FEDERAL UNIVERSITY OF PARANÁ

JOSÉ FRANCISCO BIANCHI FILHO

IDENTIFICATION AND THREE-DIMENSIONAL POSITIONING OF URBAN ENERGY LINES FROM OPTICAL IMAGES TO AID A TELEOPERATED PRUNING ROBOT

CURITIBA

2016

JOSÉ FRANCISCO BIANCHI FILHO

IDENTIFICATION AND THREE-DIMENSIONAL POSITIONING OF URBAN ENERGY LINES FROM OPTICAL IMAGES TO AID A TELEOPERATED PRUNING ROBOT

Dissertation presented as a partial fulfilment of the requirements for the Master of Science degree in Electrical Engineering from the Postgraduate Program in Electrical Engineering, Technology Sector, Federal University of Paraná

Advisor: Prof. Dr. Leandro dos Santos Coelho

CURITIBA

2016

Bianchi Filho, José Francisco
Identification and three-dimensional positioning of urban energy lines from optical images aid a teleoperated pruning robot / José Francisco Bianchi Filho. – Curitiba, 2016.
144 f. : il.; tabs.

Dissertação (mestrado) – Universidade Federal do Paraná, Setor de Tecnologia, Programa de Pós-Graduação em Engenharia Elétrica.
Orientador: Leandro dos Santos Coelho
Bibliografia: p. 138-144

1. Robótica. 2. Processamentos de imagens. 3. Reconhecimento de objetos. I. Coelho, Leandro dos Santos. II. Título.

CDD 624

TERMO DE APROVAÇÃO

JOSÉ FRANCISCO BIANCHI FILHO

IDENTIFICATION AND THREE-DIMENSIONAL POSITIONING OF URBAN ENERGY LINES FROM OPTICAL IMAGES TO AID A TELEOPERATED PRUNING ROBOT

Dissertação apresentada como requisito parcial para obtenção do grau de Mestre no Programa de Pós-Graduação em Engenharia Elétrica da Universidade Federal do Paraná.

Prof. Dr. Leandro dos Santos Coelho – Orientador
Universidade Federal do Paraná

Prof. Dr. Alessandro Zimmer – Convidado
Universidade Federal do Paraná

Prof. Dr. Alexandre Rasi Aoki – Convidado
Universidade Federal do Paraná

Prof. Dr. Roberto Zanetti Freire – Convidado
Pontifícia Universidade Católica do Paraná

Curitiba, 30 de agosto de 2016.

AGRADECIMENTOS

Ao meu pai José Francisco Bianchi e a minha mãe Gislaine Heurich Bianchi, sem os quais, eu nada seria.

A minha namorada Nayara Penteado Sanches, pelo amor, carinho e pela infindável paciência com minha ausência.

Ao meu irmão Giovanni Heurich Bianchi, pela alegria que transmite a todos ao seu redor.

Ao meu orientador Leandro dos Santos Coelho, por toda a ajuda e paciência desde os tempos de graduação.

Aos professores Alexandre Rasi Aoki e Alessandro Zimmer pelas valiosas recomendações dadas durante a qualificação.

Ao pesquisador, colega e amigo Luciano Cavalcante Siebert, que me ajudou diversas vezes ao longo dessa jornada.

Ao meu gerente, Eduardo Kazumi Yamakawa, por todos os valiosos conselhos dados desde que ingressei nos Institutos Lactec.

Ao meu amigo, Diogo Schwerz de Lucena que é, desde que o conheci, fonte de inspiração e boas conversas.

A toda equipe dos Institutos Lactec que me ajudou de maneira direta ou indireta com esse trabalho.

Aos Institutos Lactec, pela bolsa de mestrado e por todo material fornecido, sem os quais esse trabalho não seria possível.

Whether or not you can never become great at something, you can always become better at it

Neil deGrasse Tyson

ABSTRACT

Different factors may affect energy distribution quality, among them, one of the main causes is when vegetation gets into contact with overhead energy lines. Therefore, it is of main importance to prune vegetation close to energy lines. To improve this process it is possible to use a teleoperated robot, what allows the pruning activity to be accomplished in a remote and safe way. Cameras installed in the robot arm provide images from the pruning region to the operator even when direct sight is not an option. One of the main problems viewing the prunning region using a display is the lost of depth perception, what could make the operator unintentialy colide the robot with energy lines. Therefore, it would be of great aid a computer vision method capable of detecting energy lines and their three-dimensional (3D) positioning to aid the operator. During the state of the art review of energy line detection in images, it was perceived that, in general, the already proposed works operate in regions where the images present a clear background, not urbanized, and with the energy lines seen from above. Therefore, in this work, it is proposed a technique to detect energy lines and their 3D positioning in images taken in urban settings, factor yet unexplored in the recent literature. To reach this objective it is proposed the use of two visible spectrum cameras installed in parallel. In this way, regions with potential to be energy line are selected using edge detection followed by the geometric filtering designed using techniques inspired in graphs algorithms and curve fitting. After the regions with potential to be energy lines are found, their 3D position is obtained with stereo vision. To do so, the matching among points visible by both cameras is found and with triangulation, it is possible to recover the energy line 3D position. With the 3D information available, false positives are reduced by a factor of about seven and finally the energy lines are detected. A dataset containing stereo images of a scenario built with two power poles, three energy lines, and a tree between them was created in order to evaluate the presented method. In the commented dataset it was possible to reach accuracy of 98% at the end of the detection process, with 91% true positive rate. The causes of the false negatives cases are put in evidence in order to allow them to be overcame by future works. The algorithm proposed here outputs a colormap projected over the energy lines to inform the depth of each one in 2D and a point cloud to visualize each line in 3D.

Key words: Computer vision. Object Recognition. Stereo vision. Overhead Energy Lines. Pruning Robot

RESUMO

Diversos fatores podem impactar a qualidade da distribuição de energia elétrica, entre eles, um dos mais impactantes é o contato de vegetação com linhas aéreas energizadas. Assim sendo, é de suma importância a poda de vegetação próxima à linhas energizadas. Visando-se aprimorar esse processo, pode-se empregar um robô teleoperado de poda, de forma que a poda possa ser realizada de maneira remota e segura. As câmeras instaladas no braço robótico permitem que o operador tenha visão da área de corte mesmo quando a visada direta do solo estiver obstruída. Um dos problemas de se visualizar a região de corte por meio de um monitor é a perda de noção de profundidade, o que pode dificultar a operação. Dessa forma, seria relevante uma técnica de visão computacional capaz de detectar as linhas de energia e seu posicionamento tridimensional (3D) a fim de auxiliar o operador. Revisando-se a literatura, avaliou-se que, no geral, os trabalhos já propostos para detecção de linhas em imagens operam em situações com fundo limpo, não urbanizado e com vista superior das linhas de energia. Assim sendo, nesse trabalho é proposta uma técnica para detecção de linhas energizadas em imagens de regiões urbanas e a obtenção de seu posicionamento 3D, fator ainda não explorado na literatura recente. Para se alcançar esse objetivo é proposta a utilização de câmeras de espectro visível posicionadas em paralelo. Assim, regiões com potencial para serem linhas de energia são selecionadas utilizando-se detecção de bordas seguidas por filtragens geométricas aplicando-se técnicas inspiradas em algoritmos de grafos e ajuste de pontos selecionados a uma curva. Após a seleção de regiões candidatas a linha de energia, o posicionamento 3D é obtido utilizando-se de visão estéreo. Para tal, a correspondência entre pontos visíveis em ambas as câmeras é encontrada e com triangulação o posicionamento 3D da linha de energia é recuperado. Com a informação 3D disponível falsos candidatos são reduzidos por um fator de aproximadamente sete vezes e finalmente as linhas são detectadas. Para avaliação do método foi criada uma base de dados contendo imagens estéreo obtidas de um cenário montado com dois postes, três linhas de energia e uma árvore entre essas, na qual foi possível atingir níveis de precisão de 98% ao término do processo de detecção, contando-se com 91% de taxa de verdadeiro positivos. As causas dos falsos negativos são evidenciadas para que trabalhos futuros possam encontrar alternativas às dificuldades apresentadas. O algoritmo aqui proposto fornece como saída um mapa de cor sobre as linhas de energia para identificação da profundidade em 2D e uma nuvem de pontos para visualização em 3D.

Palavras-chave: Visão Computacional. Reconhecimento de objetos. Visão estéreo. Linhas de energia. Robô de poda.

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS AND ACRONYMS

2D	- Two Dimensions
3D	- Three Dimensions
ANEEL	- Brazilian Electricity Regulatory Agency
CC	- Connected Component
COELBA	- Electric Utility of the State of Bahia
fMRI	- Functional Magnetic Resonance Imaging
FN	- False Negative
FP	- False Positive
FPGA	- Field Programmable Gate Array
FPPEL	- False Positive Per Energy Line
FPPI	- False Positive Per Image
FPR	- False Positive Rate
GPU	- Graphic Processing Unit
HMI	- Human Machine Interface
KNN	- K-Nearest Neighbor
LoG	- Laplacian Of Gaussian
OpenCV	- Open Source Computer Vision
PCNN	- Pulse Coupled Neural Network
ROI	- Region Of Interest
RWT	- Real World Thickness
SAD	- Sum Of Absolute Differences
TN	- True Negative
TP	- True Positive
TPR	- True Positive Rate
UAV	- Unmanned Aerial Vehicle
WTA	- Winner-Takes-All

TABLE OF CONTENTS

1 INTRODUCTION

Telerobotics is a field of robotics that makes use of different automated devices with a human operator in control or, in other words, a human-in-the-loop. There are different control architectures within telerobotics, where the main differences are related to the autonomy of the devices (*i.e.* the degree of operator's interference) (SICILIANO; KHATIB, 2008). Hard to automate tasks due to the complexity or environments not suitable for humans are common examples of situations where telerobotics is useful (CORKE, 2011).

There are several examples of telerobotics applied by different knowledge areas (DUFF et al., 2010; PICCIGALLO et al., 2010; BURGNER et al., 2011; AZIZ, 2013). In 2012 (KAUFMAN, 2012), a rover, called Curiosity, developed as part of the NASA's Mars Science Laboratory Mission has gained positive media's attention while exploring the red planet. Surgery telerobots allow the execution of minimum invasive and high precision operations, an example is the Da Vinci System developed by Intuitive Surgical Inc. (GUTHART; SALISBURY, 2000). Concerning distribution and transmission lines maintenance, some effort was devoted by the Universidad Politécnica de Madrid (Spain), who developed a teleoperated robot focused on live-line maintenance shown in FIGURE 1 (ARACIL et al., 2002).

FIGURE 1 - Operator using the human machine interface (HMI) and the teleoperated robot respectively.
SOURCE: (ARACIL et al., 2002).

As noted, telerobotics has an extensive spectrum of applications. However, there is still room for research concerning energy lines maintenance due the numerous factors that may compromise the energy distribution. Among the possible causes, vegetation making contact with lines is one of the leading

causes (Simpson; Bossuyt,1996). In this way, pruning vegetation nearby energy lines is a task of main importance to increase energy supply reliability.

The pruning task near energized overhead power lines may be performed in different ways (Siebert et al., 2014), depending on the risks and urgency involved. Among them, when the branches are closer than a minimum distance from the overhead energized lines, the prune must be performed by a live-line team (Ministério do Trabalho e Emprego, 2004). The mentioned pruning modality makes use of an insulated bucket truck to bring the lineman close to the point where the pruning is required as seen in FIGURE 2.

FIGURE 2 – Electric Utility of the State of Bahia (COELBA) lineman in an aerial bucket carrying out the trimming task close to energy lines.
SOURCE: The author (2016).

As one can perceive, the pruning performed by the live line team presents intrinsic risks to the lineman for two main reasons: it is a procedure performed several meters above the ground and close to live lines. Due to the constant risk faced by these workers and the lack of available options concerning the urban pruning with telerobotics close to live lines, a new equipment is being developed by a research and development project regulated by the Brazilian Electricity Regulatory Agency (ANEEL, in portuguese, Agência Nacional de Energia Elétrica) and funded by the Electric Utility of the State of Bahia (COELBA, in portuguese, Companhia de Eletricidade do Estado da Bahia). The objective of the project (PD 0047-0062/2012) is to design and build a teleoperated robot capable of pruning vegetation near urban overhead energized lines in a safe manner.

To teleoperate a robotic arm in the field is not a trivial task. One of the difficulties of such task is the lack of sight of the trimming region caused by the

vegetation or other obstructions, as seen in FIGURE 3. This problem can be minimized with the aid of a camera system installed in the robot arm streaming images from the region of interest to a human machine interface (HMI) display.

FIGURE 3 - Occlusion caused by the tree when observing the pruning process from the ground.
SOURCE: The author (2016).

Even with the image streaming, there is a drawback if compared with direct sight of the scene. The projection of the three-dimensional world to the two-dimensional image causes loss of depth perception. Consequently, the operator becomes more susceptible to place the robot arm overly close to the energy lines unintentionally. Thus, an algorithm capable of identifying energy lines and its three-dimensional positioning to warn the operator of energy lines contact risk is promising. In such case, feedback to the system may be provided to slow its actuating speed or even block completely any movement.

1.1 OBJECTIVE

1.1.1 GENERAL OBJECTIVE

The main objective is to develop a computer vision algorithm capable of detecting overhead energy lines and its positions to aid the control of a teleoperated robotic arm able to perform pruning close to overhead energy lines.

1.1.2 SPECIFIC OBJECTIVES

To achieve the general objective it is necessary to:

- Establish databases of vegetation close to overhead energy lines;

- Evaluate different computer vision algorithms;
- Find the state of the art of overhead energy lines detection;
- Develop a methodology to detect power lines and measure its distance in relation with the camera installed in the robot arm;
- Test and validate the developed methodology.

1.1.3 CONTRIBUTION

The contribution of this work goes beyond the aid to a teleoperated pruning robot, as it can be used as well to aid unmanned air vehicles (UAVs) navigation and to aid power line maintenance robots, just to cite a few applications. Therefore, it is expected to contribute to areas correlated with distribution/transmission lines maintenance, resulting in an indirect improvement to the quality of the energy distribution or transmission service. Furthermore, it seems that it is the first time that depth information obtained using stereo correspondence in the images of two cameras is applied during the energy line detection in urban regions, what, hopefully, will open a door to new techniques using similar approaches.

1.2 WORK STRUCTURE

Along this chapter, the dissertation subject was introduced highlighting its scope, goals, and contributions. The remainder of this work is presented as follows.

Chapter 2 details the necessary theoretical background to understand the methodology, covering an energy line literature review, graph theory concepts, edge detection and finally stereo vision.

Chapter 3 covers the materials used, as software, cameras and datasets. Furthermore, the chapter details the developed methodology to detect energy lines.

The obtained results are presented in Chapter 4, separated in 2D and 3D results.

Finally, chapter 5 highlights the key findings of this work and provides recommendations for future research.

2 THEORETICAL BACKGROUND

This chapter summarizes the theoretical background necessary to understand the techniques applied along the methodology passing through the review of already proposed energy line detection algorithms, graph theory, edge detection and stereo vision.

2.1 OVERHEAD ENERGY LINE DETECTION

Power line maintenance and UAV navigation are the two main reasons behind research focused on overhead energy line detection. There are related works that perform detection of similar objects, as Chen et al. (2015) that present a method to detect a guiding rope to aid remotely operated vehicles to perform inspections underwater and Ortiz et al. (2009) that proposed a technique to detect undersea telecommunication cables.

As enlightened by Mirallès et al. (2014) state-of-the-art review, different kinds of sensors are used to detect power lines, visible cameras, infrared or millimeter wave cameras. Using electromagnetic sensors is a problem because it would fail to detect overhead wires during power outages and have its signal disturbed by vegetation near to it (MARTINS et al. 2014). Thermal images suffer from similar problems, during outages or low consumptions hours, the energy line may present a temperature similar to the environment, letting it hard to be distinguished (MIRALLÈS et al., 2014). This was confirmed during tests using a thermal camera, where the line appeared with high contrast against the sky (greater temperature difference) but disappeared in front of vegetation (smaller temperature difference) as shown in FIGURE 4.

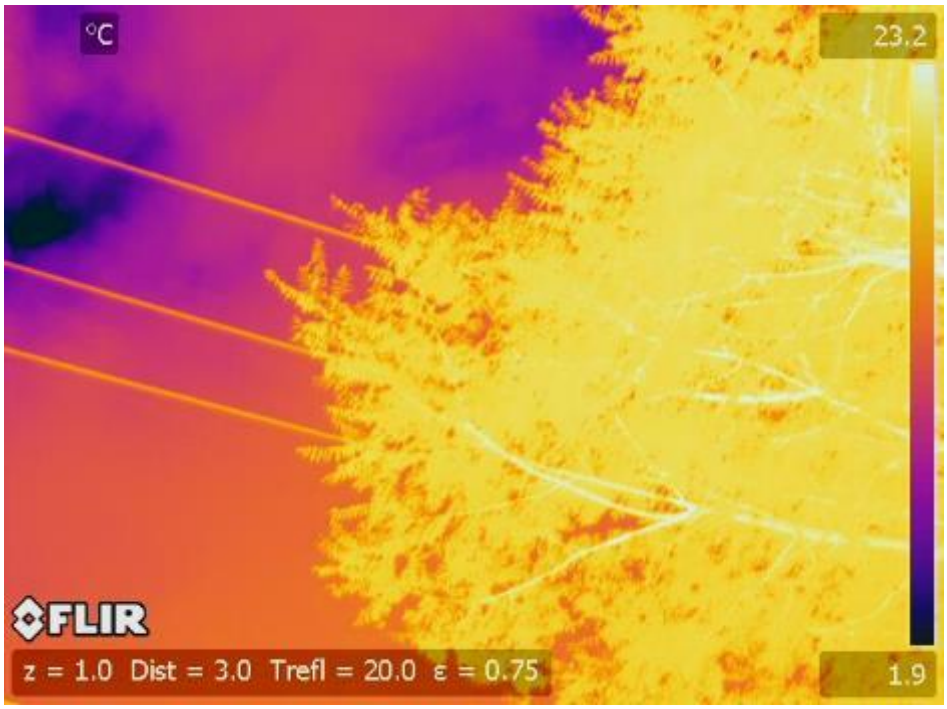


FIGURE 4 - Thermal image of power lines with vegetation and sky in background.
SOURCE: The author (2016).

In the following paragraphs, different methods to detect energy lines or similar objects are described; consider that all of them uses grayscale images and visible spectrum cameras, except when noted.

Candamo and Goldgof (2008) used a straight model of lines and defined a maximum thickness in pixels. Their approach start by applying Canny edge detector to obtain initial edges. Later connected components are eliminated based on its size and eccentricity. The error of fitting each connected component to a straight line is used to eliminate remaining false candidates. In a final step, lines without symmetric line profiles at both sides are eliminated. FIGURE 5 shows the line and the profiles positioned *h* pixels from it.

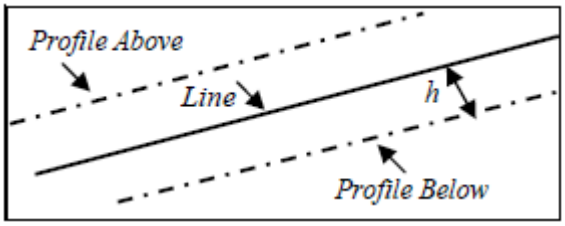


FIGURE 5 - Profile lines above and bellow a line.
Source: Candamo and Goldgof (2008).

Candamo et al. (2009) sought to detect lines in videos to aid aircraft in low altitude urban settings. The used technique starts with Canny detector to generate an edge map. Thresholding using size and eccentricity is then applied

to create a feature map. This map is complemented with information from a motion descriptor. In the last step, line detection is performed using a windowed Hough transform for straight line detection.

Using a filter based on a pulse coupled neural network (PCNN) model Li et al. (2009) generated an initial edge map. Candidates to Power lines are then found using a Hough transform for straight lines. Finally, K-means clustering groups lines with the same orientation. Considering that power lines occupies most of an image, the cluster with more votes is considered the one with the real power lines.

Yang et al. (2012) applied an adaptive thresholding scheme to obtain the initial binary map. Then, Hough transform was used to detect line candidates, while a fuzzy C-means clustering algorithm discriminate power lines from other candidates using direction and length features.

