three situation where a line and a sphere can be, as represented in figure 5.1.

Since the contour point were all placed on the spheres and are reasonably distant from the circumference of the circle fitted, all rays will result secant to the sphere,

and so the equation will return two different results. There's a need to disambiguate the two solutions returned by the equation, since the only point of interest is the one which represents the starting point on the image. Since λ indicates how far the point lies on the ray and it is known that the sphere is located over the camera, the point in the image will be the one with the smaller λ, hence the first point on the ray which intersects the sphere.

By repeating this procedure for all the rays,a set of points in the world reference frame describing the geometry of the contour on the sphere is obtained.

From the contours projected, a plane is fitted using as set the intersections on the sphere. That plane's normal should be a good estimate for the light vector, since the plane represent a good reflection area.

It is possible to use a **least square minimization** in order to fit the best plane in a set of points. Using the centroid of the "cloud" of points considered, a matrix X of column vectors is created, whose elements are the difference between each point and the centroid. By using **Singular Value Decomposition**(SVD), the normal of the best fitting plane will be the left singular vector corresponding to the least singular value. However, since the image has a lot of noise, it is possible that some outliers have been considered as parts of the contours, and so the plane can be titled, as can be seen in figure 5.2. Since changing Marching squares in order to inspect the neighborhood for each pixel is very expensive and it does not guarantee to eliminate those outliers, an approach similar to optimization is used.

The distance from the centroid for each point is calculated and the farther away points which are over a certain threshold is discarded. The threshold considered is nearly equal to the mean of distance of all points to the centroid. By considering all positions for both small and big ball the position, the normals were projected in space and almost converged into a single points.

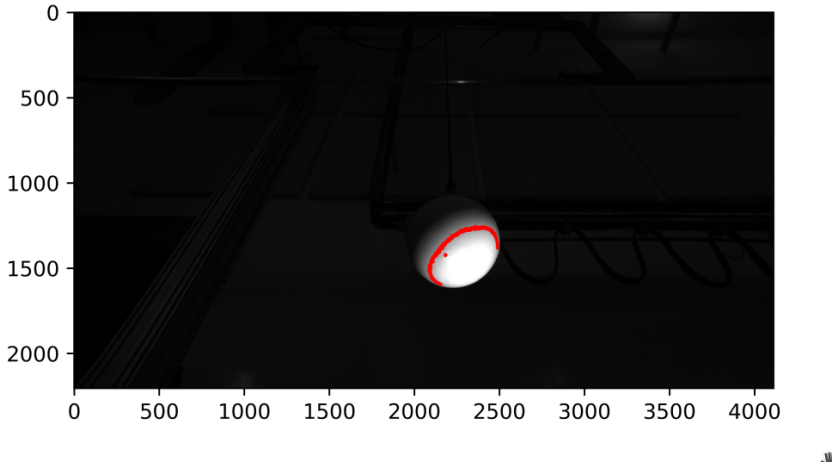


Figure 5.2: Big ball contour calculated with a noisy point, which is discarded after the first plane fitted.

This problem happens when bigger surfaces are considered, since noise is more incisive. The normals considered almost converged to a decent position, however they remain skew since our images are not optimal.

5.2 Ellipse Fitting

Another approach that can be used to get a good estimate of the plane is to consider the ellipse generated by the reflection of the light in the sphere. By doing so, the normal of the plane on which the ellipse resides can be calculated using a specific method.

An ellipse can be fitted by considering the contour as an arc of the ellipse and then the ellipse is fitted. However the fitting is not easy as the circle case, since the distance point-curve is not easily formulated and maybe will result in a non-linear function. So an approach similar to Hough or RANSAC is not convenient.

A technique which exploits **least squares** has been explained in [34]. The general conic equation can be written as:

$$ax^2 + bxy + cy^2 + dx + ey + f = 0 \tag{5.11}$$

and can be rewritten in a simpler vectorial form

$$F(a, x) = a * x = 0 \tag{5.12}$$

where $a = [a\ b\ c\ d\ e\ f]^T$ and $x = [x^2\ xy\ y^2x\ y\ 1]$.

As stated in [35], the problem of fitting a conic given a set of points p_i, can be solved by minimizing in a least squares sense the algebraic distances:

$$\mathbb{D}_{\mathbb{A}}(a) = \sum_{i=1}^{N} F(x_i)^2 \tag{5.13}$$

In order for this to not consider trivial solutions and be specific for ellipses, two things must be taken into account:

- The vector a should have some contraints, for example must be different than zero.
- Given the coefficient vector a, the constraint which identify a general conic into an ellipse is that this condition is fulfilled:

$$b^2 - 4ac < 0 \tag{5.14}$$

However, by adding the inequality constrain indicated in 5.14 the problem becomes hard to solve. What the researchers did in [34] is to scale the parameters so that the discriminant conditions becomes an equality constraint, since this can be done without loss of generality:

$$b^2 - 4ac = 1 \tag{5.15}$$

By transforming the constraint the complexity of the problem decreases, and it can be transformed into a linear system using Lagrange multiplier and differentiation:

$$\begin{aligned} Sa &= \lambda Ca \\ a^T Ca &= 1 \end{aligned} \tag{5.16}$$

where C is the matrix that expresses the constraint.

5.2.1 Improved Fitgibbonz's method

The method proposed in [34] by Fitzgibbon is a good method, but it has some weaknesses some problems. First, the method is numerically unstable, since in some cases, it returns complex numbers, moreover in some cases can return wrong or null results.

Halir and Flusser in [36] succeded in stabilizing the solution which exploits the "special structure" of the *scatter matrix* S and the coefficient matrix C, which can be decomposed into smaller matrices in order to stabilize the solution.

The problem with the method proposed by [34] in detail was:

1. If the input points lies exactly on the ellipse, the eigenvalue will become less or equal than zero, but with [34] only the smallest positive one is considered. In [36] the condition is altered and checked for all eigenvectors, and the one which return a positive value can be proven to be unique and is actually the solution of the problem.

2. The eigenvectors computed in the Fitzgibbon's method can be infinite or complex number, so the algorithm becomes numerically unstable. By scattering the matrix, researchers in [36] stabilized the algorithm and eased the computation.

The points for which the ellipse must be fitted are grouped as a matrix, which is called *design matrix*:

$$D = \begin{bmatrix} x_1^2 & x_1y_1 & y_1^2 & x_1 & y_1 & 1 \\ & & \vdots & & & \\ x_n^2 & x_ny_n & y_n^2 & x_n & y_n & 1 \end{bmatrix} \tag{5.17}$$

In [36], this matrix is divided into two submatrices, the formaer containing the linear part of the starting matrix, labelled as D_1, and the latter, labelled as D_2, which contains the quadratic side of $\mathbb{D}_a$.