Zhang et al. (2012) proposed RGB to HIS color space conversion and used the I channel as the input image. Otsu threshold method was used to generate an initial binary image. Later, Hough transform for a straight line is applied, and the result is filtered using K-means. As the method is used in video sequences, a Kalman filter is applied over the Hough Space to keep track of power lines from the previous frame to the next frame.

Cao and Yang (2013) proposal is to detect power lines in aerial images, therefore assuming that power lines appear as straight lines in images. The proposed method starts by creating a derivative map and then applying Radon transform over it. As the derivative map of a power line presents an edge rising and another edge falling, two peaks with the same direction and similar magnitude are selected in the Radon matrix. Finally, with k-means clustering power lines with similar directions are detected. Lines that are not parallel with any other are removed, and lines close to each other are merged.

Song and Li (2014) took advantage from a matched filter response to adjust a thresholding of a first order derivative of Gaussian in order to obtain an edge map. To eliminate false candidates, connected components with fewer pixels than a threshold are eliminated. Noticing that any power line can be fitted by a quadratic polynomial and its second order derivative is a constant, a metric that evaluates this constancy is used together with an evaluation of the eccentricity of the connected component. Evaluations above a threshold

eliminate the related connected component. To conclude, a graph cut model is used to connect segments and form whole power lines.

To improve energy line detection rates, Luo et al. (2014) proposed the use of color and near-infrared images (RGB-NIR). The method starts by using a modified line segment detector (GROMPONE VON GIOI et al., 2010)) in red, green, blue, near infrared (RGB-NIR) images to extract initial segments. Quasi-Parallel line segments with a distance minor than a threshold are kept as power line segments candidates. These candidates are filtered again using NIR and color features. Finally, the remaining candidates are connected using straight line Hough transform.

Chen et al. (2016) developed a method to detect power lines automatically in high-resolution remote sensing images. First, the Curvelet transform is used to enhance the original image in order to improve the edge extraction. Then an improved Radon Transform is used to detect features from the edge detection image. Finally, the expected symmetry in both sides of a line seen from afar is used to filter out false candidates.

Methods used for maintenance in aquatic conditions uses slightly different approaches. Ortiz et al. (2009) proposed a method that performs segmentation, detection and validation to detect undersea communication cables followed by the use of Kalman filter to keep track of the cables. Chen et al. (2015) applied image enhancement using color space conversion, followed by a Canny edge detection and line detection using Hough transform for straight lines.

As one can perceive, all the commented works use a similar framework. In the first step, a technique is used to detect edges, this step is followed by some kind of filtering and in a final step candidates to power line are merged to form whole lines. Some techniques apply an extra phase to use temporal coherency, comparing frames from a video stream.

The techniques used to obtain the binary image vary from article to article, going from the simple edge thresholding, the use of Canny detector or even PCNN.

When the filtering phase is present, normally it is performed using some threshold related to a number of pixels in the segment, geometric relations, the eccentricity of an ellipse evolving the connected component of the segment or by using color thresholding.

In the final phase, most articles try to merge segments, using Hough transform, Radon transform, clustering techniques or graph algorithms.

Most of the algorithms test the developed techniques in own datasets, making it difficult to check the real efficiency of each one, with exception to the work of Candamo et al. (2009) that used an public available dataset. This dataset own a rich variety of power lines images with different graphic quality, going from images with high noise and blurred images to steady images, but none with high resolution. It even has a groundtruth, but this uses points to mark the starting point and ending point of each power line, which is inaccurate due to the catenary form assumed by a power line when affected by the gravity.

Mirallès et al. (2014) remembered that power lines normally appears as a very thin structure in an image. Therefore no texture characteristic is used. Even if this was not the case, and a close view form the power lines was available, it is necessary to remember that there are several types of power lines varying in texture due to a different type of outside isolation, diameter, material and even the manufacture process. Probably this is the reason because no article found proposed the use of texture.

The use of the Hough/Radon transform for straight lines is quite common among the articles. Most of them use images taken from afar while UAV is flying or using satellites. Taking images from the top of an energy line make it looks a straight line, but in fact, it is not. As already stated the real shape of a power line is a catenary, what would make some methods fail if the images were taken from a different angle.

Noticing that the majority of the works consider that the images are taken from afar and from the top, this lets a door open for a new algorithm that takes in consideration images taken from any angle. Another characteristic not yet explored is the use of multiple cameras, a feature that may show potential to aid during the detection process.

2.2 GRAPHS

Commonly new research fields start from the urge to solve a real world problem; it is believed that with graph theory this was not different (ALEXANDERSON, 2006). Leonhard Euler built the foundations of the area in

the earliest known paper of graph theory to solve a problem known today as the Königsberg Bridge Problem (BONDY; MURTY, 2008; RAO, 2006).

This problem involves the river Pregel, in the section that crosses the East Prussian city of Könisberg (now Kaliningrad), where lies the Kneiphof Island (ALEXANDERSON, 2006). A total of seven bridges connected the two sides of the river and the Island during Euler's time; FIGURE 6 shows the bridges and the city.

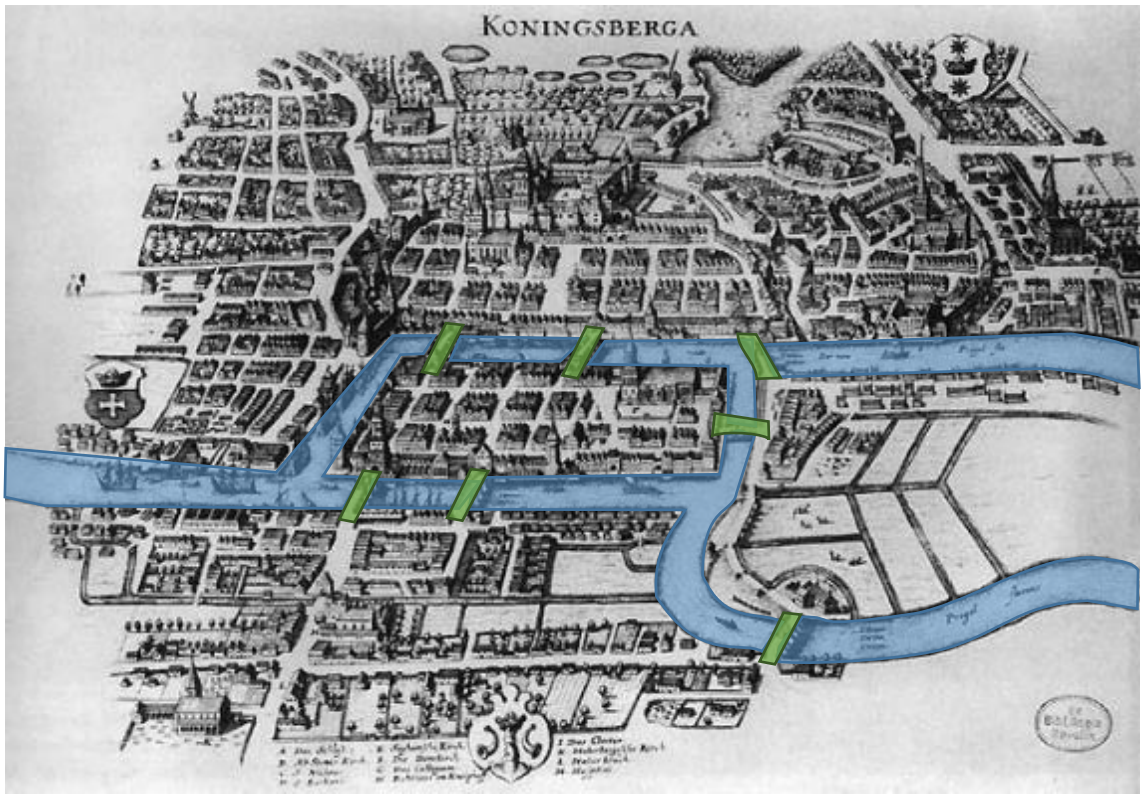


FIGURE 6 - Seventh century Königsberg with highlighted Pregel river in blue and the seven bridges in green.
SOURCE: Based on (Merian-Erben, 1652).

The Königsberg Bridge Problem inquires: could a person start a path passing through all the seven bridges and return to the initial point without crossing any bridge more than once? When Euler became aware of the problem, he perceived that the dimension of each bridge or even the shape of the island do not matter, in opposite with the connectivity. In fact, he proved that there is no solution to this problem and began the generalization of it to the case of two islands and four rivers, laying the foundations of graph theory. (Alexanderson, 2006; Chen, 1997)

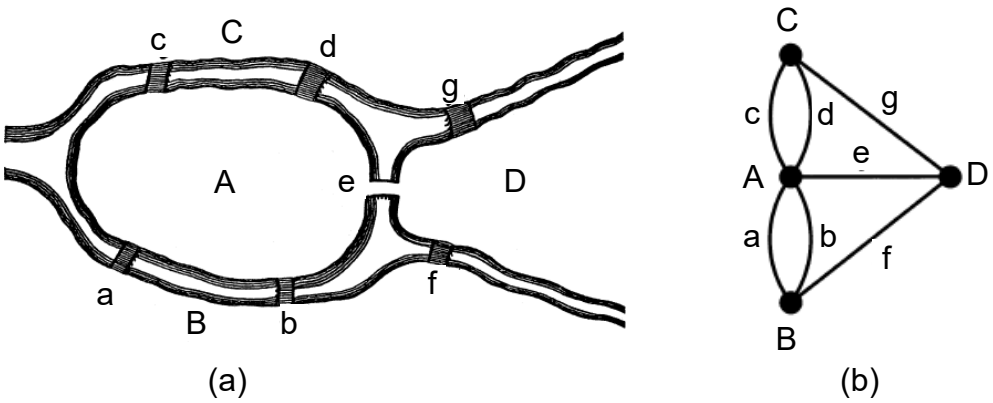


FIGURE 7 - (a) Diagram from Euler's paper (b) Representation using graphs.
SOURCE: Adapted from (a) (Euler, 1741) (b) (Hopkins; Wilson, 2004).

FIGURE 7(a) shows how Euler represented the Königsberg Bridge Problem during his time. However, the kind of representation shown in FIGURE 7(b) where the dimensions are not important are common nowadays, but it only appeared in the 19ths (HOPKINS; WILSON, 2004). In fact, many problems are modeled like diagrams of sets of dots and lines linking certain pairs of them (BONDY; MURTY, 2008). These diagrams may represent different sort of problems, e.g. social networks, where lines linking two dots represent a pair of friends; roads, where dots are cities and a line represents the road linking them; and so on. In short: the concept of Graph rises from the mathematical abstraction of situations of this type (BONDY; MURTY, 2008). In the following sub-sections, the graph mathematical representation and some concepts of graph theory are described and finally related with image processing.

2.2.1 MATHEMATICAL REPRESENTATION

A graph G is an ordered pair $G = (V(G),\ E(G))$, where $V(G)$ is a set and $E(G)$ is a subset from $\binom{V(G)}{2}$. Each element of $V(G)$ is called vertex and each element of $E(G)$ is called edge (BONDY; MURTY, 2008). To represent a graph each vertex is drawn as a point and each edge by as a curve joining two vertices (not necessarily distinct) called its endpoints (WEST, 2001)

Take as an example a graph where:

$$G = \begin{cases} V(G) = \{v_1,\ v_2,\ v_3,\ v_4,\ v_5\} \\ E(G) = \{\{v_1,\ v_2\}, \{v_2,\ v_4\}, \{v_4,\ v_5\}, \{v_5,\ v_5\}\} \end{cases}$$

The graphical representation of G is not unique, in fact it may be represented in different ways as long as the connections among the vertices are respected. FIGURE 8 shows two valid representations of the same graph G.

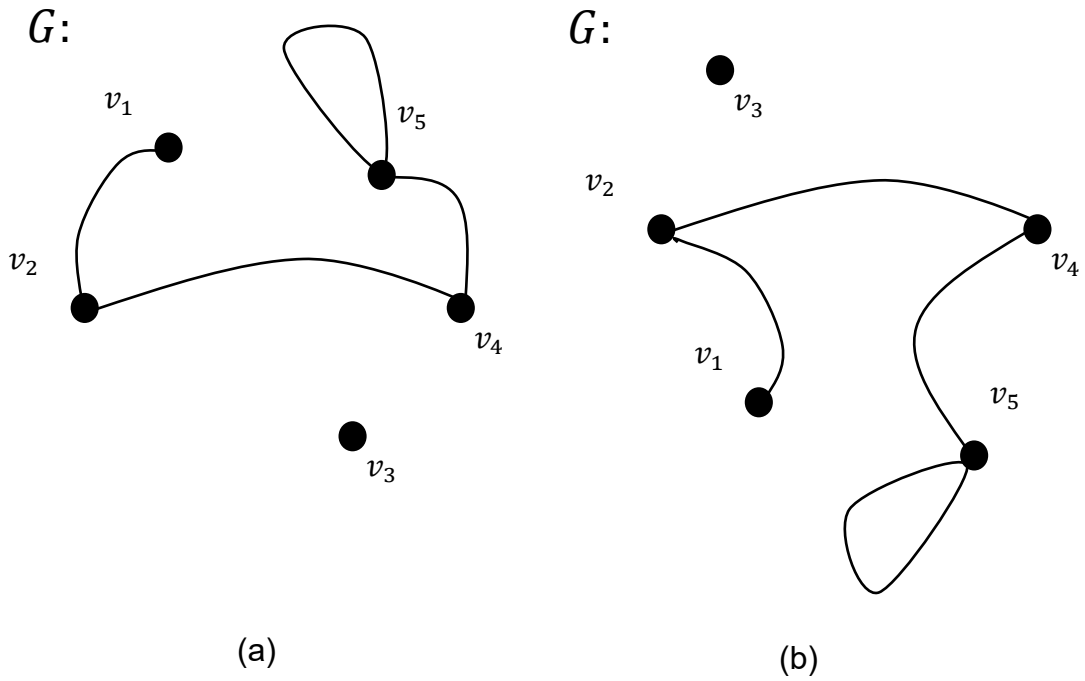


FIGURE 8 - (a) Representation of graph G. (b) Another way to represent it.
SOURCE: The author (2016).

2.2.2 DEFINITIONS

In the next subsections, a brief overview of graph theory definitions is given to aid the understanding of the graph approaches used along this work.

2.2.2.1 Subgraph

A graph H is called subgraph of a graph G if every vertex and edge in H is a vertex and edge in G (BONDY; MURTY, 2008). FIGURE 9 shows a graph and some of the possible subgraphs

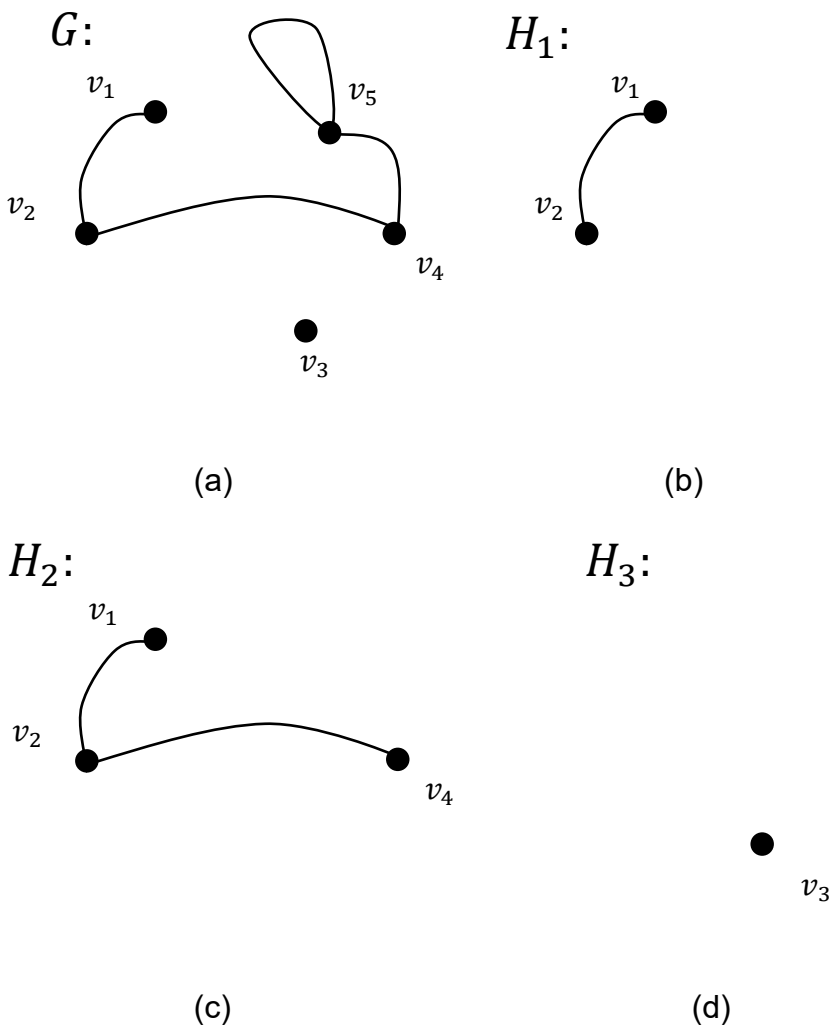


FIGURE 9 (a) Graph G AND (b)(c)(d) some of the possible subgraphs.
SOURCE: The author (2016).

2.2.2.2 Connectivity

A graph is called connected if for every partition of its vertex set into two sets A and B there is, at least, one edge connecting A and B. On the other hand if this edge does not exist, the graph is called disconnected (BONDY; MURTY, 2008). An example is shown in FIGURE 10, where G is connected because starting from any vertex it is possible to reach any other. Additionally, H is disconnected as there is no edge connecting v_1 or v_2 to the other vertices.

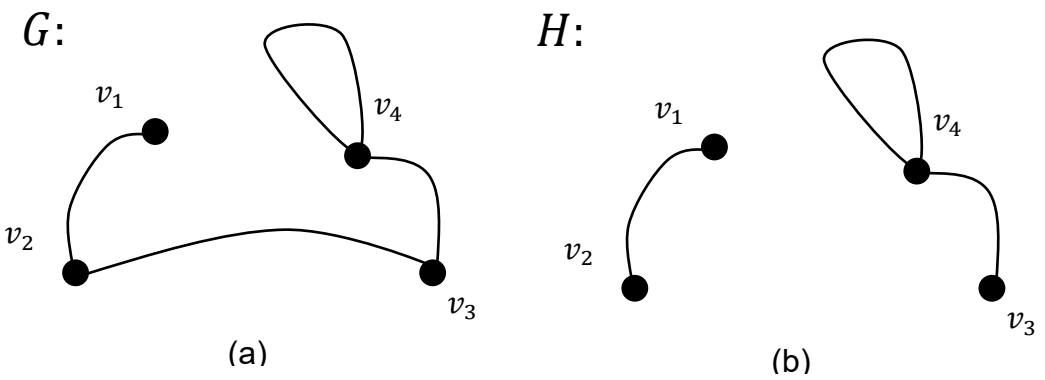


FIGURE 10 - (a) G is a connected graph while H is disconnected.
SOURCE: The author (2016).

2.2.2.3 Connected component

A maximal connected subgraph is the largest possible connected subgraph; this means that it is not possible to find another vertex in the graph such that it could be added to the subgraph without disconnecting it. A connected component of a graph is a maximal connected subgraph of it (BONDY; MURTY, 2008; DIESTEL, 2006). FIGURE 11 shows a graph G with two maximal connected subgraphs.

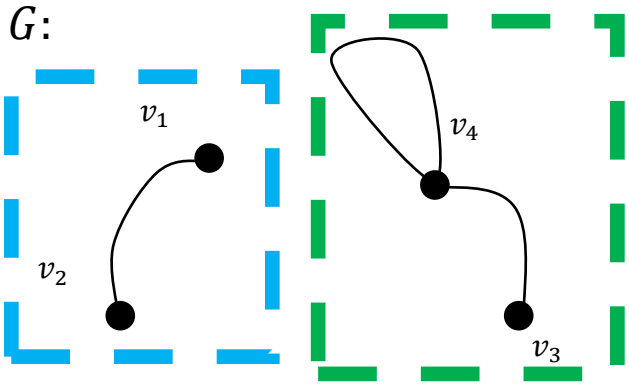


FIGURE 11 - A graph G with two connected components (one of them surrounded by a blue box and the other by a green box).
SOURCE: The author (2016).

2.2.2.4 Cycles

A graph is a cycle if it contain the same number of vertices and edges and if its vertices can be rearranged in the shape of a circle in a way that two vertices are adjacent if and only if they appear consecutively along the cycle

(BONDY; MURTY, 2008; WEST, 2001. FIGURE 12 shows two examples of graphs that are cycles.

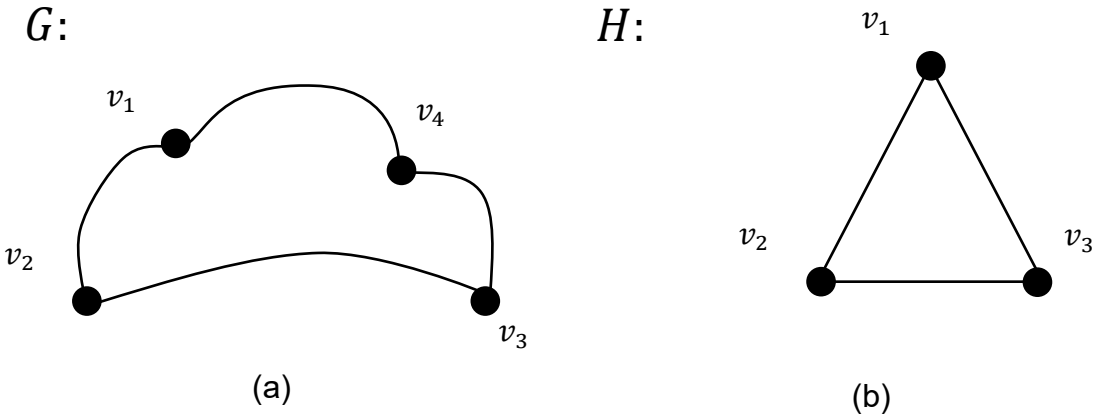


FIGURE 12 - (a) G is a cycle with four vertices. (b) H is a Cycle with three vertices.
SOURCE: The author(2016).

2.2.2.5 Trees and Forest

The world tree brings the idea of a root and a branching starting from it. In graph theory, a tree is a connected graph with no cycles (BONDY; MURTY, 2008; WEST, 2001). A disconnected graph with no cycles is a forest, and each of its components is a tree (DIESTEL, 2006). FIGURE 13 shows a forest and examples of trees.

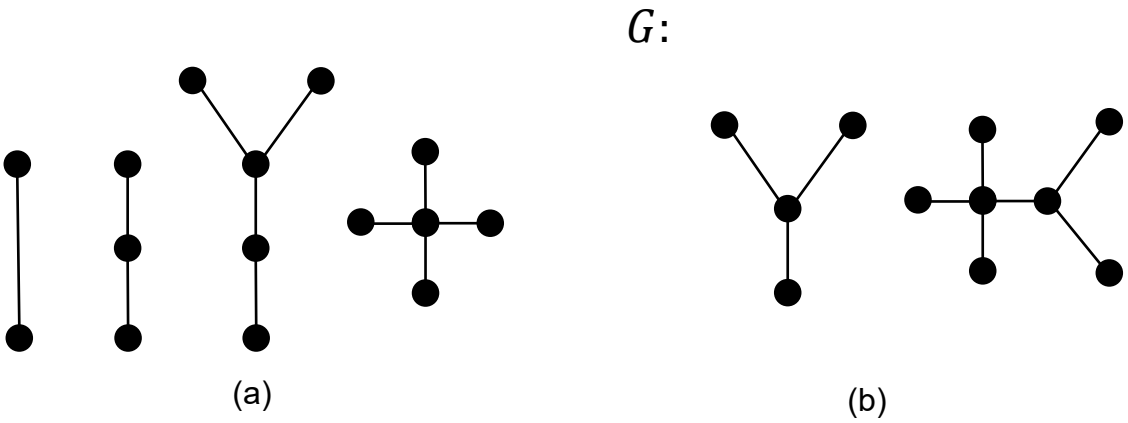


FIGURE 13 - (a) examples of trees (b) The GRAPH G is a forest with two connected components, where each one is a tree.
SOURCE: THE AUTHOR(2016).

2.2.3 APPLICATIONS IN IMAGE PROCESSING

Graph theory has different applications in computer vision, as image segmentation and recognition (FELZENSZWALB; HUTTENLOCHER, 2004; SONKA et al., 2014). In image segmentation, normally the image is modeled as a graph where each pixel is a vertex, and there is an edge connecting every neighbor pixel. A weight is associated with every edge, where the edge weight is related to the similarity between its ending vertices. Segments are then constructed by removing (cutting) edges. Different algorithms based on graph theory were created with this objective as minimum cut (WU; LEAHY, 1993), normalized cuts (SHI; MALIK, 2000), minimum spanning tree (FELZENSZWALB; HUTTENLOCHER, 2004) and so on.

Commonly in recognition, images are modeled as graphs. Where graph comparison algorithms are used to determine if the images are similar or not. This is performed by comparing the graph representation of one image with the graph representation of the other (SONKA et al., 2014). To detect if one image contains the other (the case of object detection), it is possible to compare the graph representation of the object in one image with subgraphs of the other image (SONKA et al., 2014). Some examples are the works of Luo and Hancock (2001), that perform image matching using graphs, of Torsello and Hancock (2003) for shape matching and of Robles-Kelly and Hancock (2005) that uses graphs for pattern recognition.

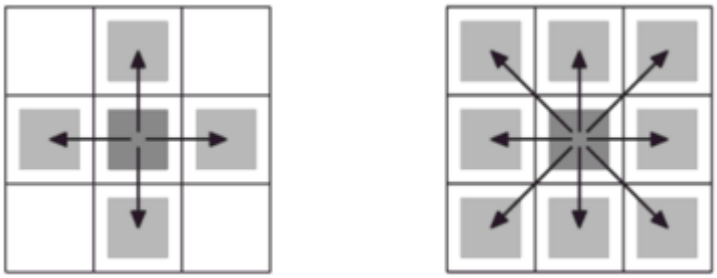
To identify regions with unique characteristics connected components may be used, to explain this technique first how connectivity occurs in images is elucidated followed by the connected component analysis.

2.2.3.1 Connectivity analysis in images

With slight adaptations, the concept of connectivity introduced for graph theory may be applied to image processing. Note that digital image connectivity can be interpreted as a graph, where each pixel is looked as one vertex, and there are edges linking neighbor pixels with the same value.

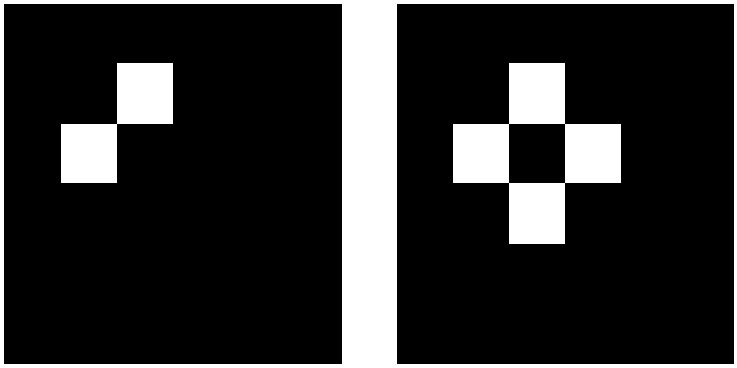
The connectivity among pixels may be verified using the concept of neighborhood. Rosenfeld and Pfaltz (1966) and Rosenfeld (1970) proposed two main neighborhood relations: 4-connected and 8-connected. In the case of 4-

connected neighborhood, pixels from north, south, west and east are neighbors of the central pixel. As expected, 8-connected consider the diagonal pixels as neighbors too. FIGURE 14 shows an example:



(a) (b)

FIGURE 14 - Center pixel in dark gray, Pixels considered neighbors in light gray. (a) 4-connected neighborhood. (b) 8-connected neighborhood.
SOURCE: Adrian and Westerweel (2011).



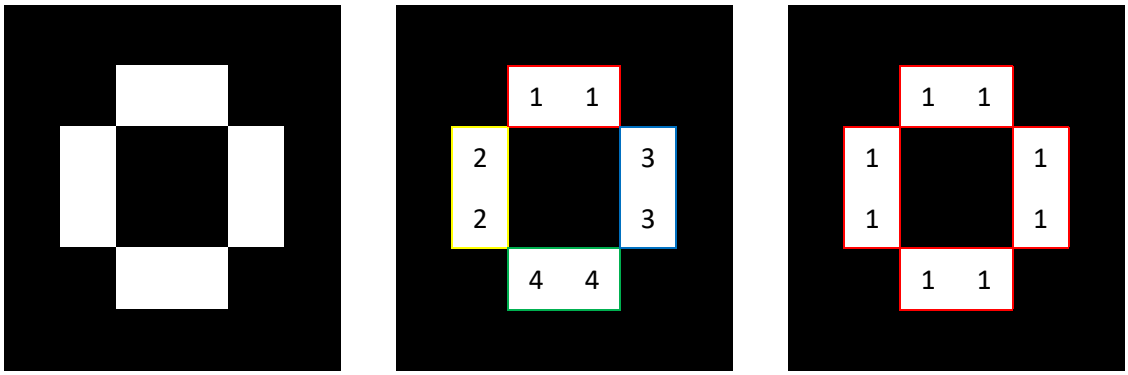
(a) (b)

FIGURE 15 - Paradox examples.
SOURCE: The author (2016).

The use of 4-connected or 8-connected neighborhood in algorithms may cause different results. Considering binary images if using 4-connected the FIGURE 15(a) show two white pixels not connected, on the other hand, if using 8-connected the pixels are connected. As noted by Rosenfeld and Pfaltz (1966) the type of connectivity may cause a paradox, take as an example the FIGURE 15(b), if 4-connected is applied all the white pixels are disconnected, while the black ones are separated into two components. On the contrary, if 8-connected is used there is only one black component and one white component, what is counter intuitive as it is expected two black components and one white. Duda et al. (1967 apud. ROSENFELD, 1970) suggested using 8-connected for the white pixels and 4-connected for the black pixels, solving the paradox for this case.

2.2.3.2 Connected components labeling

As the name implies, connected components labeling applies a unique label to each connected component of a given graph. When applying it to image processing, it is necessary to use the connectivity concept introduced in subsection 2.2.3.1. It seeks to form sets of connected pixels (connected component), where only similar connected pixels are in the same set. If there is no interest in background pixels (black), one common approach is to consider all black pixels as a unique component (background) and only analyze the white ones (foreground). In this way, the paradox problem mentioned in previous section vanishes (Dawson-Howe, 2014). Once the sets are arranged, each one receives an integer number (label), so if at the end of the process there are K sets, the sets may be numbered for example from 1 to K. FIGURE 16 shows an example of connected component labelling over an image using 4-connected and 8-connected for white pixels.



(a) (b) (c)

FIGURE 16 - (a) Input image. Connected component labelling using (b) 4-connected and (c) 8-connected.
SOURCE: The author (2016).

The connected component labeling is important to extract information from a binary image because it moves the analysis from an individual pixel to a higher level of information extracted from the component, like size, shape, and location.

Since this is a special case of the application of the graph theory, several algorithms are easily applied to images with almost no modification. The first connected component labeling algorithm proposed for images is called classical, as it is based on algorithms for graphs, is the algorithm of Rosenfeld and Pfaltz

(1966). It makes use of depth-first search, which will be later described. Other algorithms and implementations were proposed, differing in speed and memory requirements as algorithms of Manohar and Ramapriyan (1989), Suzuki et al. (2003), Johnston and Bailey (2008), Hawick et al. (2010), and Kalentev et al. (2011). The book of Kong and Rosenfeld (1996) brings a list of different implementations and a detailed explanation of each one.

2.3 EDGES

Edges are points where a strong signal variation occurs in an image intensity function (CYGANEK; SIEBERT, 2009; SONKA et al., 2014). The reason of this variation may vary, being caused by shadows, highlights, depth discontinuities, color/texture change or event surface normal discontinuities (SONKA et al., 2014). FIGURE 17 shows an image with the presence of these different phenomena and the respective edge image.

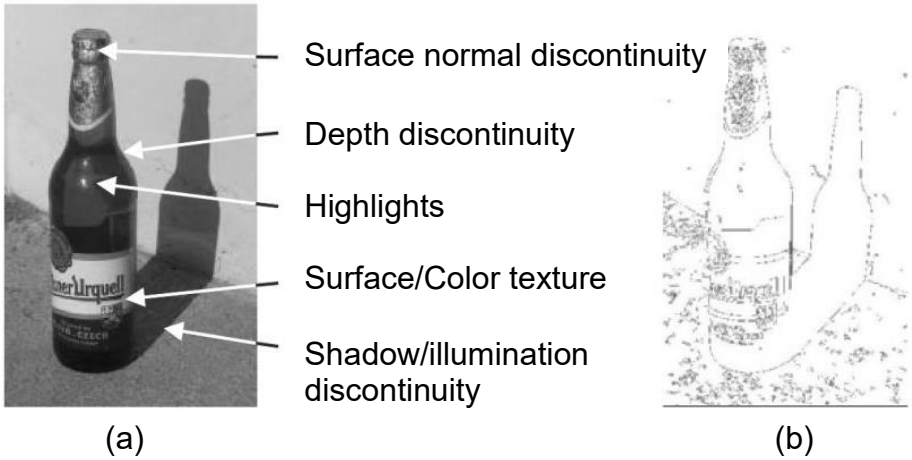


FIGURE 17 - (a) Different physical phenomena related with the formation of edges in images. (B) Edges detected by canny's method.
SOURCE: Sonka et al. (2014).

As noted by different authors, edges play an important role in the human visual system, being a crucial feature to our image perception. Using electrodes plugged into the brain of a cat, Hubel and Wiesel (1959) discovered neurons that respond to edges in a specific orientation. Analyzing functional magnetic resonance imaging (fMRI) patterns from humans viewing either line or color images, Walther et al. (2011) suggested that the information used to categorize each scene is similar for both cases, what indicates that it is edge based (see FIGURE 18) (BANSAL et al., 2013). This is corroborated by another study, which

put in evidence that people with damage in brain part responsible for the perception of contours become unable to recognize objects (ZEKI, 1993).

FIGURE 18 - Color and line drawing from six categories.
SOURCE: Walther et al. (2011).

Considering the importance of edges to biological visual systems, it is expected that they play an important role as well in most artificial vision applications. It is not surprising that this field has been studied for the at least 50 years since the development of the first edge detection techniques as Roberts edge detector (ROBERTS, 1963). In the next section, it is described the basic idea on how to detect edges in images, and Robert, Prewitt, Sobel and Canny detectors are detailed (CANNY, 1986; DANIELSSON; SEGER, 1990; PREWITT, 1970; ROBERTS, 1963). Second-order approaches and frequency analysis techniques are briefly commented as well.

2.3.1 EDGE DETECTORS

An edge may be mathematically described as a vector variable with magnitude and direction components (SONKA et al., 2014). Considering the image as a function $f(x,y)$, where x and y is the pixel coordinate and $f(x,y)$ is its intensity function, an edge can be detected by differentiating adjacent points. As the image is a multivariable function, the edge magnitude is the gradient magnitude and the edge direction is the gradient direction rotated by -90º (SONKA et al., 2014). Equations 1 and 2 shows the gradient magnitude ∇f and the direction θ respectively.

$$|\nabla f(x,y)| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}. \quad (1)$$

$$\theta = \tan^{-1}\frac{\frac{\partial f}{\partial y}}{\frac{\partial f}{\partial x}} \quad (2)$$

where $\frac{\partial f}{\partial x}$ and $\frac{\partial f}{\partial y}$ are the derivatives of the intensity function $f(x, y)$ in x and y direction respectivally.

This can be easily understandable seeing the FIGURE 19, where the image is represented as a height field and the edges only occur at steep slopes.

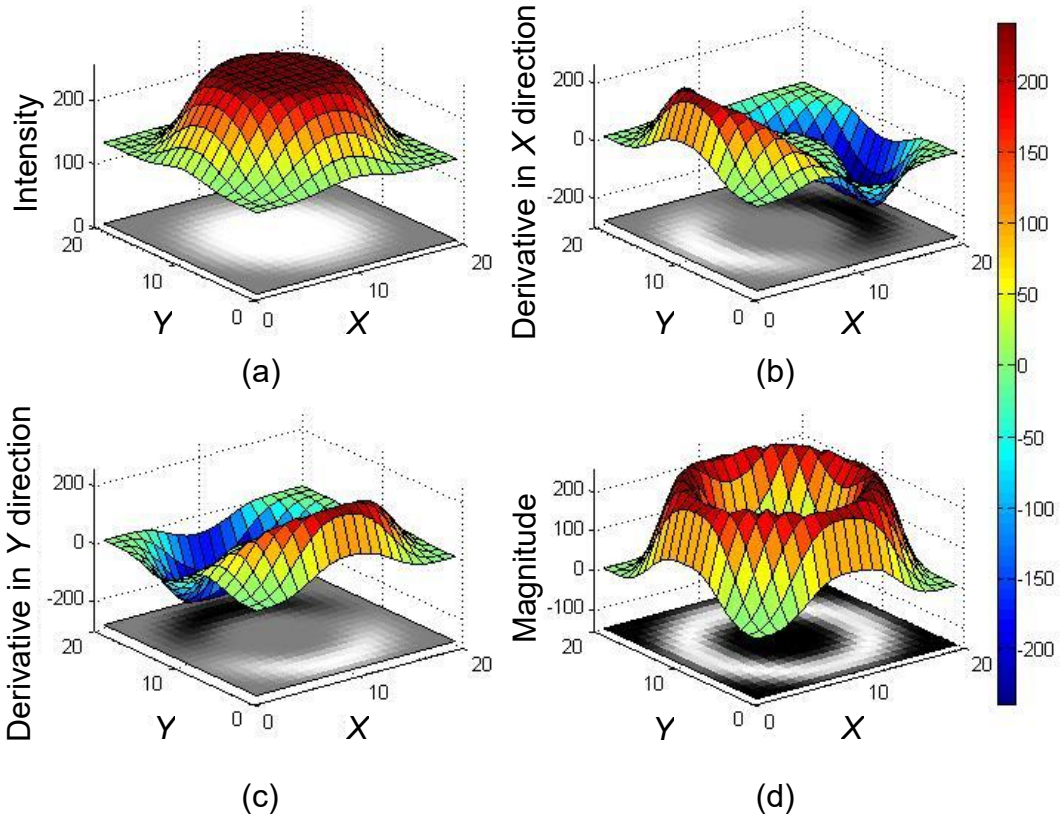


FIGURE 19 - All graphs show the associated image and the height field ploted over it. (a) Input image. (b) Derivative response in *X* direction. (c) Derivative response in *Y* direction. (d) Gradient Magnitude.
SOURCE: The Author(2016).

Due to the discrete nature of a digital image, before applying the equations (1) and (2), it is necessary to perform some kind of numeric approximation over the neighborhood of a pixel to obtain the partial derivatives

(GONZALEZ; WOODS, 2008). This can be performed over an image using a mask.

As the dimensions and value of the mask affect the output image directly, different authors proposed different masks, also known as operators.

One of the first operators, the Robert cross-gradient operator is computed in a 2 x 2 neighborhood in a diagonal fashion Roberts (1963). The magnitude is calculated as shown in equation 3. The mask is shown in FIGURE 20, where

$$|\nabla f_{x,y}| \cong \sqrt{\left(f_{x,y} - f_{x+1,y+1}\right)^2 + \left(f_{x+1,y} - f_{x,y+1}\right)^2} \quad (3)$$

-1	0
0	1

(a)

0	-1
1	0

(b)

FIGURE 20 - (a) First Roberts operator diagonal mask. (b) Opposite diagonal.
SOURCE: The author (2016).

Despite the simplicity of the 2 X 2 masks, they are not adequate for computing edge direction as the pixel under calculation cannot be centered (GONZALEZ; WOODS, 2008). Furthermore, masks with this dimension have high sensitivity to noise due to the fewer pixels used during the calculation if compared with bigger masks (SONKA et al., 2014).