By representing D in such a way, the scatter matrix is also splittable in many submatrices which can be calculated separately, four in total. The coefficient matrix C imposes the constraint indicated at equation 5.14, and so its matrix representation will mostly contain zeroesm since it implies only three coefficient, so matrix C can be reduced. Summarizing the splitting and the reducing operations:

$$\begin{aligned} S &= \left(\frac{S_1|S_2}{S_2^T|S_3}\right) \\ C &= \left(\frac{C_1|\ 0}{0\ \ |\ 0}\right) \end{aligned} \tag{5.18}$$

where

$$C1 = \begin{bmatrix} 0 & 0 & 2 \\ 0 & -1 & 0 \\ 2 & 0 & 0 \end{bmatrix} \tag{5.19}$$

By performing this splits and considering submatrices, the numerical stability of the algorithm increases, diminishing the probability of resulting in a complex number solution. Now the equation 5.16 can be rewritten as:

$$\left(\frac{S_1|S_2}{S_2^T|S_3}\right) * \left(\frac{a_1}{a_2}\right) = \lambda\left(\frac{C_1|\ 0}{0\ \ |\ 0}\right) * \left(\frac{a_1}{a_2}\right) \tag{5.20}$$

which returns the following system:

$$S_1a_1 + S_2a_2 = \lambda C_1a_1S_2^Ta_1 + S_3a_2 = 0 \tag{5.21}$$

The system indicated in equation 5.16 becomes:

$$\begin{aligned} Ma_1 &= \lambda a_1 \\ a_1^TC_1a_1 &= 1 \\ a_2 &= -S_3^{-1}S_2^Ta_1 \\ a &= (a1^T|a2^T) \end{aligned} \tag{5.22}$$

where M is equal to:

$$M = C_1^{-1}(S_1 - S_2S_3^{-1}S_2^T) \tag{5.23}$$

The condition considered here is indicated in 5.22, where instead of looking for the least positive eigenvalue, the condition $a_1^TC_1a_1 = 1$ is considered. As stated in [36], it can be proven that the only eigenvector which respects the condition is the correct one.

5.2.2 Normal computation

The goal here is to find the normal of the plane in which the ellipse resides. Of course finding a plane in 3D coordinates using only the information on the image is not simple. A possible way of estimating the normal is by finding the possible camera orientation which transformed a circle into the ellipse fitted. This method is exploited in a calibration method which uses as calibration target some visible coplanar circle, which is explained in detail in [32].

Since all conics are invariant under any generic projective transformation, a circle in the world will be represented as an ellipse on the undistorted image, if the image was took using a pinhole camera. This case the cameras are approximated to the pinhole model, so the condition holds. The ellipse fitted can be represented as this 3x3 matrix:

$$\begin{bmatrix} a & b & d \\ b & c & f \\ d & f & g \end{bmatrix} \tag{5.24}$$

where any points on the ellipse in $\mathbb{E}$ satisfies $x^TEx = 0$.

As stated in [37], it is known that the ellipse originated from a circle, it is possible to retrieve the rotation which transform the ellipse in a circle. Normalizing the matrix of the intrinsic K with the focal length f, will translate the same ellipse in the normalized image plane, whose origin is the optical center and has unitary focal length:

$$E_n = K_n^T E K_n \tag{5.25}$$

As stated in [37], the new matrix representing the ellipse can be decomposing via SVD:

$$E_n = V \Lambda V^T \; with \; \Lambda = diag(\lambda_1, \lambda_2, \lambda_3) \tag{5.26}$$

By using V, it is possible to define the rotation which trasforms the ellipse into a circle. There can be four possible camera position which imaged the starting circle as the fitted ellipse. The plane normals will have four different possibilities and can be defined using two signs, namely s_1, s_2, a angle ρ wrt the plane normal:

$$R = V \begin{bmatrix} gcos\rho & s_1 g sin\rho & s_2 h \\ sin\rho & -s_1 cos\rho & 0 \\ s_1 s_2 h cos\rho & s_2 h sin\rho & -s_1 g \end{bmatrix} \tag{5.27}$$

where

$$\begin{aligned} g &= \sqrt{\frac{\lambda_2 - \lambda_3}{\lambda_1 - \lambda_3}} \\ h &= \sqrt{\frac{\lambda_1 - \lambda_2}{\lambda_1 - \lambda_3}} \end{aligned} \tag{5.28}$$

In order to simplify the calculation, the angle ρ is fixed without loss of generality, letting the matrix become:

$$R = V \begin{bmatrix} gc & 0\rho & s_2 h \\ sin\rho & -s_1 & 0 \\ s_1 s_2 h & s_2 0 & -s_1 g \end{bmatrix} \tag{5.29}$$

The last column vector of R defines the plane normal in the camera reference frame, hence the normal of the ellipse. Since there are four possibility, a disambiguation is needed. As stated in [32], the last value of R indicates to which camera's is that normal referred to; if the last value is less than zero, it means that the normal is from the cameras looking at the back side of the plane. However, this is relative to the specific camera's reference frame. Since in this work a common world reference frame is considered. The most coherent normals are the one which has the negative z coordinate if lights near camera two are considered, positive if the illuminants are placed below camera one. This is because of the position of the camera four with respect to the entire system. Without some heuristic, a disambiguation between the two normals is impossible and unlike the methods [32] and [37], only one circle is present, not two coplanar one. So, the correct normal was selected by projecting it into the world reference frame and choosing the most coherent one.

5.3 Light puntual position

After the estimation of the light direction, the sphere was moved into different positions, in order to inspect if the rays actually "converged" to a point. By estimating light for multiple position, a beam of rays will be considered. In order to have a starting point, all the light direction are projected from the sphere center as rays,

with the usual parametrization seen in equation 5.4. However, those rays will never intersect in reality, given noise and approximation errors, and they will be skewed in the 3D space. In this case, it is possible to retrieve an approximation of the closest point to the rays.

The position estimate will be a good approximation if the level curve selected is a good one for all sphere position considered, which returns the correct estimate of the light.