Prewitt operator (PREWITT, 1970) approximation of the derivative using a 3 X 3 mask incorporates more elements than Robert Cross operator, what let this operator less sensitive to noise and due to the symmetry around the central point it carries more information regarding edge direction (GONZALEZ; WOODS, 2008; NIXON; AGUADO, 2012). The masks are shown in FIGURE 21.

-1	-1	-1
0	0	0
1	1	1

(a)

-1	0	1
-1	0	1
-1	0	1

(b)

FIGURE 21 - Prewitt operator mask to detect (a) horizontal edges and (b) vertical edges. SOURCE: The author (2016).

The Sobel operator (DANIELSSON; SEGER,1990) is similar to Prewitt, but it gives emphasis on pixels closer to the central pixel. The 3 X 3 Sobel mask is shown in FIGURE 22.

-1	-2	-1
0	0	0
1	2	1

(a)

-1	0	1
-2	0	2
-1	0	1

(b)

FIGURE 22 - Sobel operator mask to detect (a) horizontal edges and (b) vertical edges. SOURCE: The author (2016).

Both Prewitt (PREWITT, 1970) and Sobel (DANIELSSON; SEGER,1990) operators smooth the gradient (DAWSON-HOWE, 2014). The Prewitt operator smooths by simple averaging while the Sobel by weighted average. This gives the Sobel operator better noise-suppression, which is an important characteristic when dealing with derivatives (GONZALEZ; WOODS, 2008). Another common characteristic between Sobel and Prewitt is that as the masks are centered in a point, there is no shift in the final position of the found edges. In opposite with the Roberts operator, which cause a ½ pixel shift (DAWSON-HOWE, 2014).

FIGURE 23 shows different results of the Roberts, Prewitt, and Sobel operators applied to the same image after converting it to grayscale, one unmodified and the other with Gaussian filter applied to it. The drawback of using directly gradient to edge detection is that derivatives increase high frequencies and as the noise to signal ratio increases at higher frequencies it ends up increasing noise as well. This is easily perceived in some results shown in FIGURE 23. Therefore, it is prudent to filter the image before performing any edge detection method.

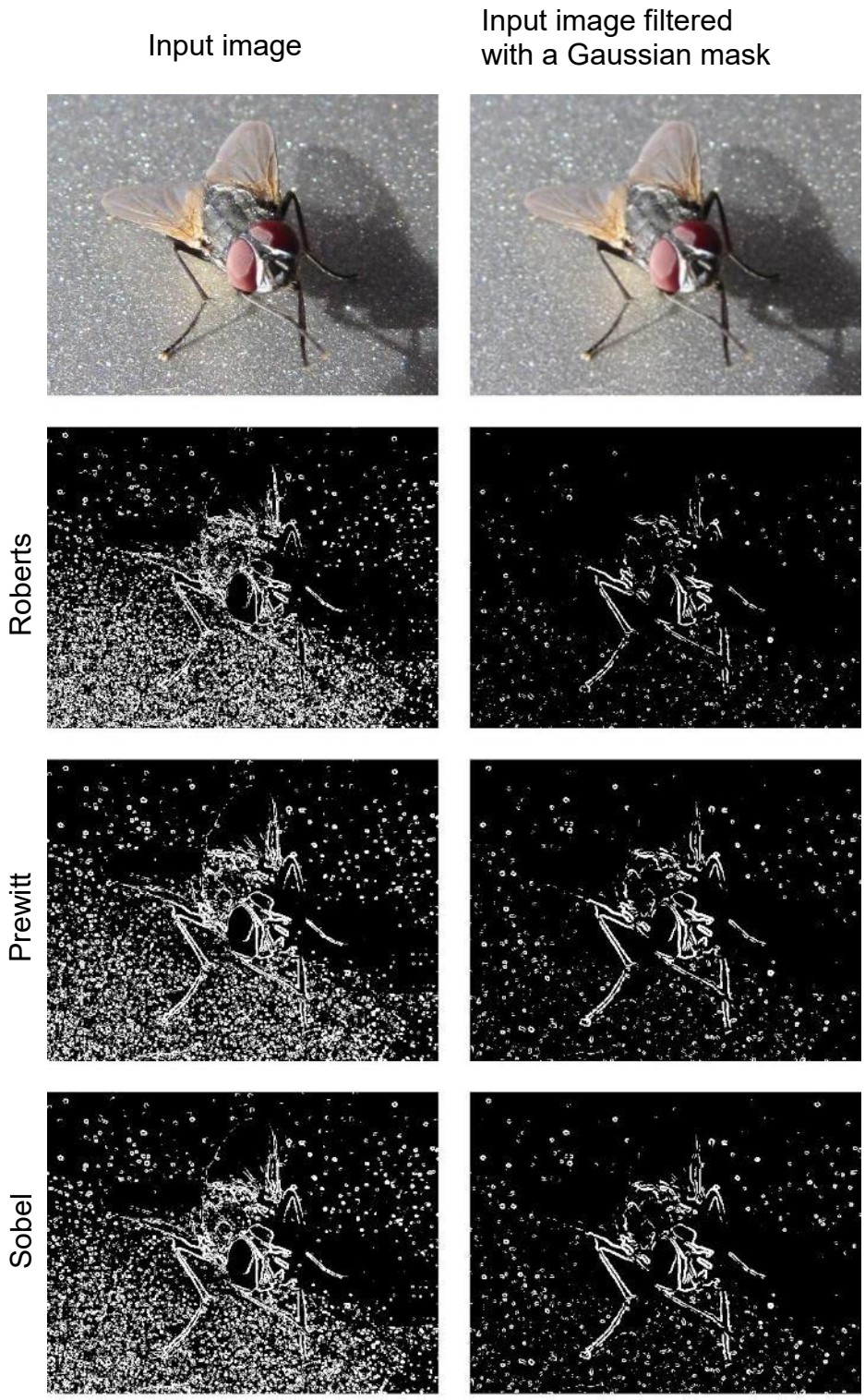


FIGURE 23 - Different operators applied on the same image, the left one without Gaussian filtering and the right one with it.
SOURCE: The author (2016).

Note that even if the Gaussian low pass filter is applied before the differentiation, another drawback remains: the border thickness. The Canny filter was designed to circumvent the commented issues by applying differentiation,

Gaussian filtering and border thinning. A detailed description of the Canny edge detector is given in the next section.

2.3.2 CANNY EDGE DETECTOR

Canny (1986) wrote the paper describing the detector know by his name with three performance criteria in mind:

1. Obtain good detection maximizing the noise-to-signal ratio in order to obtain a low probability of failing to mark real edges and a low probability of falsely marking a non-edge point as an edge;
2. Obtain accurate localization by marking the edge point as close as possible to the center of the real edge;
3. Obtain only one response per edge. As Canny stated this is implicitly captured in the first criterion, but he made it explicit here, as the mathematical form of the first criterion did not capture the multiple response requirements.

To respect the stated criterions the Canny edge detector performs the steps described in the subsections 2.3.2.1 to 2.3.2.5.

2.3.2.1 Gaussian Filtering

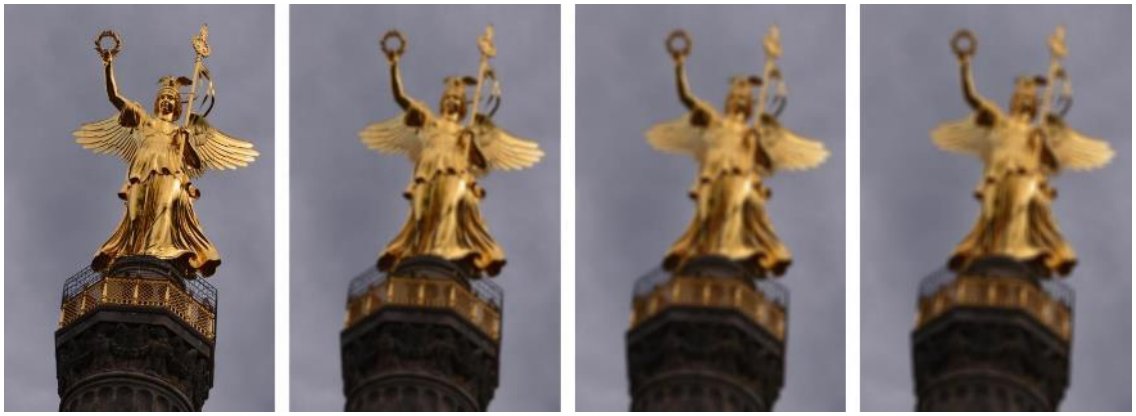
The first step is to apply Gaussian filtering to the image to remove noise. The Gaussian function $G(x, y)$ is defined as:

$$G(x, y) = e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{4}$$

where x and y are coordinates and σ the standard deviation.

Due to the symmetry of the Gaussian function, it can be applied on the image using convolution or cross-correlation, and the results will still be the same. The output of this step is a smoothed image f_s filtered by the low-pass Gaussian Function. This is performed convolving the Gaussian function G with the image f using the equation 5, where

$$f_s(x, y) = G(x, y) * f(x, y) \tag{5}$$



(a) (b) (c) (d)

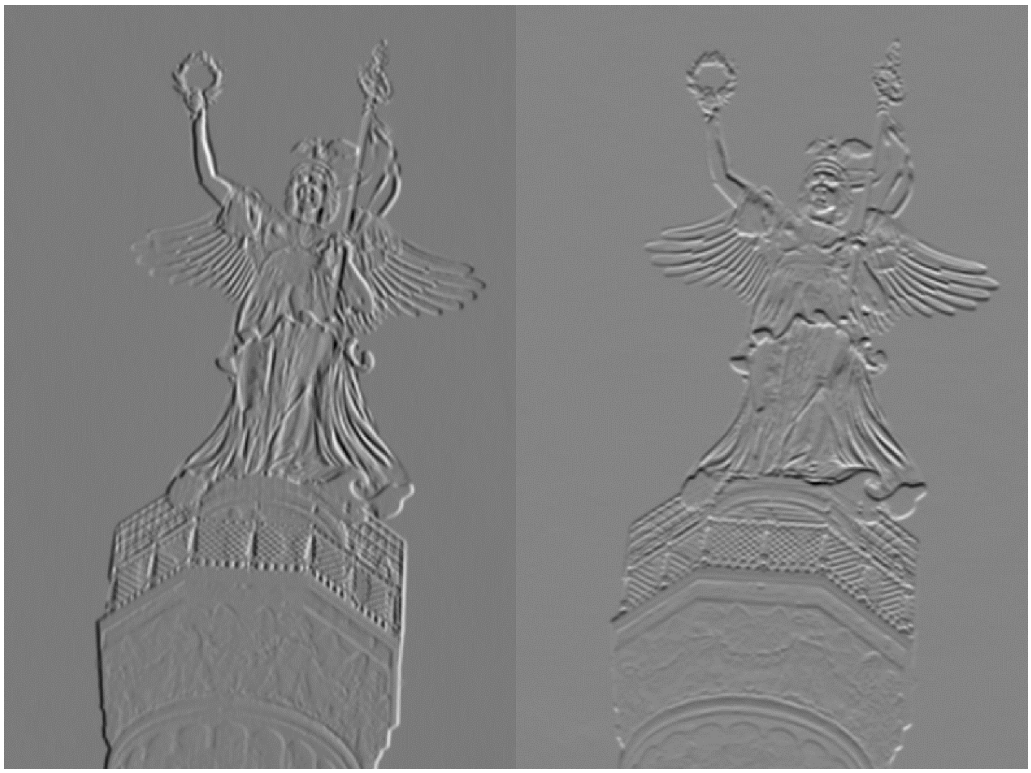
FIGURE 24 - (a) A picture of the Victory Column (Berlin, Germany) used as input image. (b), (c) and (d) shows (a) filtered using Gaussian Filter with sigma equals to 2, 4 and 8 respectively.

SOURCE: The author (2016).

2.3.2.2 Find the gradient magnitude and direction

After the Gaussian filter is convolved with the image, the gradient magnitude, and direction of each pixel are calculated from the smoothed image using the already defined equations 1 and 2.

FIGURE 25 shows the result of applying the derivatives $\frac{\partial f}{\partial x}$ and $\frac{\partial f}{\partial y}$ over the grayscale version of the input image and FIGURE 26 shows the magnitude and direction calculated using equations 1 and 2.



(a) (b)

FIGURE 25 - Derivatives calculated along (a) *X* Direction and *Y* direction using Sobel.
SOURCE: The author (2016).

(a) (b)

FIGURE 26 - (a) Magnitude and (b) direction of the gradient.
SOURCE: The author (2016).

2.3.2.3 Non-maxima suppression

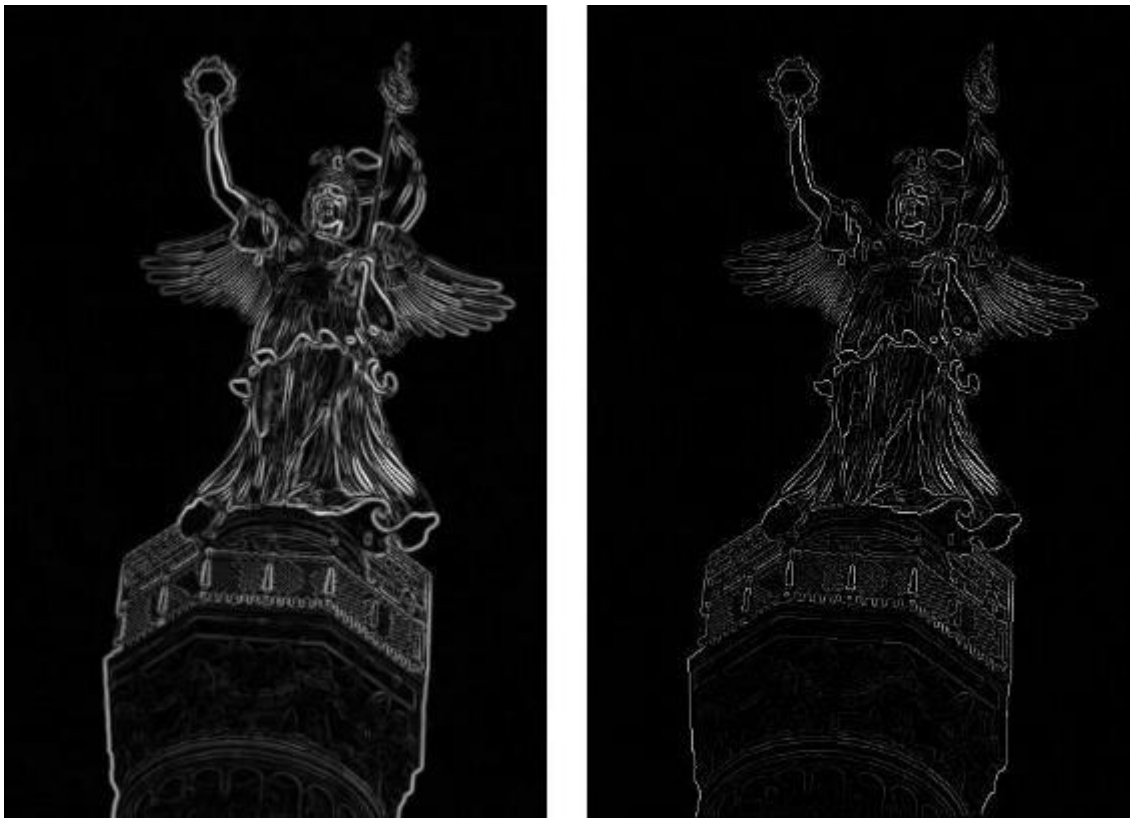
During this step only the maxima edge in maintained. This is performed by checking the neighbors (*e.g.* 8-connected) of each edge in which the gradient points in the “same” direction and keeping only the center edge if it is a maximum. There are different ways to perform this step; one is to use the algorithm shown in FIGURE 27.

Non-maxima suppression Algorithm

1. Consider 8 possible edge directions. Each compromising 360/8 degrees
2. **For** each pixel *p* in the GradientMap(*i,j*)
3. Get 2 neighbor pixels *np* that are orthogonal to the direction of *p*
4. **If** any *np* gradient is greater than *p* gradient
5. output(*i,j*)=0
6. **Else**
7. output(*i,j*)=GradientMap(*i,j*)

FIGURE 27 - Non-maxima suppression algorithm.
SOURCE: Dawson-Howe (2014).

After applying the non-maxima suppression algorithm, the edge of the last step is now thinned. FIGURE 28 shows a comparison of the original Gradient Magnitude image with the Gradient Magnitude after non-maxima suppression.



(a) (b)

FIGURE 28 - (a) Gradient magnitude. (b) Gradient magnitude after non-maxima-suppression.
SOURCE: The author (2016).

2.3.2.4 Hysteresis thresholding

To perform hysteresis thresholding two thresholds must be set, one called low and other called high. If a point magnitude is over the high threshold, this point is considered an edge immediately. On the other hand, if a point is below the low threshold it is not added to the output. The point with a magnitude between the low and high threshold are kept if they are part of a connected component with at least one point with a magnitude over the high threshold. FIGURE 29 let this step clearer, observe item (e) where points in green connected with red points are kept, while disconnected green points are removed.

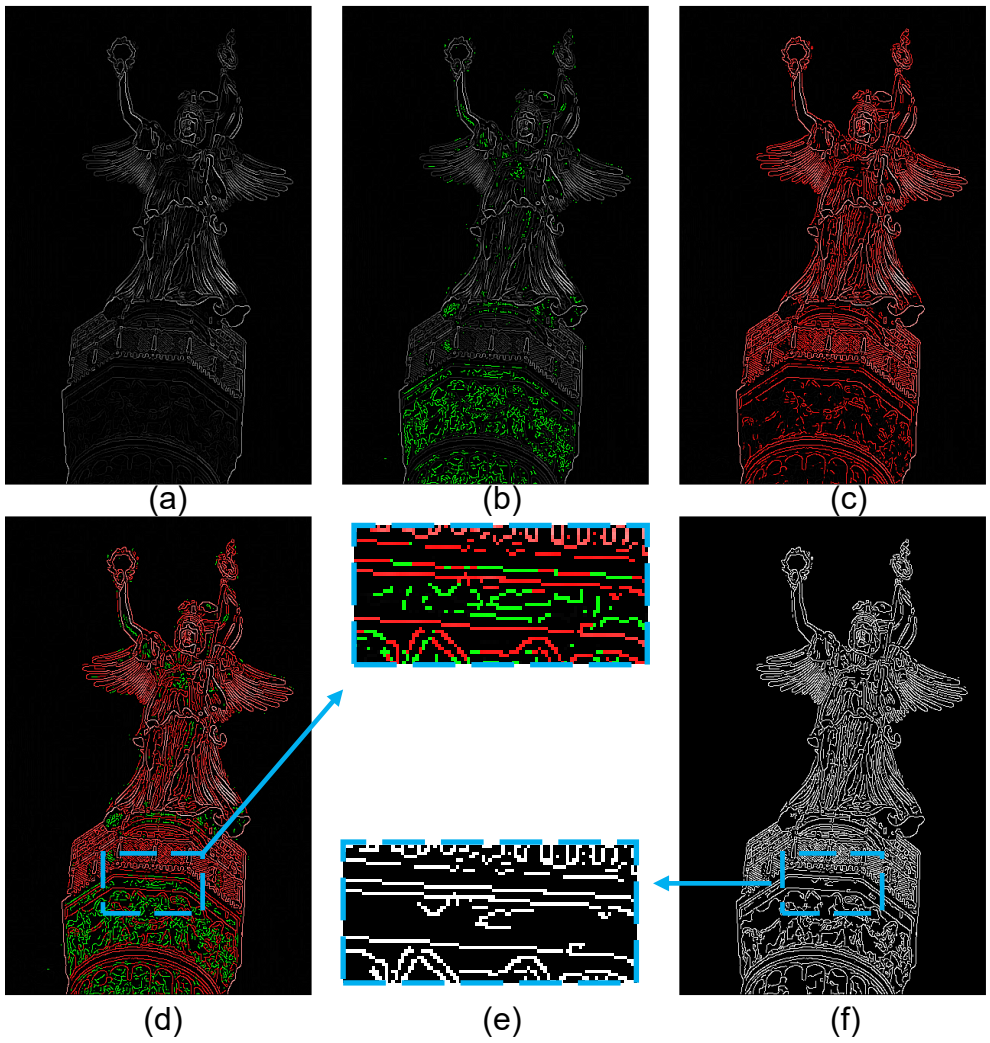


FIGURE 29 - (a) Non-Maximum Suppression output. (b) In green Point between the low threshold and high threshold. (c) In read Points above the high threshold. (d) Point above low threshold. (e) Zoom over the same area of (d) and (f). (f) shows the final output.
SOURCE: The author (2016).

2.3.2.5 Feature Synthesis

The steps from 2.3.2.1 to 2.3.2.4 are repeated with increasing values of sigma to the Gaussian filter. A synthesized output is created considering that the only edges are the edges from the smaller operator (but using the next sigma value). If the difference between the outputs of the synthesized output and the current is significant, new edges are added to the final output. Canny is not clear for how long this process should continue or where it should start. Due to the complexity added to the edge detector, this step is commonly skipped in different implementations, *e.g.* MATLAB edge function.

2.3.3 OTHER APPROACHES

Besides the first order differential presented operators, there are as well other approaches such as second order differential operators. Second order techniques are also known as zero crossing techniques, due to the fact that the step edge position is where occurs a zero crossing in its second derivative.

One example of a second order differential operator is the Marr-Hildreth operator that applies a Gaussian filter and the Laplacian (sum of second order partial derivatives) and ends by finding zero-crossings (MARR; HILDRETH, 1980). The Laplacian and Gaussian may be convolved with an image in a single step by convolving both filters and then applying them to the image. The resulting operator is known as Laplacian of Gaussian or LoG for short.

Based on Canny's work, Spacek (1986) suggested the use of a different noise suppressing filter and a modified performance measure. Different from Canny and Spacek, Petrou and Kittler (1991) proposed an operator that uses a different model of edge, a ramp instead of a step edge.

An alternative to the presented differential approaches is to use phase congruency. This method detects edges in the image by analyzing the image in the frequency domain. Phase congruency method makes use of the fact that edges are perceived as points where each of the frequency components is maximally in phase. This characteristic allows it to detect edges despite the image contrast (NIXON; AGUADO, 2012). Normally what differs the implementation is the measure used to detect where the phase congruency occurs, such as the proposed by Morrone and Owens (1987), and Kovesi (2003).

2.4 STEREO VISION

Stereo vision is a term coined from Greek, where stereo means "solid" (HOWARD; ROGERS, 1995), therefore stereo vision is "solid vision" what implies 3D vision. The 3D vision may be generated using monocular clues, binocular vision (2 cameras), trinocular (3 cameras) and so on. Here it is considered the binocular case.

Binocular vision is the term used to refer to "having two eyes" (HARPER, 2016). Therefore, all animals with two eyes, even if they are on opposed sides of the face, present binocular vision. However, this term is usually used for animals

that present at least a small overlap between the field of view of each eye and use it to code depth information (HOWARD; ROGERS, 1995). FIGURE 30 shows an example of a field of view of each eye and the overlap region.

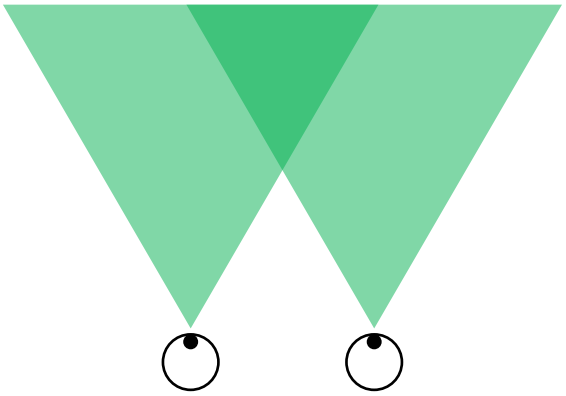


FIGURE 30 - Light green represent single eye view. Darker green represents a region seen by both eyes.
SOURCE: The author (2016).

In spite of the first records of study regarding binocular vision dates back to the ancient Greeks, a deeper understanding of it only happened about two centuries ago (CYGANEK; SIEBERT, 2009; HOWARD; ROGERS, 1995). Possibly this took so long due to our powerful vision system, capable of coding depth even with a single eye, making less apparent the importance of binocular vision for depth perception (HOWARD; ROGERS, 1995).

Noticing that the human brain is capable of recognizing depth information from the slight difference from the views of each eye, Sir Charles Wheatstone at the beginning of the 19th century created the stereoscope, a device for depth perception from two views of the same scene (CYGANEK; SIEBERT, 2009). In the 1950s started to appear the first 3D movies making use of this characteristic (SONKA et al., 2014). Characteristic which equally allow one to recover depth information from two pictures of the same scene combined with sensor geometry (SONKA et al., 2014). Therefore, with the advances of a digital computer, the first algorithms were designed in the 1970s to recover depth information from different views of the same scene (SCHARSTEIN; SZELISKI, 2002). Different algorithms were proposed since then and in order to evaluate their performance, Scharstein and Szeliski (2002) wrote an overview and created a benchmark of stereo algorithms in 2002. This topic currently is more than biological field research, it is a computer vision research field as well and has been broadly studied and still is

one of the most active research areas with more efficient and fast algorithms being developed every day (SZELISKI, 2010).

2.4.1 Pinhole camera model

When light strikes an object, part of it is absorbed and another part reflected. An image capture device, such as an eye or a camera can detect the reflected part (SONKA et al., 2014). One way to model this physical phenomenon is to use the pinhole camera model (BRADSKI; KAEHLER, 2008).

The pinhole camera model states that all rays of light coming from outside the camera passes thought a single pinhole and are projected into a plane positioned f units from the pinhole. As shown in Figure 31.

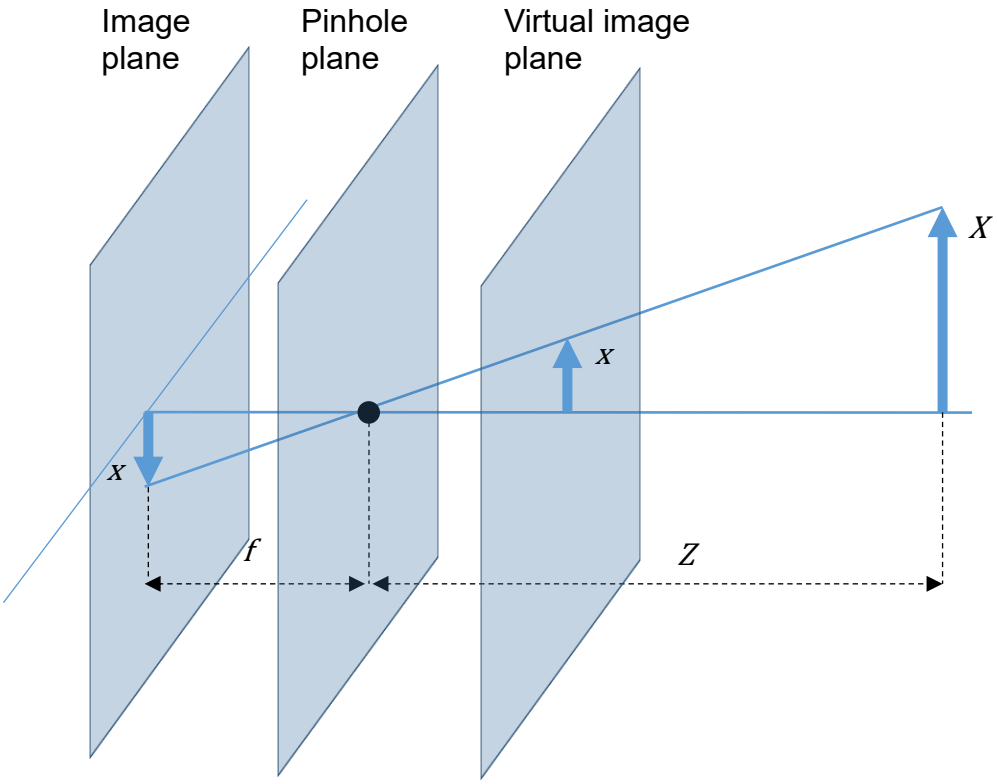


Figure 31 - Pinhole camera model. Where f is the focal length, x is the projection of X in the image plane and Z is the distance from X to the camera.
Source: Adapted from Bradski; Kaehler (2008) and Sonka et al. (2014)

Notice that using this model directly the image formed by the rays is projected upside down in the image plane. Adding a virtual plane can solve this without any prejudice, what results in an equivalent model. Using similar triangles it is possible to obtain the equation:

$$\frac{x}{f} = \frac{X}{Z} \tag{6}$$

where f is the focal length, x is the projection of the point X in the image plane and Z is the distance from X to the camera.

Notice that this model considers that the pinhole is perfectly aligned with the center of the image plane, what does not happen in reality during the manufacturing of a camera, to circumvent this, the model can consider small offsets c_x and c_y as in (BRADSKI; KAEHLER, 2008):

$$\begin{aligned} x &= f_x \frac{X}{Z} + c_x \\ y &= f_y \frac{Y}{Z} + c_y \end{aligned} \tag{7}$$

where x and y is the point coordinates in the image plane, f_x and f_y are the focal lengths. Two different equations are used to each coordinate because the offsets can be different in each axis and the pixels in an imager system normally are rectangular, what causes different focal lengths (BRADSKI; KAEHLER, 2008).

This simple model demonstrate the reason because it is not possible to know the distance from an object to the camera without knowing additional geometric information about the object.

2.4.2 DEPTH PERCEPTION

The working principle of depth perception with stereo vision is that a scene point and two camera points form a triangle. If the distance between the optical center of each camera and the angle formed by the camera rays are known, than it is possible to estimate the distance from the cameras to the object (BRADSKI; KAEHLER, 2008; SICILIANO; KHATIB, 2008). FIGURE 32 illustrates this concept using two pinhole models.

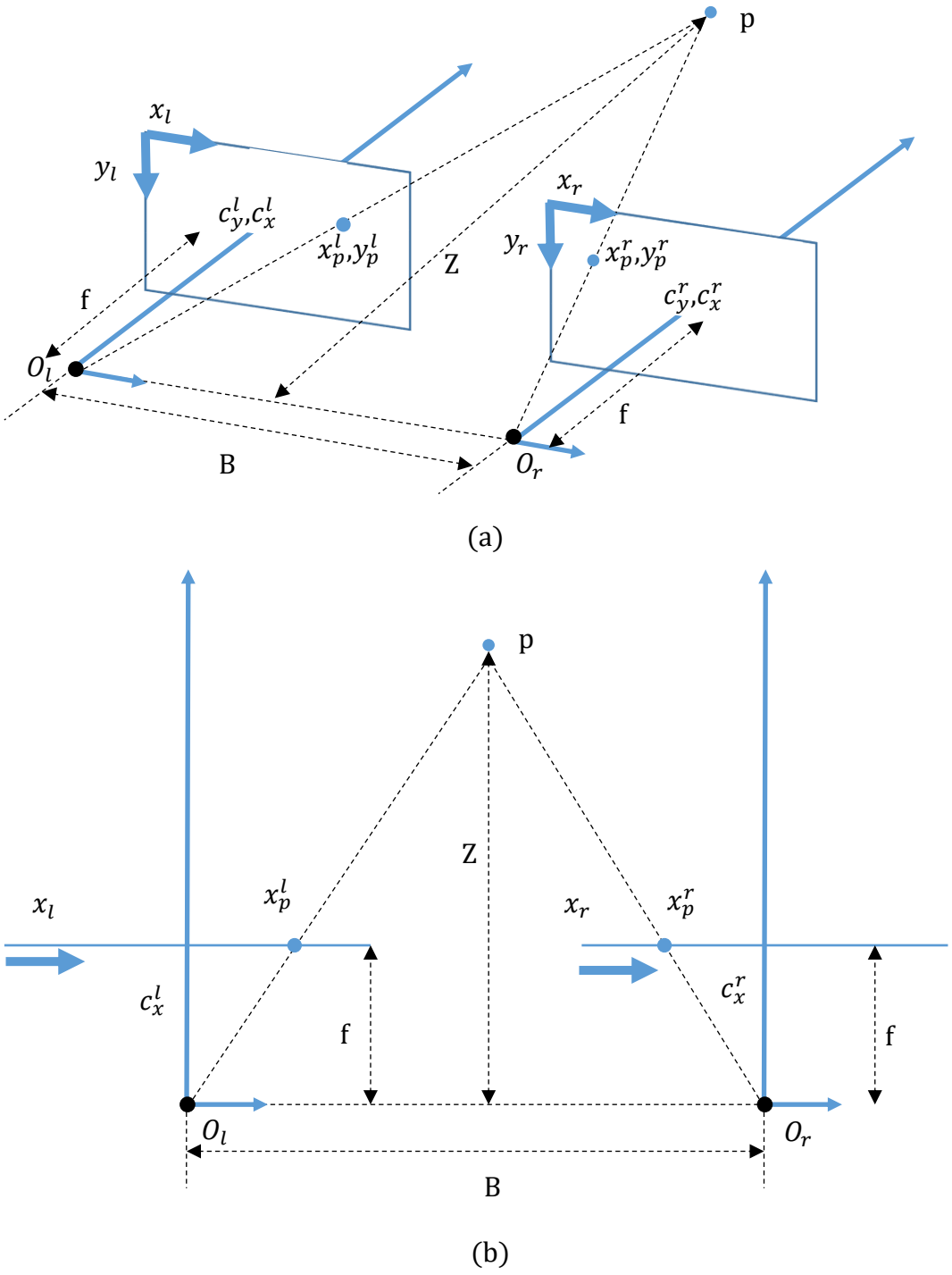


FIGURE 32 - (a) representation of Ideal stereo geometry. where a point p is projected in two views. (b) Same representation, but seen from top, ignoring y axis for easier understanding. SOURCE: Adapted from (Bradski; Kaehler, 2008; Siciliano; Khatib, 2008)

In FIGURE 32, l and and r mean left and right respectively. O is the center of projection of each camera, c is the point where the principal ray crosses the image plane, B is the baseline (distance between the centers of projection of each camera), Z is de depth of the point p, f is the camera focal length, x_p and

y_p are the horizontal and vertical coordinates of the projection of p respectively and x, y are the coordinates of each image plane.

Consider the width of an image as w and the similar triangles $\Delta x_p^l p x_p^r$ and $\Delta O_l p O_r$, then:

$$\frac{B - \left(x_p^l - \frac{w}{2}\right) - \left(\frac{w}{2} - x_p^r\right)}{Z - f} = \frac{B - (x_p^l - x_p^r)}{Z - f} = \frac{B}{Z} \tag{8}$$

Considering the disparity as $d = x_p^l - x_p^r$ and isolating it:

$$d = \frac{B(Z - f) - BZ}{-Z} = \frac{-Bf}{-Z} \tag{9}$$

Finally, isolating Z:

$$Z = \frac{Bf}{d} \tag{10}$$

Which is the formula for recovering depth from any point in FIGURE 32 scheme. Notice that the relation between the left and right horizontal coordinates projection is given by:

$$x_p^r + d = x_p^l \tag{11}$$

where d is the disparity.

One could point out that the depth was obtained considering that the cameras are perfectly parallel and with distortion free lens, arguing that this would not happen in practice. This perfectly makes sense, which is the reason because the images need to be undistorted and rectified before finding depth, what is explained in the next sections.

2.4.3 LENS DISTORTIONS

An aberration is any deviation of a lens system when compared to the pinhole projection model (CLAUS, 2007). Philipp Ludwig von Seidel described in 1857 the five primary monochromatic aberrations: spherical aberration, coma, astigmatism, field curvature and distortion (BURKE, 1996; GUY, 2012).

These aberrations (known as Seidel aberrations) may be minimized during the design of a lens system using a combination of stops, multiple lens elements, and aspherical lens surfaces (CLAUS, 2007). Commercial lenses tend to assume that distortion is less noticeable for human viewing than blurring and defocus, therefore, normally the distortion is not a priority during lens design. In machine vision applications this design choice may impact negatively geometric measurements taken from images (BURKE, 1996). On the other hand, distortion does not affect the quality of the image regarding its sharpness or focus, only its shape (WALKER, 1995). In other words, different from other aberrations, distortion preserves information, allowing algorithmic correction (BURKE, 1996). Two types of distortion are covered here and detailed in the next subsections.

2.4.3.1 Radial distortion

Radial distortion causes a displacement of a given image point from its ideal location (WENG et al., 1992). This displacement occurs along radial lines that pass through the image center (CORKE, 2011). If the points are displaced away from the optical center, the distortion is called barrel distortion, otherwise pincushion distortion. FIGURE 33 shows an example of an image affected by the commented variations of radial distortion.

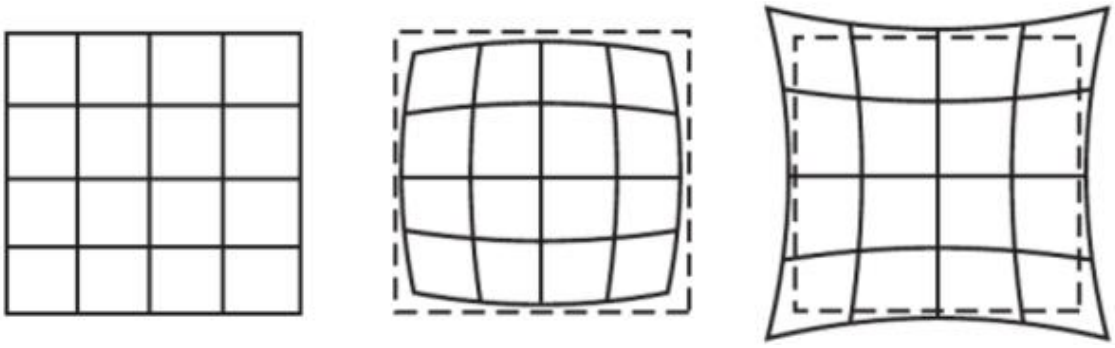


FIGURE 33 - (a) Undistorted image. (b) Barrel distortion effect. (c) Pincushion distortion effect.

SOURCE: Allen and Triantaphillidou (2012).

The radial distortion is zero at the optical center and increases near the border of the lens. This happens because the distortion is proportional to the radial distance from the optical center.

Radial distortion occurs due to the shape of the lens (BRADSKI; KAEHLER, 2008) and considering the image origin at the optical axis this behavior may be modeled as (BROWN, 1971; DAWSON-HOWE, 2014):

$$x' = x(1 + k_1 r^2 + k_2 r^4 + k_3 r^6)$$
$$y' = y(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \quad (12)$$

where $r = \sqrt{(x^2 + y^2)}$ is the distance from the current pixel at (x, y) to the optical axis,k_1, k_2 and k_3 are the coefficients describing the distortion and (x', y') are the corrected coordinates.

2.4.3.2 Decentering Distortion

The distortion described by Seidel is purely a radial effect, however, small errors in alignment and centering of multi-element lenses allows the second type of distortion called tangential distortion (CLAUS, 2007). In fact, this type of distortion owns two components: radial distortion and tangential distortion. Different from the radial distortion, tangential distortion does not occurs equally to all the lines passing through the optical center, but only to part of them. FIGURE 34 shows each component, and FIGURE 35 shows the tangential distortion effect.

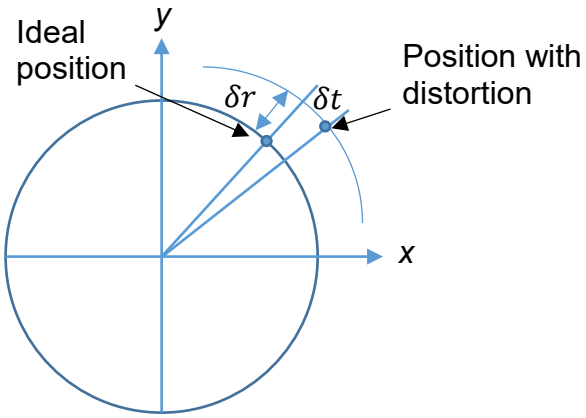


FIGURE 34 – Radial (δr) and tangential (δt) distortions. The horizontal *x* and vertical *y* coordinates intercept at the optical center.
SOURCE: Adapted from Weng et al. (1992).