The closest point to a set of rays can be retrieve in a least square fashion, the problem can be seen as a minimization of the distance between a point and a ray. The closest point on a ray to a point P is actually the reprojection of X on the ray or the starting point itself. By looking at the projection , the distance can be expressed as a function:

$$\begin{aligned} D = ||X - (o + \lambda d)|| \ \lambda > 0 \\ D = ||X - o|| \ \lambda < 0 \end{aligned} \tag{5.30}$$

where λ is the projection of X on the ray, using the equation at 5.4. The least square minimization problem can be considered using this function as:

$$argmin_p \sum_{i=1}^{n} D_i(P)^2 \tag{5.31}$$

However, as can be seen from 5.30, the function is **piecewise** and there can be cases where the solution does not converge. In order to give the best solution, an initial guess is done by mediating the starting point of all the rays. A loss function $F_i(P)$ is considered, whose built such that $E_i(P)^2 = F_i(P)^T F_i(P)$, whose is defined as:

$$F(X) = \begin{cases} X - (o - \lambda d) & \text{if } \lambda > 0 \\ X - S & \text{otherwise} \end{cases} \tag{5.32}$$

By using the initial guess and the error function as input of the least square algorithm, the center of the illuminant guessed is return as a point in 3D space.

Chapter 6

Experimental results

In this chapter the results of the method implementation are presented. Some comparisons between different approaches for the same step are presented. First,vthe dataset of images is presented, on which the information on the light has to be extracted. Then, the result of each step is presented. Furthermore, a qualitative measure of the precision of the method is retrieved. A precise metric could not be retrieved, since the real position of the light in the real world was not available and could not be retrieved.

6.1 Starting dataset

Several images and acquisition of the sphere were taken, in order to have a reasonable dataset. Moreover, two different sized balls were used. The impact of the object's size for this method was wanted to be derived. The resolution of the all the cameras is the same. The performance of both small ball and big ball are confronted. The shade of the light will be more accentuated and precise in a big ball, since a bigger surface is used. However, noise in the surface of the sphere is more accentuated for the big ball case. The small ball was approximatively the radius of a ping pong ball, which has a 40 millimeters diameter, while the big ball has a 90 millimiters diameter.

The photos were taken in a closed room with the minimum illumination, in that way only one dominant light source is really influent and detectable, which is the one used by the system.

In total, nine different acquistion for the small ball were used, while seven for the big one.

An example of image is considered in 6.1, where the ball is clearly visible and the reflection of the ball creates a visible region on the ball. This image would not be considered optimal for the calculation of the normal, since most of the contour is occluded and positioned on the other side of the sphere.

6.2 Triangulation

Each image represented the ball in a specific position, however, since the acquiring of a picture considering twenty different lights happens in different times(some milliseconds away). The position of the ball has been considered different between

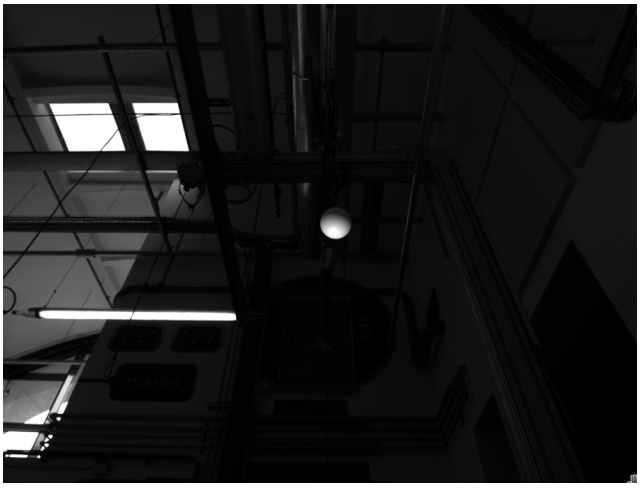


Figure 6.1: An image used for the estimation.

images considering different lights. This is because the ball was actually attached to a small wire which hanged over from the roof, so the ball was slightly moving during the acquisitions.

The triangulation was performed using a particular version of DLT, which requires the rotation matrix to be expressed as a rotation vector. This is possible using Rodrigues' formula, with can express every rotation using a vector and the a rotation angle θ. The angle θ represents the magnitude of the rotation vector.

Both the nine position of the small ball and the seven position for the big ball was triangulated for every light. Since the first four light are of different type, precisely a line of leds, and are not recognized by the light model considered, they will not be considered.

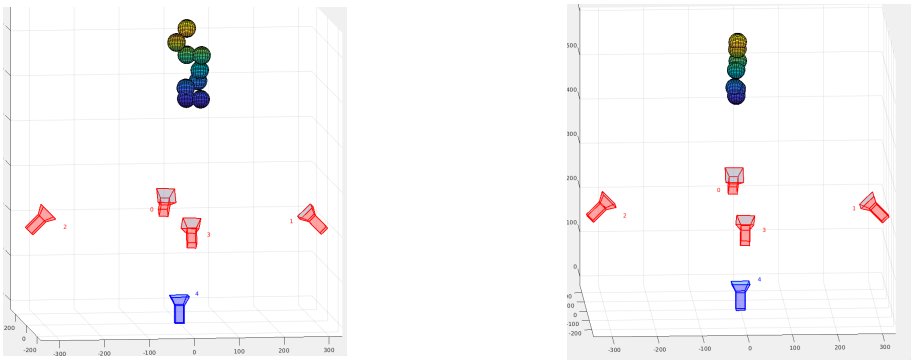


Figure 6.2: A visual representation of the position of, respectively, the small ball on the left and the big ball on the right. The ball on the right is kept small in order to be albe to recognize the sphere position in this image, in reality the radius of each sphere is 45, not 20

The triangulation returned good results, since the z coordinate of the triangulated was a pretty close approximation of the real distance in millimeters. The precision of the triangulation and the fitting algorithms are explored considering the reprojection

of the points calculated, comparing Hough gradient and RANSAC in terms of good sphere center's localization.

6.2.1 Ransac versus Hough gradient

The first comparison explored for these two methods was applicability. The images given contained more than one area with concentrated white pixels. The detection algorithm works only with intensity level, so many contours will be derived.

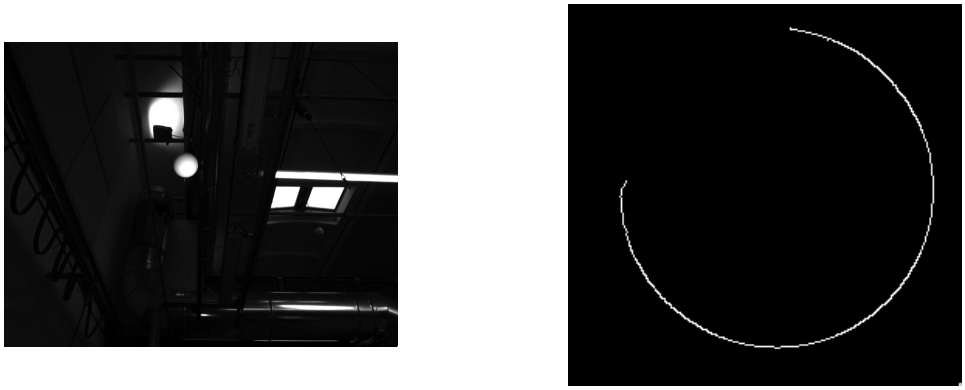


Figure 6.3: An example of the edges detected by Canny on the sphere picture

For the most problematic images, RANSAC was more robust and always returned a good enough estimate, but in return it was very slow on the calculations. On the other hand, Hough needed a maximum radius to work with, if a reasonable one was selected, the algorithm returned an estimate faster than RANSAC, otherwise the circles returned were completely wrong. After a few tries, since the radiuses of the image would change only by a little amount, a good maximum radius has been selected for both small and big sphere. As for the visual results, both of the methods gave a good estimate of the circle radius.