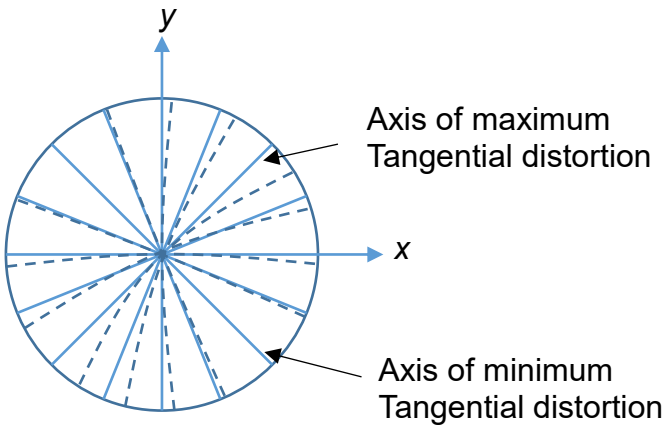


FIGURE 35 - Tangential distortion effect. Continuous lines represents a distortion free lens, while dashed lines the position shift caused by tangential distortion. The horizontal *x* and vertical *y* coordinates intercept at the optical center.
SOURCE: Adapted from Weng et al. (1992).

Decentering distortion occurs when the optical centers of lens elements are not collinear (WENG et al., 1992), what can happen due to problems during the assembly process as a whole (BRADSKI; KAEHLER, 2008).

The tangential component of this distortion can be modeled as (BROWN, 1971; DAWSON-HOWE, 2014):

$$\begin{aligned} x' &= x + 2p_1xy + p_2(r^2 + 2x^2) \\ y' &= y + 2p_2xy + p_1(r^2 + 2y^2) \end{aligned} \tag{13}$$

where $r = \sqrt{(x^2 + y^2)}$ is the distance from the current pixel to the optical axis, p_1 and p_2 are the coefficients describing the distortion, (x, y) is the pixel position and (x', y') is the pixel corrected position.

2.4.4 UNDISTORTION

To remove distortions from an image, it is necessary to identify the parameters of equations (12) and (13). With these parameters available, it is possible to use the presented distortion models to recover an undistorted image.

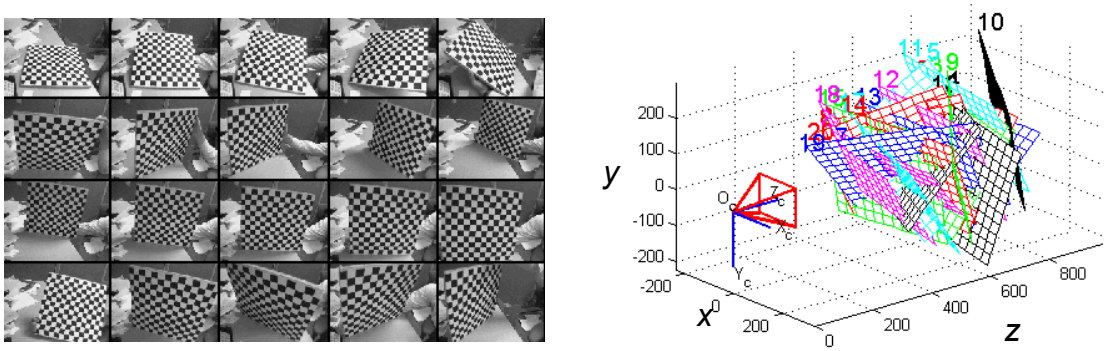
Different approaches to recover the commented parameter were proposed. As straight lines appear curved in distorted images, the plumb line model (BROWN, 1971) uses a scene with a straight line to find the parameters, adjusting them until all the lines in the image became straight as well. There are methods that discard the use of a known object, requiring only to move the camera in a static scene, but these techniques are not indicated when metric accuracy is required (SZELISKI, 2010). Another approach is to use a known

calibration target with known 3-D geometry. The method in (FAUGERAS, 1993 apud ZHANG, 1999) use two or more orthogonal planes with known geometry to calibrate, and the work of Tsai (1987) requires a precisely known translation of a plane. The problem with these kinds of setup is that they, in general, are expensive and require an elaborate setup. As Zhang (1999) method uses a 2D marker (as a chessboard) of unknown 3D position, it overcome the previous methods issues, what made it gain attention from the community as the results are good, and the requirements are simple.

Zhang (1999) methodology requires that a camera observes a planar pattern from at least two different orientations. Then the feature points in the image pattern are detected and used to estimate intrinsic and extrinsic parameters of the camera. Radial distortion parameters are estimated right after and in the last step all parameter are refined.

One drawback is that Zhang technique only alludes radial distortion, Bouguet (1999) overcome this, his method is based on the work of Zhang and Heikkila and Silven (1997), which includes tangential distortion as well.

Appling Bouguet (1999) method allows one to recover the commented parameters and it is possible to undistort the image with steps similar to Zhang's; FIGURE 36 shows an example of possible inputs to the method.



(a) (b)

FIGURE 36 - (a) Input images with chessboard pattern in different orientations. (b) Using input images a 3D plot is created. It shows the relative position of each chessboard with respect to the camera coordinate system (*x*,*y*,*z*).
SOURCE: Bouguet (1999).

After this procedure, it is possible to generate the distortion models, as shown in FIGURE 37 and to undistort images, as shown in FIGURE 38.

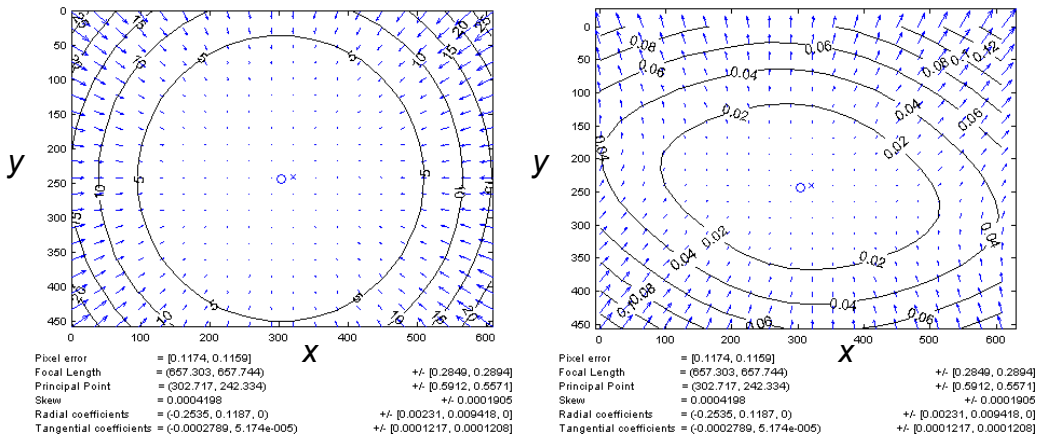


FIGURE 37 - (a) Radial distortion model. (b) Tangential distortion model. Where *x* and *y* are axis coordinates in pixels.
SOURCE: Bouguet (1999).

FIGURE 38 - (a) Image before undistortion and (b) after undistortion.
SOURCE: Bradski and Kaehler (2008).

2.4.5 EPIPOLAR GEOMETRY

Epipolar geometry is the geometry between two views (HARTLEY; ZISSERMAN, 2004), and, therefore, it is the basic geometry used by a stereo vision system (BRADSKI; KAEHLER, 2008). This geometry relates corresponding points in two images of the same scene and is dependent only on the intrinsic parameters of each camera and their relative three-dimensional position (PRINCE, 2012).

Consider two points perfectly aligned in 3D space in relation to the center of projection of a camera observing them. Notice that the projection of both points is exactly the same. This would occur at any point along the same line of projection, as shown in FIGURE 39.

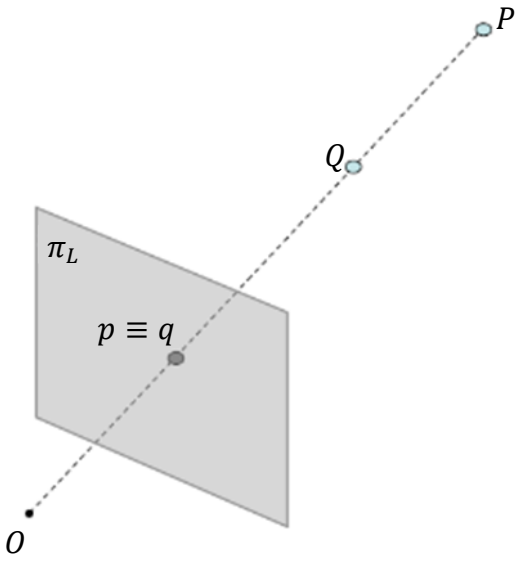


FIGURE 39 - Two points P and Q with the same projection in image plane π_R with center of projection O.
SOURCE: Mattoccia (2012).

Consider now that a new camera is added to the scenario presented in FIGURE 39 (here they are called as left and right view) and that both cameras are able to view the same points P and Q in the 3D world as shown in FIGURE 40.

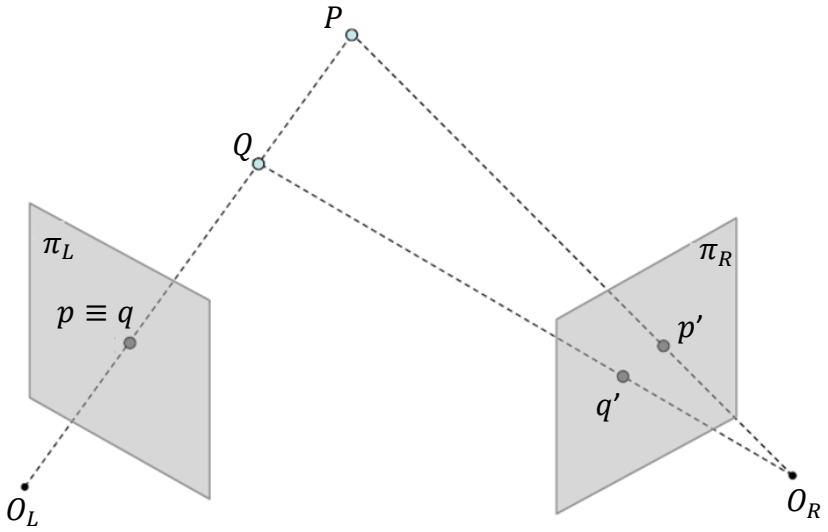


FIGURE 40 - Two points P and Q projected in two image planes π_L and π_R with centers of projection O_L and O_R respectively.
SOURCE: Mattoccia (2012).

In FIGURE 40 it is possible to perceive that to the left view nothing changed compared with FIGURE 39, all points have the same projection, but for the right view, each point has its own projection. See that the projection of all

possible candidates to be the point P in the left view form a line in the right view. This line is known as epipolar line. Another point, let us say Q, could form another epipolar line. Notice that this work the other way around as well, generating what is called the epipolar plane as seen in FIGURE 41 and FIGURE 42.

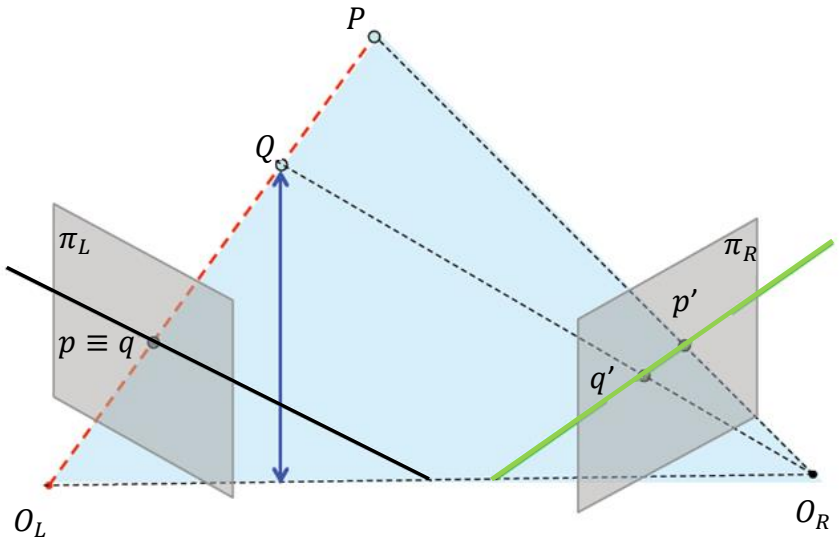


FIGURE 41 - Epipolar plane in blue. Epipolar lines of each view in black and green.
SOURCE: Mattoccia (2012).

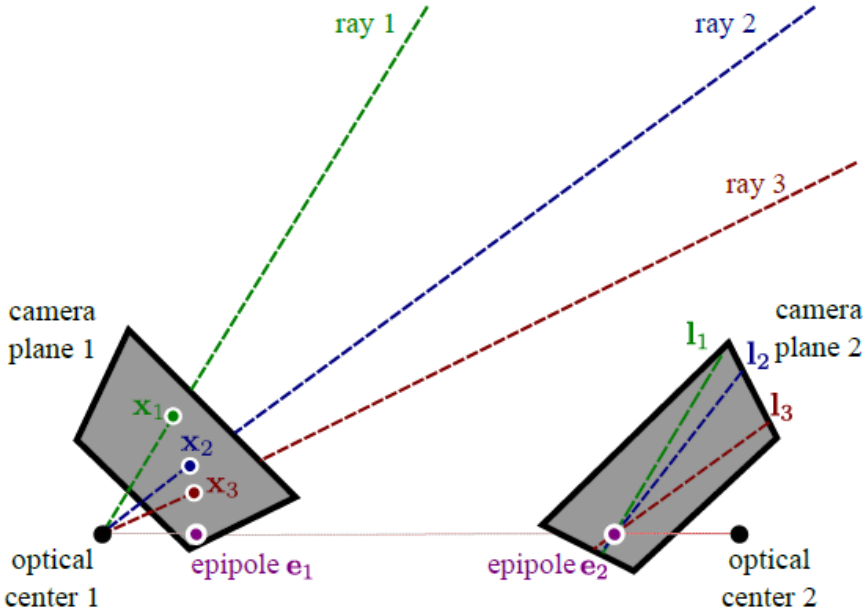


FIGURE 42 - For each ray in one view, there is an epipolar line in the other view.
SOURCE: Prince (2012).

The intersection point of the epipolar lines is called epipole. The ray that passes through the epipole of both cameras connects their centers of projection.

Remember that to recover depth one must be able to know the disparity and to know the disparity it is necessary to identify where a point is projected in

both views. Stereo matching is the name given to the process of finding the projection of a point (matching point) in the other view. This process is simplified with the epipolar geometry, as it shows that one only needs to search for the match of a given pixel in one line of the other image, instead of a whole region, as shown in FIGURE 43.

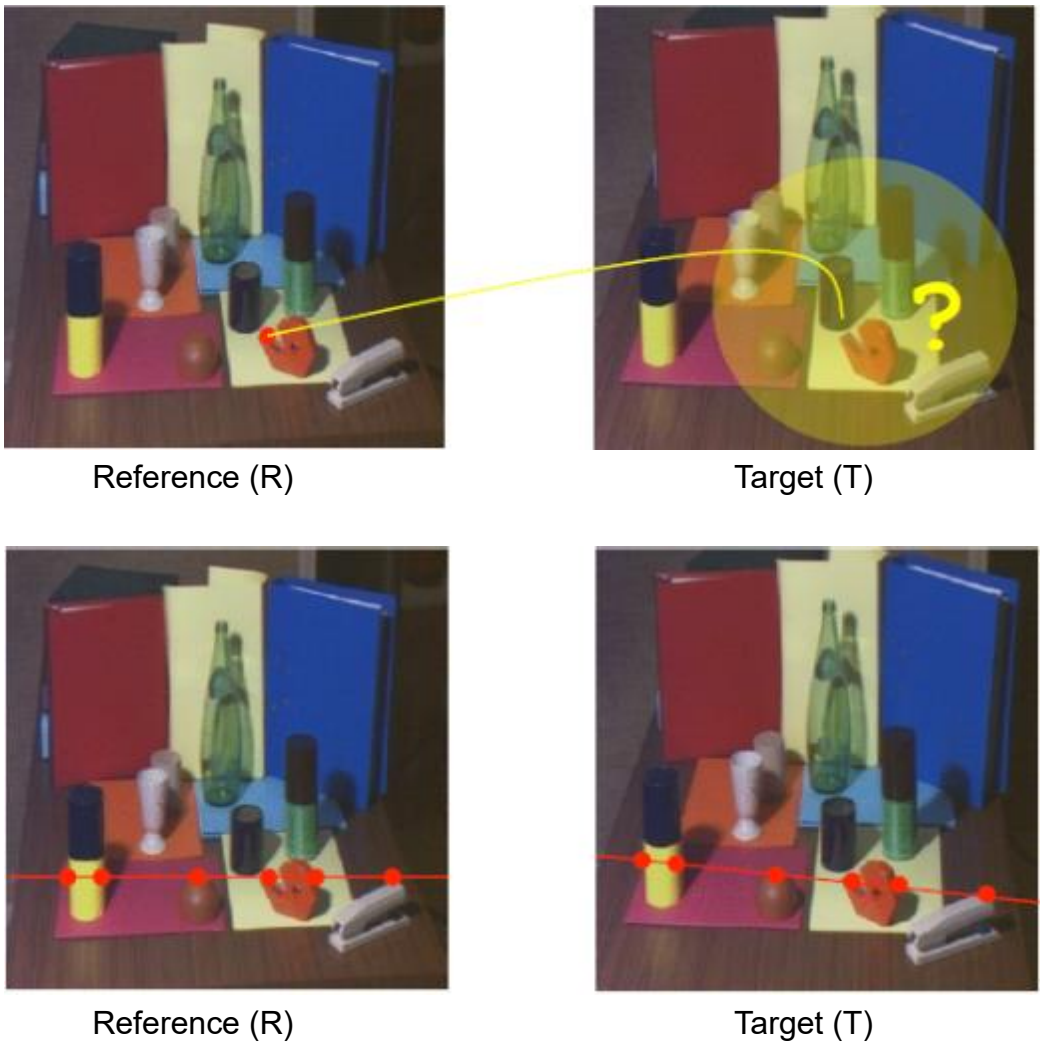


FIGURE 43 - Reduction in search space due the epipolar geometry.
SOURCE: Mattoccia (2012).

Despite the reduction of search space, it is not straightforward to look for pixels using diagonals, happily there is an image transformation called rectification that can ensure that both epipolar lines are aligned, as shown in the next section

2.4.6 RECTIFICATION

With the epipolar geometry described in the previous section, it is possible to reduce the search space for matching pixels. However, note that computers stores images in a rectangular form, with *m* rows by *n* columns. Using the epipolar lines directly may force a search in the image to be performed using diagonals, different from the natural way that pixels are stored using rows and columns. In order to relax this constraint it is possible to rectify the images, that is, corresponding epipolar lines in both images become collinear. In other words, one pixel in the left image in row *m* would be ling in the same row in the right image (if the match exists).

To achieve rectification, the epipoles are shifted to infinity (BRADSKI; KAEHLER, 2008), making the optical axes in both images parallel (CYGANEK; SIEBERT, 2009), as presented in FIGURE 44. But this is not sufficient, as there are infinite possible front parallel planes, it is necessary to add more constraints, as maximizing view overlap and/or minimizing distortion (BRADSKI; KAEHLER, 2008). These constraints vary according to the algorithm applied to obtain the rectification, such as in (BOUGUET, 1999; HARTLEY, 1999; HARTLEY; ZISSERMAN, 2004; LOOP; ZHANG, 1999).

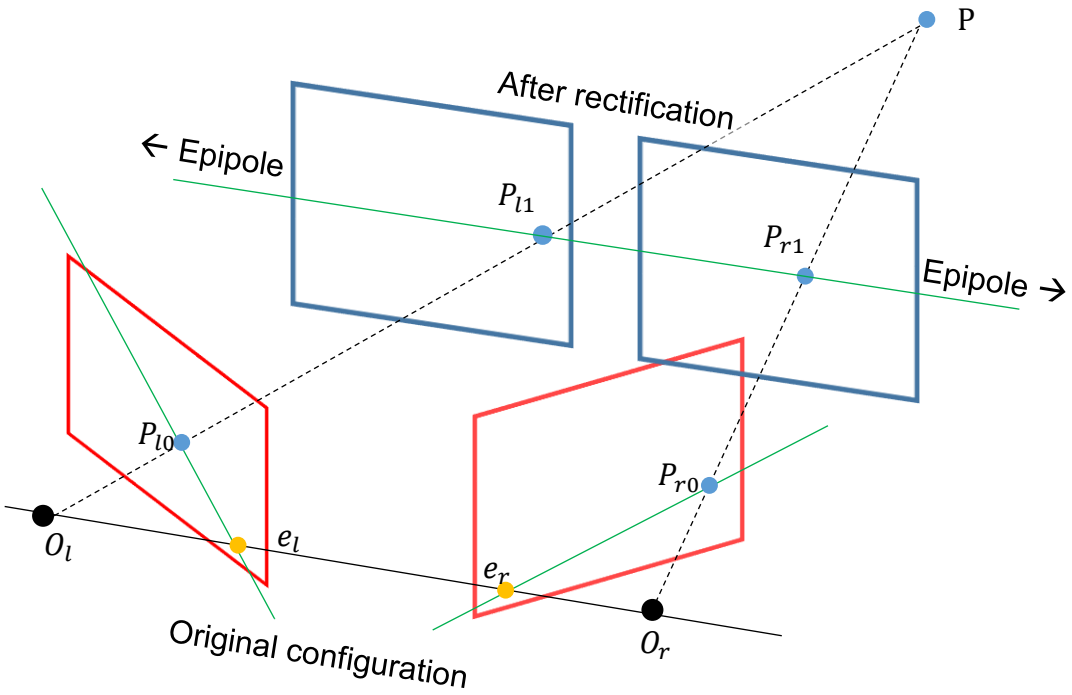


FIGURE 44 - Stereo image rectification. Epipoles from left and right image e_l and e_r in yellow are projected to infinity. The epipolar lines in green are made collinear after the rectification. The red planes represents the original position, while the blue represents the position after rectification. P_{l0},P_{l1},P_{r0}, P_{r1} are projections of the 3D point P over the image planes.
SOURCE: Adapted from Cyganek and Siebert (2009).

2.4.7 STEREO MATCHING

As the name suggests, stereo matching is the process of finding matching pixels in at least two different images of the same scene. Based on the relation found between matching pixels (disparity) and considering that the images are undistorted and rectified, it is possible to calculate the depth using equation (10) (SZELISKI, 2010). The disparity information is generally stored as a map, called disparity map, where higher intensities are associated with closer pixels and lower intensities with the opposite. FIGURE 45 shows an example of a stereo pair and its depth map.

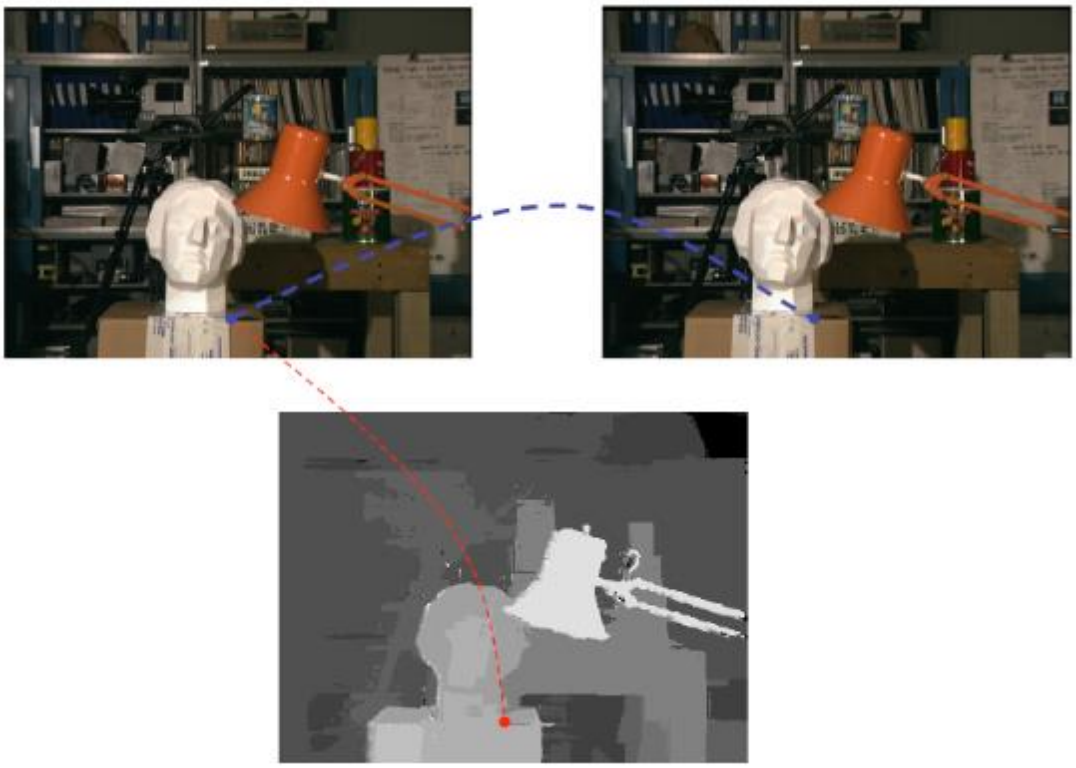


FIGURE 45 - Stereo pair with a line in blue showing matching pixels. The red line shows the disparity value associated with the reference pixel.
SOURCE: Mattoccia (2012).

Finding matching pixels is not a trivial task as the problem is inherently ambiguous (SONKA et al., 2014). Consider the case of a textureless region present in both images, how can one tell with precision what is the right matching pixel? Another problem is with objects that present reflections, this may cause wrong depth measurements as the reflection appears in different places to different viewers. A third problem is the occlusion, when some points are visible only to one camera, what generates pixels without matches. FIGURE 46 presents these common problems.

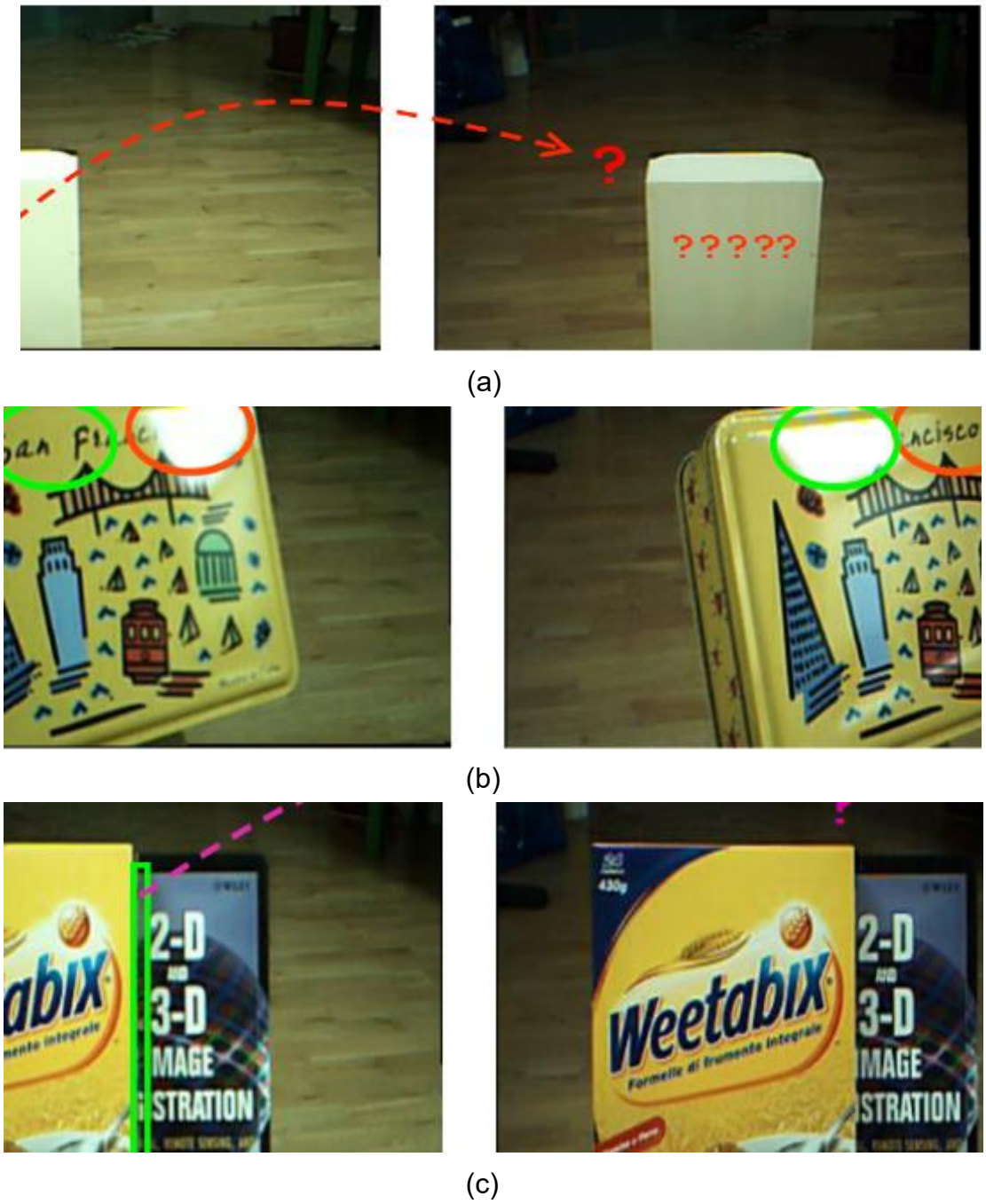


FIGURE 46 - (a) Repetitive pattern. (b) Specular highlight. (c) Occlusion.
SOURCE: Mattoccia (2012).

There are different approaches to treat the commented stereo vision inherent problems; they are briefly described in the next section.

2.4.7.1 Stereo matching algorithms

Considering undistorted rectified images, the simplest algorithm would be to search for the most similar pixel, checking the intensity of pixels in the other image, but in the same line. Different metrics can be used, as absolute difference,

squared difference and so on (HIRSCHMULLER; SCHARSTEIN, 2007). The metric measures what is called matching cost. The compared pixel with the lowest cost is considered the matching pixel; this approach is known as winner-takes-all (WTA). FIGURE 47 shows an example of WTA method being used to find a matching pixel. FIGURE 48 shows the winner selection.

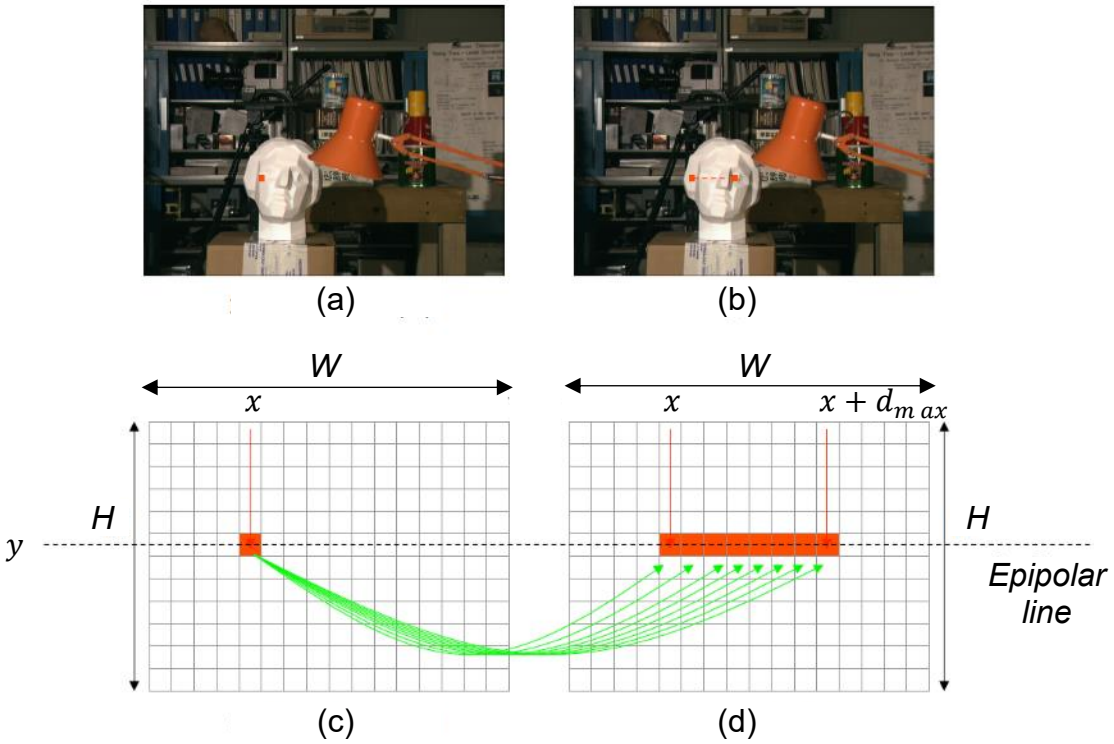


FIGURE 47 - Stereo pair and stereo cost computation. (a) Reference image. (b) Target image. (c) Grid with the pixels of the reference image. (d) Grid with the pixels from the target image. The red pixel in (c) is searched along the red line in (d). Each green arrow indicates a matching cost computation. W and H are the image width and height respectively. (x, y) indicates the position of the red pixel in the reference and d_{max} the maximum disparity used during the search.

SOURCE: Mattoccia (2012).

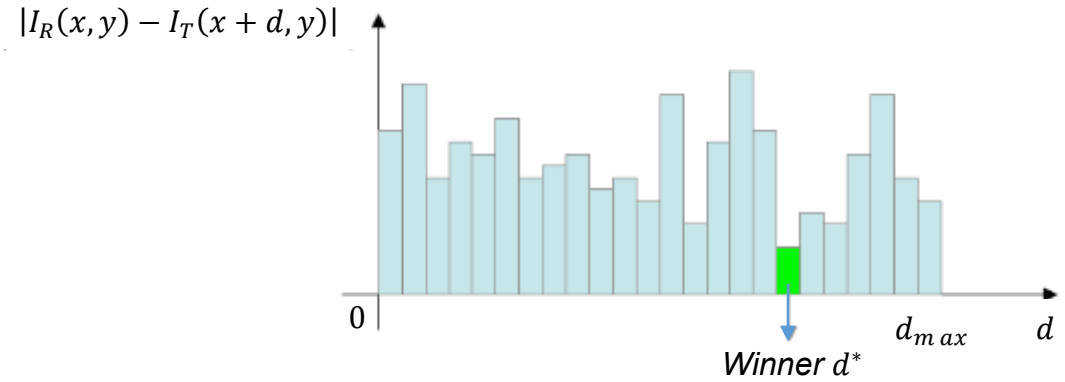
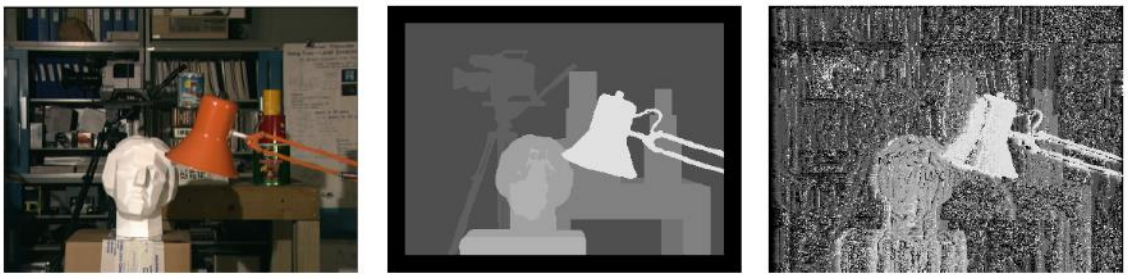


FIGURE 48 – Matching cost value for different disparities d using absolute difference cost. the pixel with disparity d^* in green presents the lowest cost. I_R is the reference image, while I_T is the target image. (x, y) are the horizontal and vertical coordinates in I_R and I_L.
SOURCE: Mattoccia (2012).

As one can perceive in FIGURE 48, the matching pixel is considered the pixel with the lower cost; in the example the metric used is the absolute difference. The value of the found disparity is used to fulfill a disparity map. This process is performed for each pixel in the reference image. Unfortunately, as shown in FIGURE 49, this generates poor results.



(a) (b) (c)

FIGURE 49 – (a) Reference image and the associate (b) ground truth disparity map followed by the (c) disparity map found using single pixel WTA with absolute difference matching cost.
SOURCE: Mattoccia (2012).

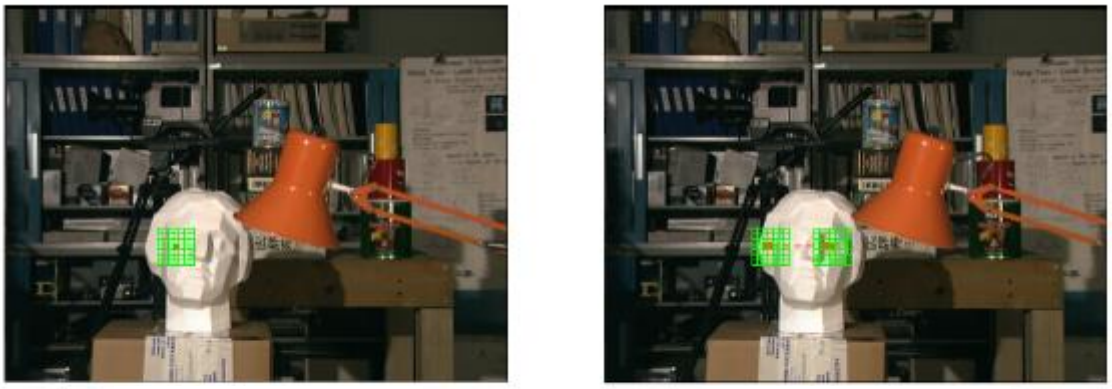
On the contrary, of the presented method, sparse correspondence algorithms produce sparse disparity maps because they work using feature-based and recover depth information only for these points. The feature used varies, but edge detector is commonly used. Examples are given by Grimson (1985) and Ohta and Kanade (1985) works.

Dense correspondence techniques generate complete disparity maps. This problem is harder than sparse correspondence, as it requires that depth values are found in regions where it is hard to find the correct matching pixel.

These algorithms normally perform matching cost computation, support aggregation (neighboring pixels are used as well to compute the cost), disparity computation and a step for refinement (SZELISKI, 2010).

Dense correspondence techniques are divided between local, semi-global and global techniques.

The first example of this section is an example of the local technique. To improve the poor results obtained it is possible to use window-based algorithms. Instead of using a single pixel, they search for the minimum matching cost considering the neighborhood around the pixel. FIGURE 50 show the idea with the window-based approach.



(a) (b)

FIGURE 50 - Windowed stereo matching approach. In (a), the reference image, windows in green and the pixel in red. In (b), the target image, the red dashed line represents the path by which the windows is shifted searching for the correct match.
SOURCE: Mattoccia (2012).

Global algorithms, in general, are modeled as an optimization problem, where a cost function of the whole disparity map is minimized. This cost function normally is formulated as a sum of the matching costs with penalties for not smooth regions in the disparity map (SZELISKI, 2010). The big challenge is to minimize the function, for that different techniques are used, as a graph based (BOYKOV et al., 2001); belief propagation (SUN et al., 2003) and dynamic programming (BLEYER; GELAUTZ, 2008).

3 MATERIALS AND METHODS

During the development of the present dissertation different materials have been adopted, these are presented in the next section followed by the applied method.

3.1 MATERIALS

In this section details about the teleoperated robot specification are not described, only the parts that affect the computer vision design. The robot design is fully described by Siebert et al. (2015). Different software was used in different parts of the method. Open Source Computer Vision (OpenCV) 3.1 library was used to prepare the stereo pairs and Mathworks MATLAB R2015a to detect lines (ITSEEZ, 2016; THE MATHWORKS INC., 2016). The camera and the developed datasets are described in the next sections.

3.1.1 The camera

As already stated the focus of the main project is to build a teleoperated robot to prune vegetation close to overhead energy lines. Considering this fact, during its design were taken into account settings as the field of view (FOV), to allow the operator to work properly, the size of the cameras, as they must fit in the robot arm, and so on. Over all this factors, the only decision that can be taken here is the camera positioning, detailed in the next subsection.