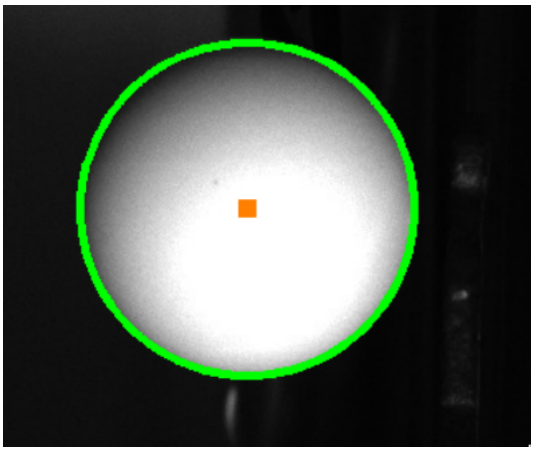


Figure 6.4: An example of the estimated circle

In order to test the precision of the estimates, the images when light with code

four was active was taken into consideration. As measurements, the reprojection error mean and standard deviation is considered.

From the visual reprojection on the image nothing can be inferenced since the distance between the point will be small, close to one or two pixels. In order to have a big enough dataset for comparing the method, all position of both small ball and big ball with respect to light four involved have been taken, the circle was fitted in every image for six different camera couples with both methods. Then, for each method the point was triangulated in space and reprojected.

	Mean	Standard deviation
Hough gradient small ball	1.651757	1.183154
RANSAC small ball	1.528608	1.135654
Hough gradient big ball	2.583920	2.138819
RANSAC big ball	2.230089	1.731760

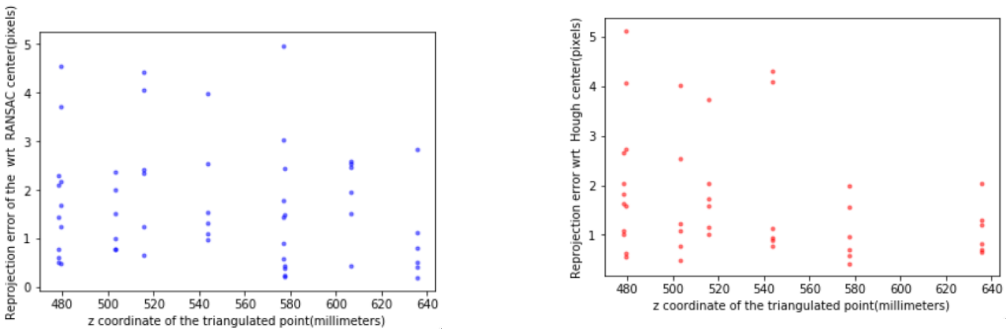


Figure 6.5: Scatterplot which expresses the reprojection error of both centers triangulated from circle fitted from RANSAC(right) and Hough(left) with respect to the z coordinate of the triangulated sphere center.This graphics was made using the nine position of the small ball when light 4 was active.

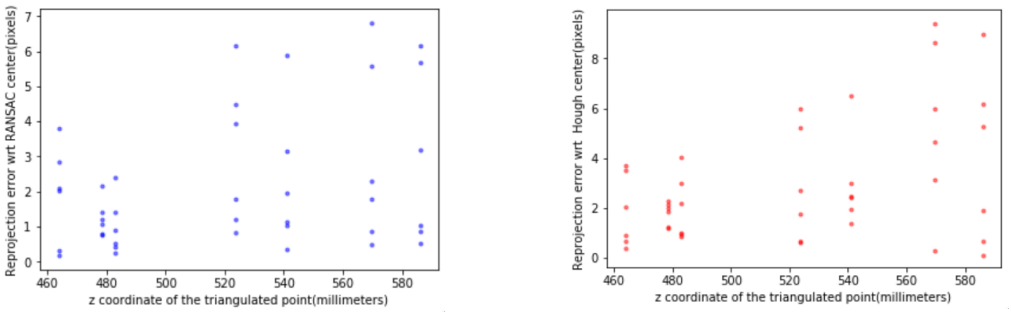


Figure 6.6: Scatterplot which expresses the reprojection error of both centers triangulated from circle fitted from RANSAC(right) and Hough(left) with respect to the z coordinate of the triangulated sphere center.This graphics was made using the seven position of the big ball when light 4 was active.

The results shows that RANSAC performs better both in terms of consistency and precision. This result was expected, since Hough discretize the space it's reason-

able that the precision would not be as good as RANSAC. However, Hough clearly outmatches RANSAC when it comes to speed.

The figures 6.5 and 6.6 represents the results of the test for all position, using the six camera couples which involves camera 0, camera 2, camera 3 and camera 4. The triangulated point is the same regardless of the order of the couple. It can be seen that the reprojection were more precise in the small case than in the big one, this is probably because the object is bigger and it is more difficult to spot the exact center of the sphere. Moreover, distance plays a big role on the triangulation of such a point, in fact as it can be seen from 6.6, the more the ball is farther away from the camera the worse the result will become. In the small ball case, however, the reverse is happening, as it can be seen from figure 6.5. This is because of the occluded area of the sphere, that is the part that is not illuminated by the flashlight, in cases of particular camera positions, might be estimated poorly and as a result the circle fitted would be slightly smaller than the "real one".

Figure 6.7: An example of why the big ball fitting is slightly more imprecise with respect to the small case. On both image, the ball is placed in its position farthest away from camera. While in the small ball the contour is clearly visible, in the big one it is obscured and might be estimated poorly

6.2.2 Application of Marching square on the images

Marching squares inspects all the pixels in the image space, calculating the contour using an isovalue and a 2x2 cell structure. This however becomes computationally tasking in our case,since the image space is huge and the point of interest here is only the image region where the sphere resided.

It is possible to extract a subimage by using the center coordinates and the radius returned by the fitting process. This is done by using the fitted center position and the radius of the circle. A cropping of the image is extracted, which extracts a square zone containing only the sphere. Now Marching squares can be applied effectively, since only the pixels of interest will be inspected, however the coordinates used in that region must be brought back on the original image, since the relationship between 2D image coordinates and the 3D world is valid only on the image space of the picture taken by the camera.

The algorithm performed quite good with different kinds of intensities and level curves, however it is very sensible to noise and might recognize some isolated points which does not stay on the isoline. The outliers are easily truncated using a small threshold on distance between the points, however the curve will not be precise, it

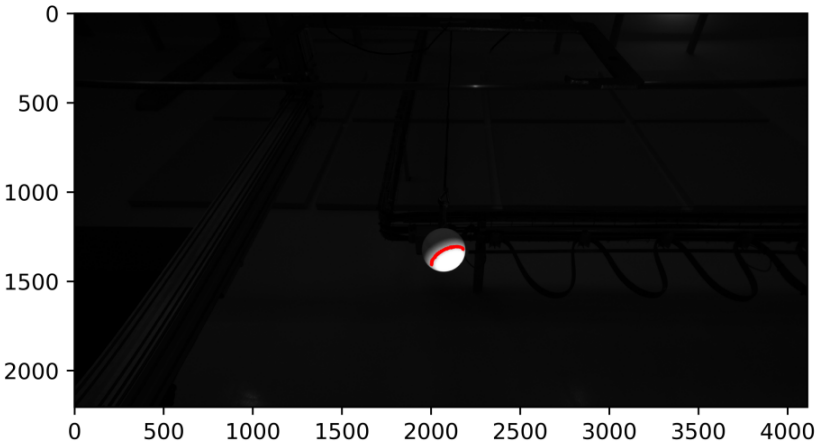


Figure 6.8: Result of the algorithm on the original image.

will be slightly jagged.

Moreover, since the original algorithm considers values greater than the isovalue as 1, the points at the extremity of the ball are considered part of the isocurve. In reality those points are "false positives", and presents them near the circumference of the "circle" delined by the sphere on the image. Considering the distance between the circumference and the actual points those outliers are eliminated from the considered contour.

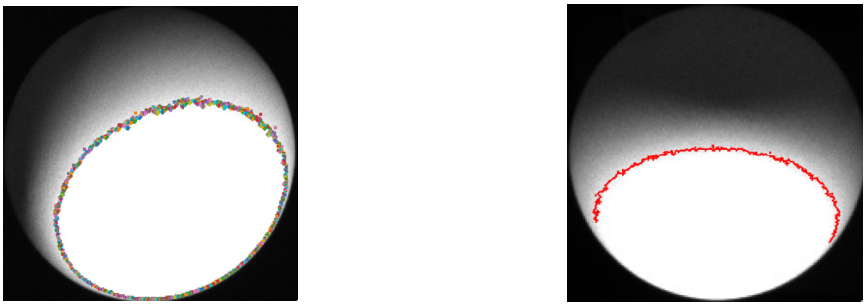


Figure 6.9: An example of the error on the calculation of the contour. Only the contour far from the circumference should be considered

6.2.3 Light direction estimation

The results of this phase will consider both ellipse and plane fitting methods. The two approaches are different, since the ellipse approach works only on the 2D image in order to extract the normal. The fitting on the plane happens in the 3D world reference frame. However, both approaches can be considered valid in a the proper isocurve is chosen.

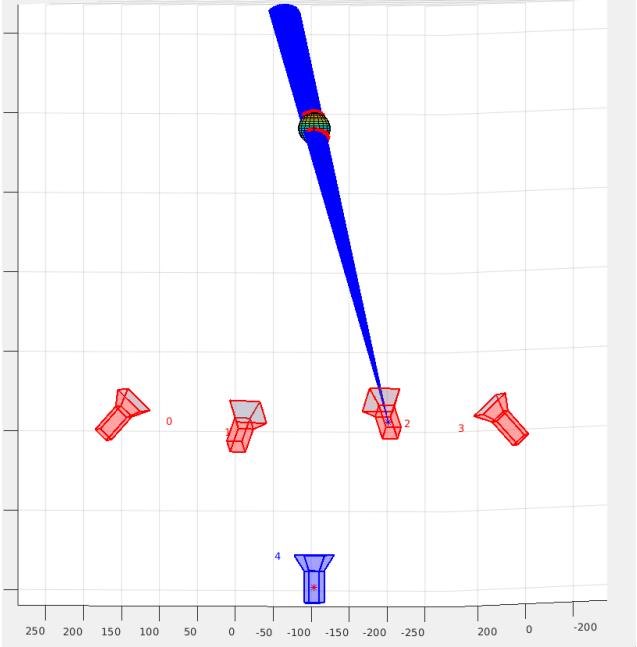
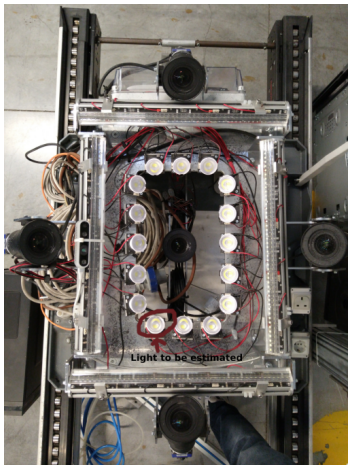


Figure 6.10: The visual representation of the intersection of the rays with the actual sphere.

6.2.4 Plane fitting

The results fitting the plane were quite promising, in fig 6.11 the normals of the nine small ball position are projected in the world, and the area in which they "concentrate" is actually a good approximation of the light position.



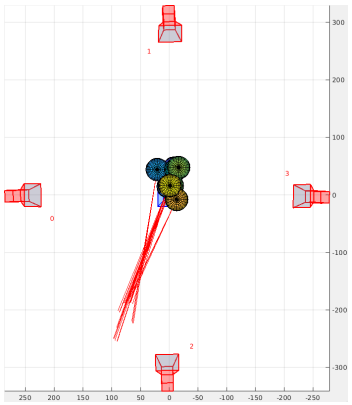


Figure 6.11: Comparison between the real position of light four with the estimated one

In order to improve the results, as said before, multiple normals using different level curves were estimated, and then mediated in order to have a more robust value, however, the result does not change that much.

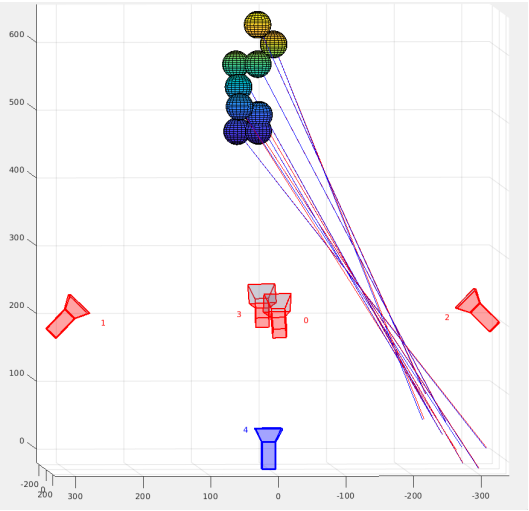


Figure 6.12: A visual comparison between single intensity normals(red) and mean multiple intensity normals(blue). The intensity curve considered where in a range $(x + 10, x - 10)$, where x is the actual intensity used for the single case.

However, as it can be seen from 6.12, there is not a big change. Of course, this would change if the actual variance of the isovalues would be increased, however it would likely result in a wrong estimate.

6.2.5 Ellipse fitting

Applying the improved Fitzgibbon's method on the contour points led to some interesting result. The fitted ellipse was a very good approximation with respect to the contour given.

The narrower the curve considered, the more tilted and "compressed" the ellipse will be. The threshold chosen for the points to be considered sufficiently far from the sphere is between five and fifteen pixel.

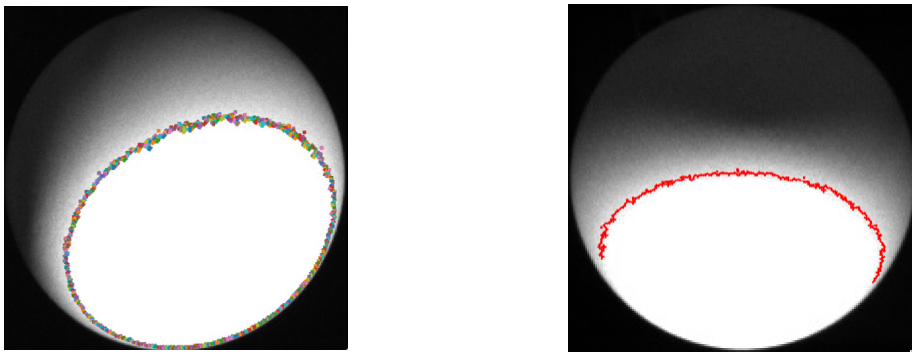


Figure 6.13: Comparison between a contour calculated without thresholding the points near the circumference. The rest of the real level curve is in fact occluded since it's on the back on the sphere. On the right, the points near the circumference are not considered.

The parameters of the ellipse, such as centers, angle of rotation and axis are derived from the coefficient vector a.

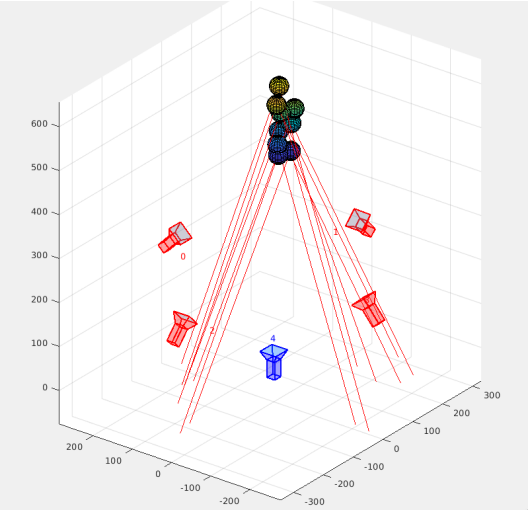


Figure 6.14: Projection of the normals calculated from the fitted ellipse, for all nine small ball position.

In figure 6.14 the two candidates normal for the light with code four are selected. As can be seen, the right-most normal is completely wrong, since light four is near to the camera with code two in reality, so it is discarded.

6.2.6 Comparisons between the two approaches

Both approaches returned some results regarding the light vector estimation. Of course, the ellipse method included some heuristics, especially for the normal computation and disambiguation. Moreover, it is not not clear how to choose the perfect level curve for a specific image, since a high value might not be optimal for noise implication, while a low value does not provide enough specificity and tilt to the plane.

While the level curve can be mediated in the plane case, in the ellipse case this approach is not so simply achieved, since the normal must be disambiguated manually.

However, by considering the two methods and their precision in equal conditions, the plane one is much more precise in estimating the light vector, since the projection of the directions are more convergent.

This is because the fitting of the plane is much more convenient in therms of the angle of the vector with respect to the center. Moreover, the ellipse approaches considers only information extracted from the image, and calculates the vector using intrinsic parameters. On the other hand, the plane method exploits the knowledge on the scene and the sphere itself in space, which should lead to better results.

Of course, while the two methods are both valid, the ellipse one was discarded, since the imprecision of the method with respect to the back-projection approach is clearly visible in figure 6.15.

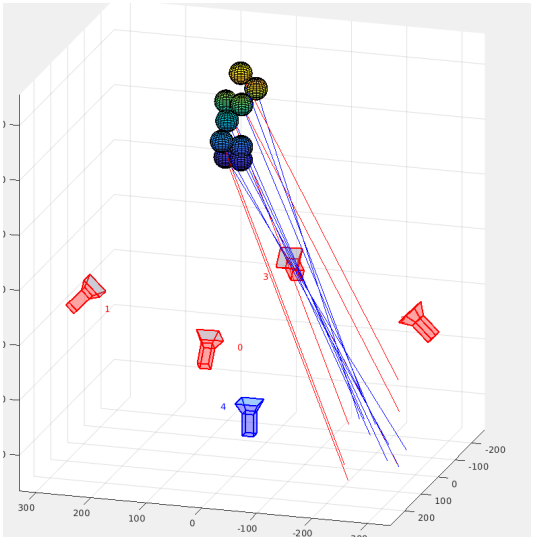


Figure 6.15: Visual comparison between the normals estimated with the ellipse method with red colors, and the plane fitting results in blue, using the same intensity value and considering the same curve

6.3 Light position estimation

The position is referred as the closest point from all the rays considered. The rays are the normals of the plane fitted with respect to the isocurve of a specific level curve considered. Nine position for the small ball are considered and seven for the big one, and so nine or seven rays are considered. Also, the possibility of using multiple curves has been inspected, in order to see if it actually improves the robustness of the estimate or if it makes it worse.

The source considered as reference for the as the previous steps, the light with code four, highlighted in figure 6.11.

The least square optimization were performed initially on the rays with only a single isocurve, extracted with the isovalue considered optimal for that particular image. The result of the algorithm can be seen in 6.16, which was estimated considering the normals visualized in figure 6.11. It can be seen that the position estimated is reasonable and coherent with respect to its real position in 6.11.

6.3.1 Method qualitative tests

As a qualitative measure for the method, the convergence of the rays is considered. However, the position of the must be taken into consideration also.

Since the real position of the light in the reference frame is not available and could not be retrieved, a different approach for testing the precision of the method was used.

Two experiments were performed, the first one was performed using a range with a greater variance between its values(a difference of fifty between maximum and minimum isovalue), the second one with a smaller variance.

The experiments returned pretty interesting results. The small ball estimates

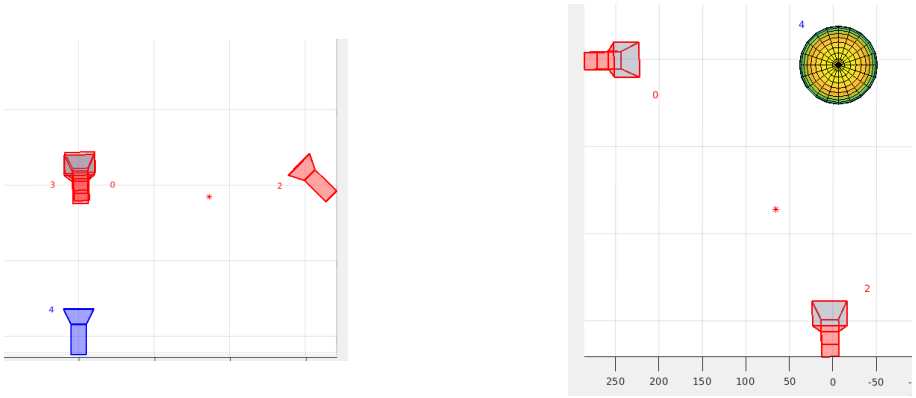


Figure 6.16: Multiple view of the estimated position of illuminant with code four.

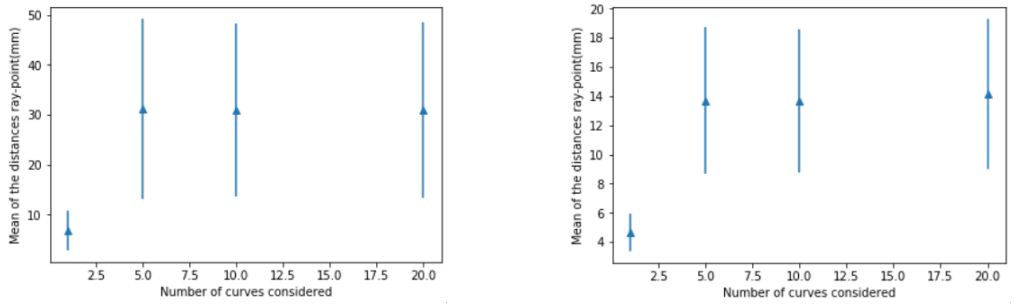


Figure 6.17: Error plot of the mean of distances between rays and the estimated light source point for both the small ball(left) and the big ball(right) using a big variance between the intensity values(difference of 5)

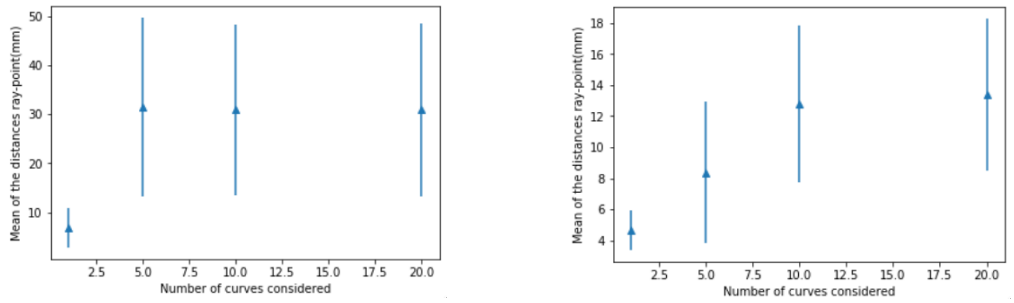


Figure 6.18: Error plot on the mean of distances between rays and the estimated light source point for both the small ball(left) and the big ball(right) using a small variance between the intensity values(difference of 1)

were negatively affected, the rays convergence was moved from a good position to a bad one, near the sphere. The general convergence of the rays, as can be seen from figure 6.17 and 6.18, deteriorates with the use of multiple level curves. However, for the big ball, which has a greater convergence but in the wrong position, greatly improved the estimated position for the light(6.19). The most interesting fact about this experiment is that the variance of the range in which level curve are selected played a little role regarding the convergence of the rays, but greatly impacted the

position estimated for the light. This is because each "stream" of ray pointed to almost the same position since the variance between the two range used was small.

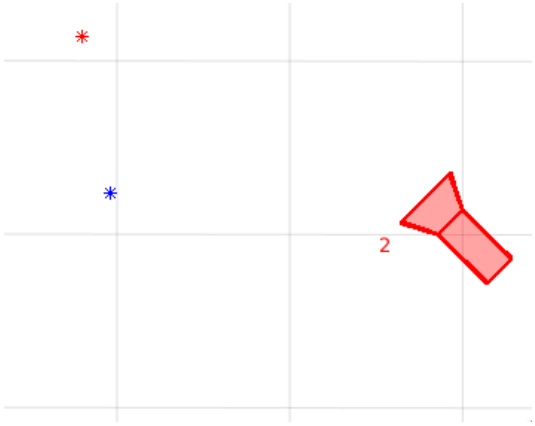


Figure 6.19: A visual representation of the improvement in the estimated position for the light four, using the big ball with one curve(red) and ten curves(blue)

Another thing to keep in mind is since the light is closer to the ball the actual area in which the light vector and surface normal have the same direction is much more difficult to pinpoint. Moreover, the normal of the plane fitted is projected from the sphere center in space, and the big ball normals are prone to be more negatively affected by this approach. Moreover, the center of the bigger sphere is most subject to error in the triangulation phase, so it might slightly move the sphere position, and might impact the projection of the contour and the plane fitted. Furthermore, the surface of the big ball it's more subject to noise than the small one.

From the experiments results, the small ball resulted more precise in the triangulation of the real position of the light while the big ball lead to a slightly better convergence of the ray. Considering our dataset and problem at hand, the small ball was preferred and returned better results.

Chapter 7

Conclusion and Future Work

In this thesis a novel approach was presented for light source position estimation, which works with point-like flashlight posed near the cameras, was presented. This was thought and implemented as an ad-hoc solution, starting from a pre-existent system in order to inspect the possibility of increasing the system's precision and power by adding the geometric properties of the illuminants.

The result were promising, all light positions were positioned at the right spot, however the light direction estimates did not converge in a precise point, which is reasonable given the considerable amount of noise on the images. Moreover, in order to really estimate the precision of the method, a real position of the lights in space should be measured, in order to be able to calculate the error that is made on the estimation.

Regarding the method itself, more in-depth analisys on the heuristics for extracting the level curve and projecting the normal can be done, considering maybe a different starting point for the light vectors. Furthermore, a heuristic study on each pixel of the sphere's surface on the image can be done in order to find the best possible isovalue which describes the optimal isocurve for a particular image. Those two points are particularly critical for the big ball case, since it has a greater convergence than the small one. If a good heuristic for the normal projection is researched for the big ball, this would lead to better results.

Another thing that can has to be done is optimize the light position using as starting point the position estimated. A non-linear optimization of the intial guess can be done in order to refine the position of such light and by consequence estimate its intensity.

Finally, after the refinement of the method, the light geometric properties should be added to the system's errors triangulation phase, in order to see if an actual improvement in the machine triangulation precision is reached.

Bibliography

[1] M. S. Langer and S. W. Zucker. What is a light source? In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 172–178, 1997.

[2] M. W. Powell, S. Sarkar, and D. Goldgof. A simple strategy for calibrating the geometry of light sources. *IEEE Transactions on Pattern Analysis and Machine Intelligence 23*, 2001.

[3] A.Iones, A.Krupkin, M. Sbert, , and S. Zhukov. Fast, realistic lighting for video games. *Computer Graphics and Applications, IEEE*, 23:54– 64, 06 2003.

[4] Q. Zheng and R. Chellappa. Estimation of illuminant direction, albedo and shape from shading. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1991.

[5] A. Pentland. Finding the illuminant direction. *Journal of the Optical Society of America*, 1982.

[6] C. Kambhamettu W. Zhou. Estimation of illuminant direction and intensity of multiple light sources. *ECCV. Lecture Notes in Computer Science*, 2353, 2002.

[7] S. Sarkar, D. Goldgof, and M. W. Powell. Calibration of light sources. In *Proceedings IEEE Conference on Computer Vision and Pattern Recognition. CVPR 2000*, volume 3, page 2263, Los Alamitos, CA, USA, jun 2000. IEEE Computer Society.

[8] W.Zhou. Scene illuminant estimation with binocular stereo matching. 2005.

[9] K. Koščević, M. Subašić, and S. Lončarić. Deep learning-based illumination estimation using light source classification. *IEEE Access*, 8:84239–84247, 2020.

[10] H. R. V. Joze, M. S. Drew, and P. A. T. Rey. The role of bright pixels in illumination estimation. *Color Imaging Conference*, 2012.

[11] W. Chojnacki, M. Brooks, and D. Gibbins. Revisiting pentland's estimator of light source direction. *Journal of The Optical Society of America A-optics Image Science and Vision - J OPT SOC AM A-OPT IMAGE SCI*, 11, 01 1994.

[12] M. Weber and R. Cipolla. A practical method for estimation of point light-sources. 12 2003.

[13] P. Nillius and J. Eklundh. Automatic estimation of the projected light source direction. *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, 1:I–I, 2001.

[14] J. R. Peterson H. Mytum. The application of reflectance transformation imaging(rti) in historical archaeology. *Historical Archaeology volume 52*, 2018.

[15] P. Kàn and H.Kafumann. Deeplight: light source estimation for augmented reality using deep learning. *The Visual Computer*, (35):873–883, 2019.

[16] G. D. Finlayson, S. D. Hordley, and I. Tastl. Gamut constrained illuminant estimation. *The Visual Computer*, (5):5–35, 1990.

[17] D.A. Forsyth. A novel algorithm for color constancy. *International Journal of Computer Vision*, (5):5–35, 1990.

[18] K. K. Wong, D. Schnieders, and S. Li. Recovering light directions and camera poses from a single sphere, 2009.

[19] F. Remondino and C. Fraser. Digital camera calibration methods. *ISPRS Commission V Symposium 'Image Engineering and Vision Metrology'*, 2011.

[20] Z. Zhang. A flexible new technique for camera calibration. *IEE Transactions on Pattern Analysis and Machine Intelligence*, 22:1330–1334, 2000.

[21] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision.* Cambridge University Press, 2003.

[22] Wikipedia contributors. Direct linear transformation, 2007. Accessed June 2020, last revision 15 May 2020.

[23] J. Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8(6):679–698, 1986.

[24] M. A. Fischler and R.C. Bolles. Random sample consensus: A paradigm for model fitting with application to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[25] A. Torsello, E. Rodola, and A. Albarelli. Multiview registration via graph diffusion of dual quaternions. *IEEE Conf. on Computer Vision and Pattern Recognition*, 2011.

[26] Xiao-Shan Gao, Xiao-Rong Hou, Jianliang Tang, and Hang-Fei Cheng. Complete solution classification for the perspective-three-point problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(8):930–943, 2003.

[27] V. Lepetit, F.Moreno-Nogoer, and P.Fua. Epnp: An accurate o(n) solution to the pnp problem. *International Journal of Computer Vision*, 81(155), 2009.

[28] A. Kaehler G. Bradski. *Learning OpenCV: Computer Vision with the OpenCV Library.* O'Reilly Media Inc., Gracenstein Highway North, Sebastopol, 2008.

[29] R. Dosselman, X. D. Yang, and I. Tastl. Determining light direction in spheres using average gradient. *Technical Report TR-CS 2009-1*, 2009.

[30] E. S. Gedraite and M. Hadad. Investigation on the effect of a gaussian blur in image filtering and segmentation. In *Proceedings ELMAR-2011*, pages 393–396, 2011.

[31] F. Hanisch. Marching Square. In , editor, *CGEMS - Computer Graphics Educational Materials*. The Eurographics Association, 2004.

[32] F. Bergamasco, L. Cosmo, A. Albarelli, and A. Torsello. Camera calibration from coplanar circles. In , editor, *2014 22nd International Conference on Pattern Recognition*, pages 2137–2142, 2014.

[33] Wikipedia contributors. Line-sphere intersection, 2006. Accessed June 2020, last revision 6 June 2020.

[34] A. W. Fitzgibbon, M. Pilu, and R. B. Fisher. Direct least squares fitting of ellipses. In , editor, *Proceedings of 13th International Conference on Pattern Recognition*, volume 1, pages 253–257, 1996.

[35] R. Haralick and L. Shapiro. *Computer and Robot Vision*. Addison-Wesley, 1992.

[36] R.Halir and J.Flusser. Numerically stable direct least squares fitting of ellipses, 1998.

[37] T. Wada Q. Chen, H. Wu. Camera calibration with two arbitrary coplanar cirles. *Proc. European Conf. Computer Vision*, pages 521–532, 2004.