3.1.1.1 Choosing the positioning

Considering that the cameras should be placed in the robot arm close to the pruning tool and considering as well that the system will need at least two cameras side by side in order to use stereo vision, it is clear that the maximum width of a camera must be the width of the robot arm divided by two. One could argue that it is possible to add a platform or any other type of support to the cameras, but again it is necessary to remember that the cameras must not be an obstacle during the robot movement. As the robot arm region where the cameras will be fixed measures 178mm x 175mm, the camera size limits are 178mm x 87.5mm. FIGURE 51 shows the region where the cameras will be placed.

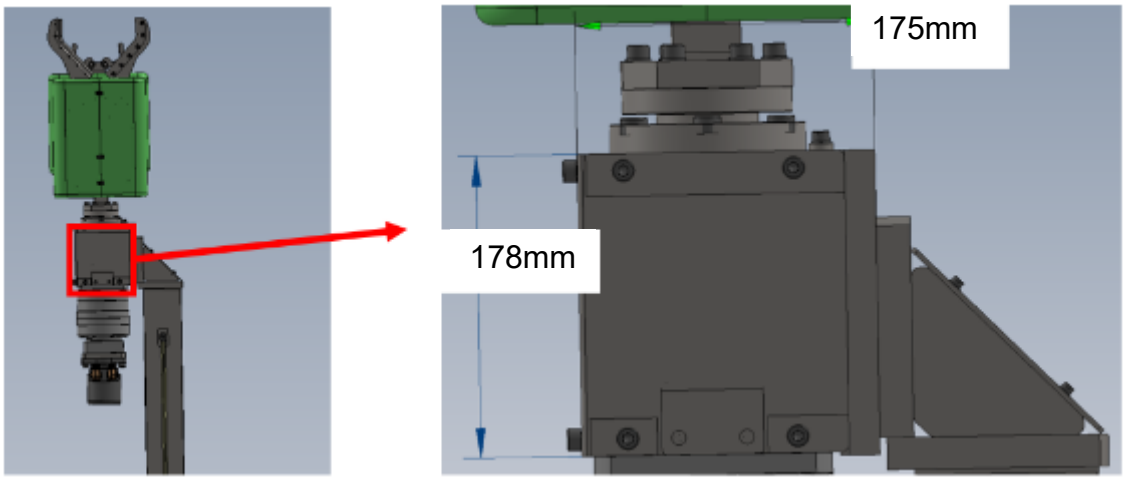


FIGURE 51 - Robot arm and zoom in the region where the cameras will be installed.
SOURCE: The author (2016).

The camera used in the robot is an AXIS 1005-E. TABLE 1 shows the camera specification and FIGURE 52 the camera.

TABLE 1 - Axis 1005-e specification

Product	AXIS 1005-E
Sensor	1/2.8”
Focal length	2.8mm
Aperture	F2.0
Resolution	1080p
IP rating	IP67
Operating temperature	-30°C ~ 55°C
Dimensions (L x A x P)	30mm X 30mm X 62mm
Weight	116g
FOV	113° X 62°

FIGURE 52 - AXIS 1005-e.

Before going further with camera positioning, it is necessary to evaluate the impact of this decision over the stereo system. To do so, first recall the equation for depth perception:

$$Z = \frac{Bf}{d} \tag{14}$$

where Z is the depth, B the baseline, f the focal length, and d the disparity.

One can perceive that the depth is directly proportional to the baseline and focal length and inversely proportional to the disparity. From the available parameters it is only possible to control the baseline. From FIGURE 51 and TABLE 1 it is noticeable that the camera width is 30mm and the robot arm width is 175mm. Therefore, the minimum baseline is 30mm and the maximum to keep the cameras over the arm is 145mm. FIGURE 53 helps to understand how the baseline range was defined.

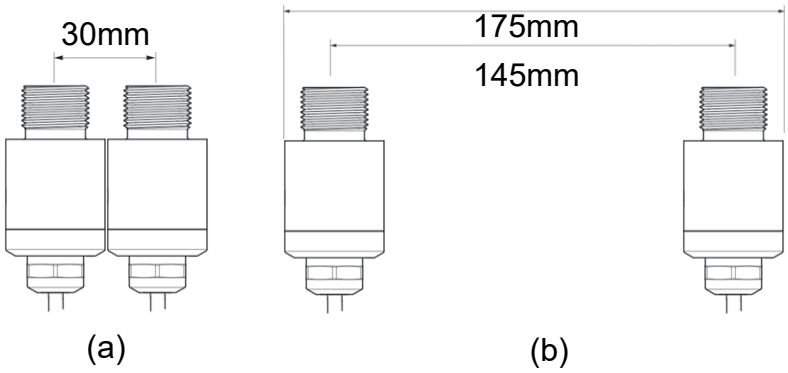


FIGURE 53 - (a) Minimum baseline. (b) Maximum baseline.
SOURCE: The author (2016).

To check the impact of the baseline in the depth resolution it is necessary to introduce another equation that is recovered from the geometry presented in FIGURE 54 and FIGURE 55. FIGURE 54 helps to explain why the depth resolution decreases for objects afar.

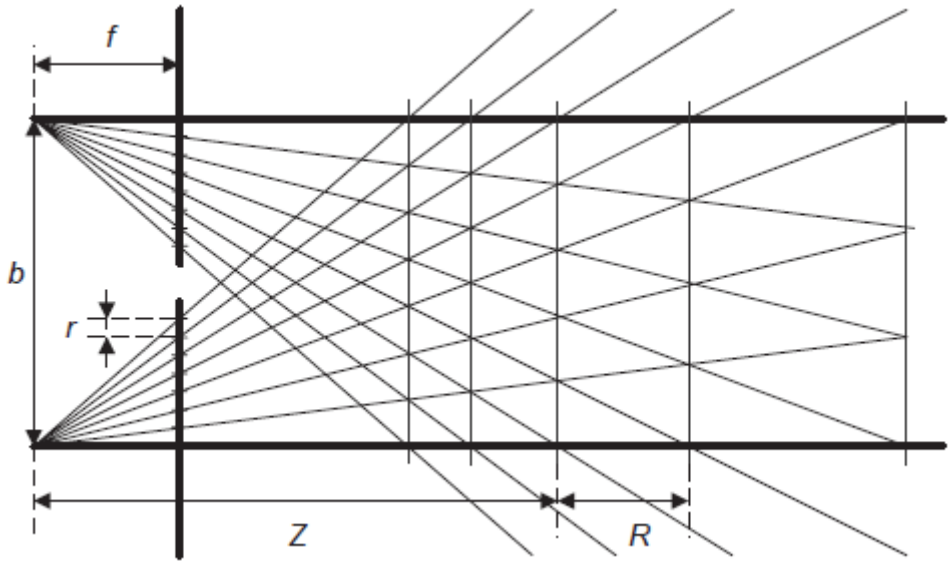


FIGURE 54 - Depth resolution in binocular stereo view. Where b is the baseline, f is the focal length, r is the pixel resolution and, Z the depth and R the depth resolution.
SOURCE: Cyganek and Siebert (2009).

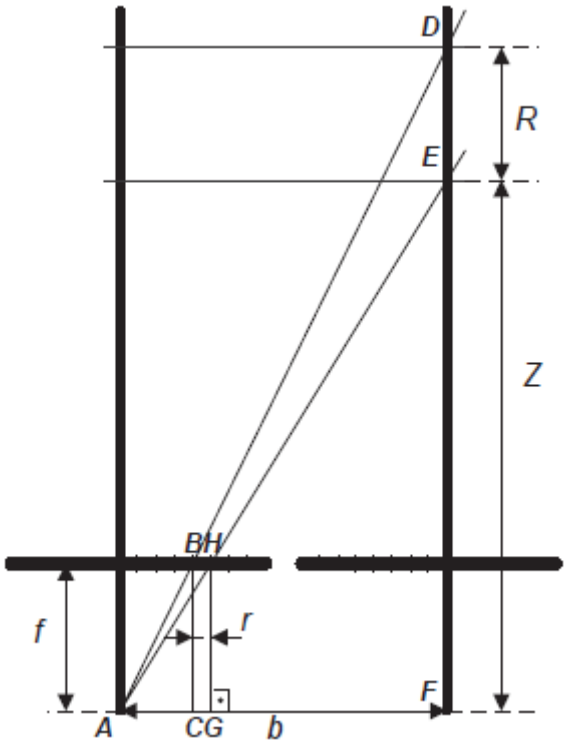


FIGURE 55 - Depth resolution related to pixel resolution.
SOURCE: Cyganek and Siebert (2009).

Using FIGURE 55 one can find what affects depth resolution (CYGANEK, SIEBERT, 2009). Consider the similar triangles ΔADF and ΔAEF. From them, one can obtain the relations:

$$\frac{\overline{DF}}{\overline{AF}} = \frac{\overline{BC}}{\overline{AC}} \qquad \frac{\overline{EF}}{\overline{AF}} = \frac{\overline{GH}}{\overline{AG}} \tag{15}$$

From the FIGURE 55 and using the presented relations, the pixel resolution r is equal to:

$$r = \overline{AG} - \overline{AC} = \overline{GH}\frac{\overline{AF}}{\overline{EF}} - \overline{BC}\frac{\overline{AF}}{\overline{DF}} = \frac{fb}{Z} - \frac{fb}{Z+R} \tag{16}$$

then,

$$r = \frac{fb(Z+R) - fbZ}{Z(Z+R)} = \frac{fbR}{Z(Z+R)} \tag{17}$$

and then,

$$Rfb = rZ(Z+R) \Rightarrow Rfb - RrZ = rZ^2 \tag{18}$$

finally,

$$R = \frac{rZ^2}{fb - rZ} \tag{19}$$

Using the obtained formula the Depth Resolution is plot within the baseline b range $[30mm \sim 145mm]$ as shown in FIGURE 56

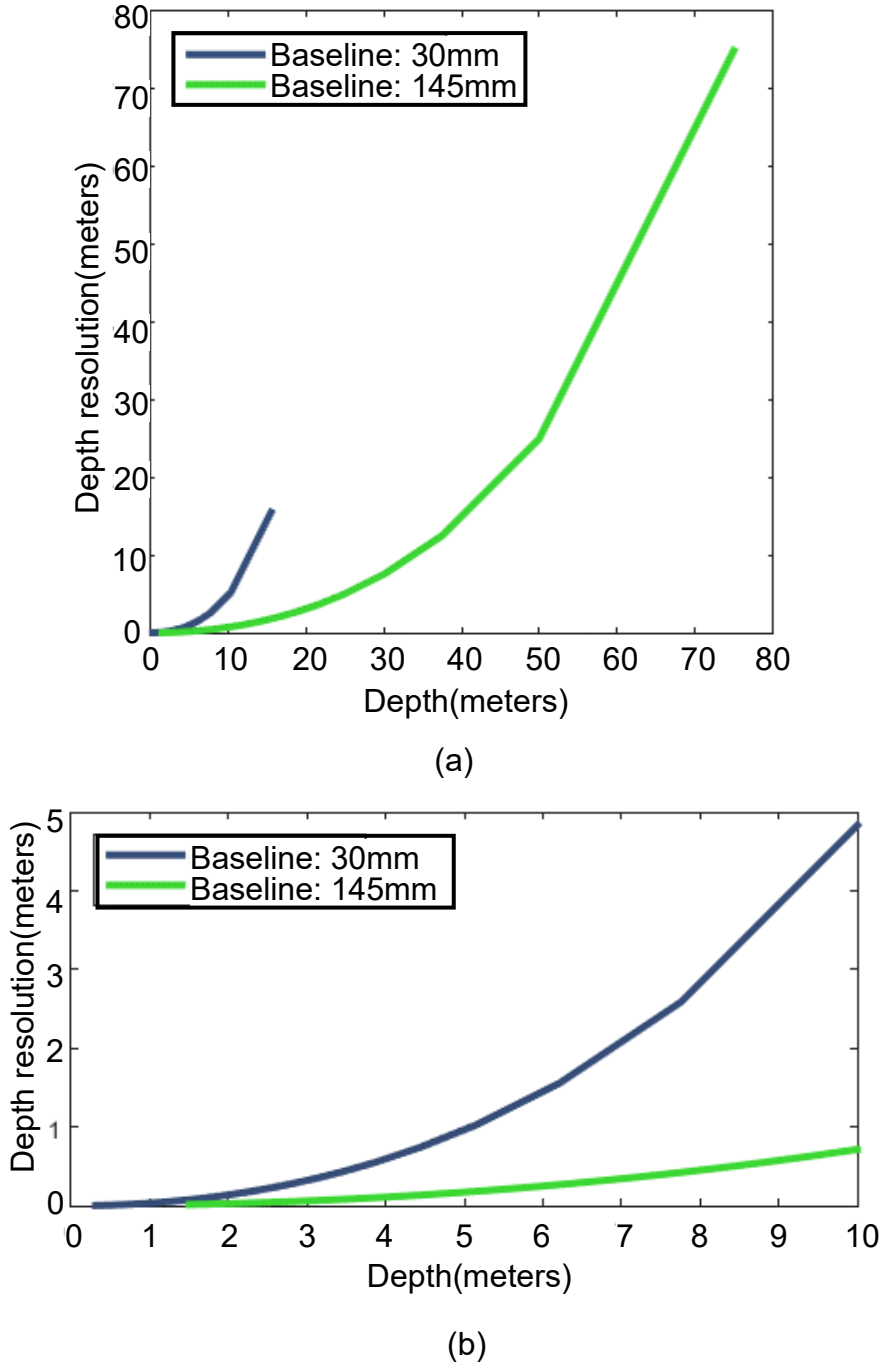


FIGURE 56 - Depth resolution compared with depth for two different base lines. (a) For the whole interval. (b) Zoom in the interval from 0 to 10 meters.
SOURCE: The author (2016).

As the robot will work close to energy lines and the goal is to measure the distance from the robot to them, it is necessary to specify the baseline in a way that ensures the necessary accuracy measuring mainly closer objects as collision avoidance is desired. Despite the baseline is capable to increase the

maximum range, it is important to remember that a wider baseline increases the occlusion problem already shown in FIGURE 46 (c). On the other hand, a shorter baseline decrease the occlusion problem and decrease the maximum range in exchange for an increase in resolution for closer ranges. As the focus of the project is close range measurements, the shorter baseline of 30mm was used during the dataset creation described in the next subsection.

3.1.2 Datasets

As the pruning robot will be used in real conditions, it is necessary to evaluate the computer vision algorithm in real scenes. Unfortunately, the Axis 1005-E cameras could not be bought since the beginning of this dissertation, as the requirements, dimensions and others characteristics of the robot were not defined in that time. Therefore, in order to allow the beginning of the studies of the line detection algorithm, a GoPro camera was bought and was used to record a video while attached to a lineman’s safety helmet during the pruning activity in Salvador, Brazil. FIGURE 57 shows one frame of the recorded footage.

FIGURE 57 - One frame from the recorded footage using a GoPro attached to the helmet of a lineman during pruning activity close to power lines.
SOURCE: The author (2016).

During the period where two cameras were not available, the first tests of stereo algorithms were performed using the dataset provided by Scharstein and Szeliski (2002). FIGURE 58 shows examples of image pairs from this dataset.

(a) (b) (c)

FIGURE 58 - (a) Left view. (b) Right view. (c) Ground truth disparity map (left view used as reference).
SOURCE: Scharstein and Szeliski (2002).

Finally, when the Axis 1005-E became available, a new footage was recorded using two cameras attached to a hot stick recording energy lines at

Institutos Lactec, in Curitiba, Paraná, Brazil. The device used to attach the cameras to the hot stick is shown in FIGURE 59

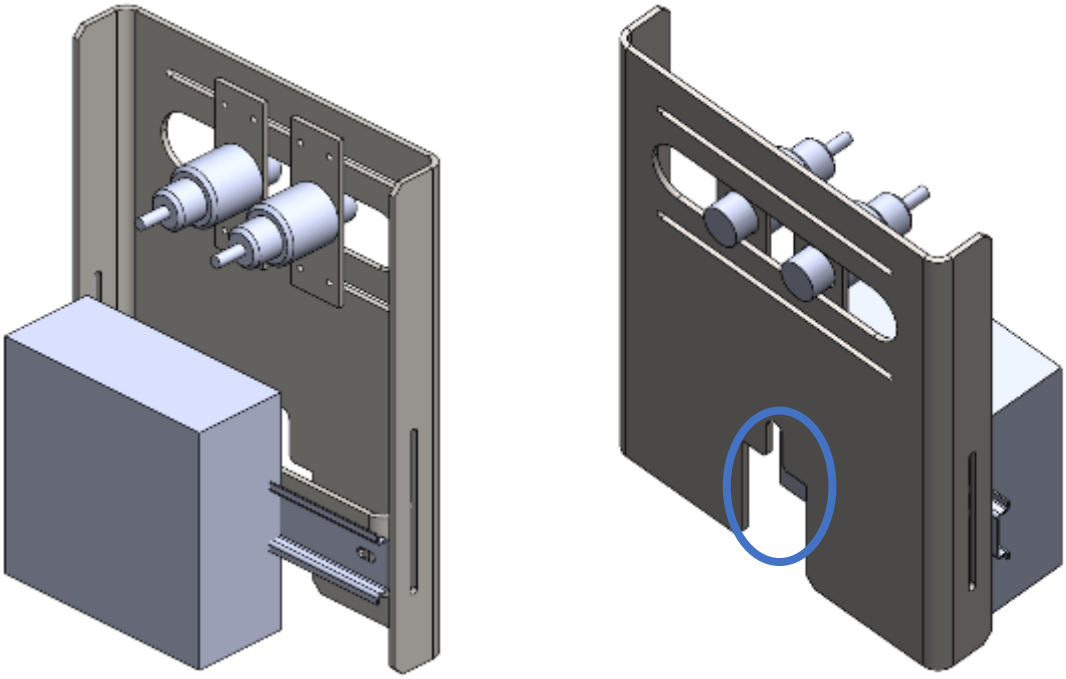


FIGURE 59 - two views of the 3D model of the device used to attach the camera to the hot stick. Inside the blue cycle is the part where the hot stick is linked to.
SOURCE: The author (2016).

A special scenario was mounted to create the dataset. Two energy poles were positioned with a tree between them. Three energy lines were installed, but they are not energized for safety reasons. FIGURE 60 shows the location and FIGURE 61 the system built and mounted to the camera.

FIGURE 60 - The site used to build the dataset.
SOURCE: The author (2016).

FIGURE 61 - Axis camera mounter in stereo setup.
SOURCE: The author (2016).

To perform the recordings the system is powered by 8 x 1.2V batteries in series connected to the unit that manage the cameras. The footage is recorded to a memory card and later uploaded to a computer. Seven stereo pairs extracted from the footage are shown in FIGURE 62.

(a) (b)

FIGURE 62 - Seven stereo pairs. (a) Left view and (b) right view frames from the footage recorded in Lactec site in Curitiba.
SOURCE: The author (2016).

During tests, a chronometer with milliseconds resolution was recorded using both cameras. The recordings put in evidence that the cameras are not

perfectly synchronized and even when the first frame of the streams are manually synchronized it loses synchronization along the footage. Further, it was found in the camera's knowing issues in the camera manufacturer website that it presents a bug that makes it skips frames along the recordings. As frame synchronization is a requirement for stereo vision, to overcome the bug and to allow tests of the proposed method, a software was developed to forward and backward each recording individually frame by frame while undistorting and rectifying each of them. This tool allowed to extract synchronized frames from recorded streams by checking the alignment of each pair, but manually. This issue reduced drastically the amount of frames available for testing and precluded the use of video footages in the method phase. A total of 25 synchronized pairs were extracted from footages recorded along two days and will be used to test the algorithm.

3.2 METHOD

The developed method consists of seven steps to detect energy lines: preprocessing, where the images are prepared for the next steps; feature extraction, where edges with specific geometric features are extracted; 2D energy line detection, that gets the most energy lines candidates as possible; depth estimation, which obtains 3D information of the candidates; 3D power line filtering, that filters the energy line candidates using depth information; merging energy lines, that merges energy line segments; and energy line growing that extend energy lines. FIGURE 63 shows the method steps.

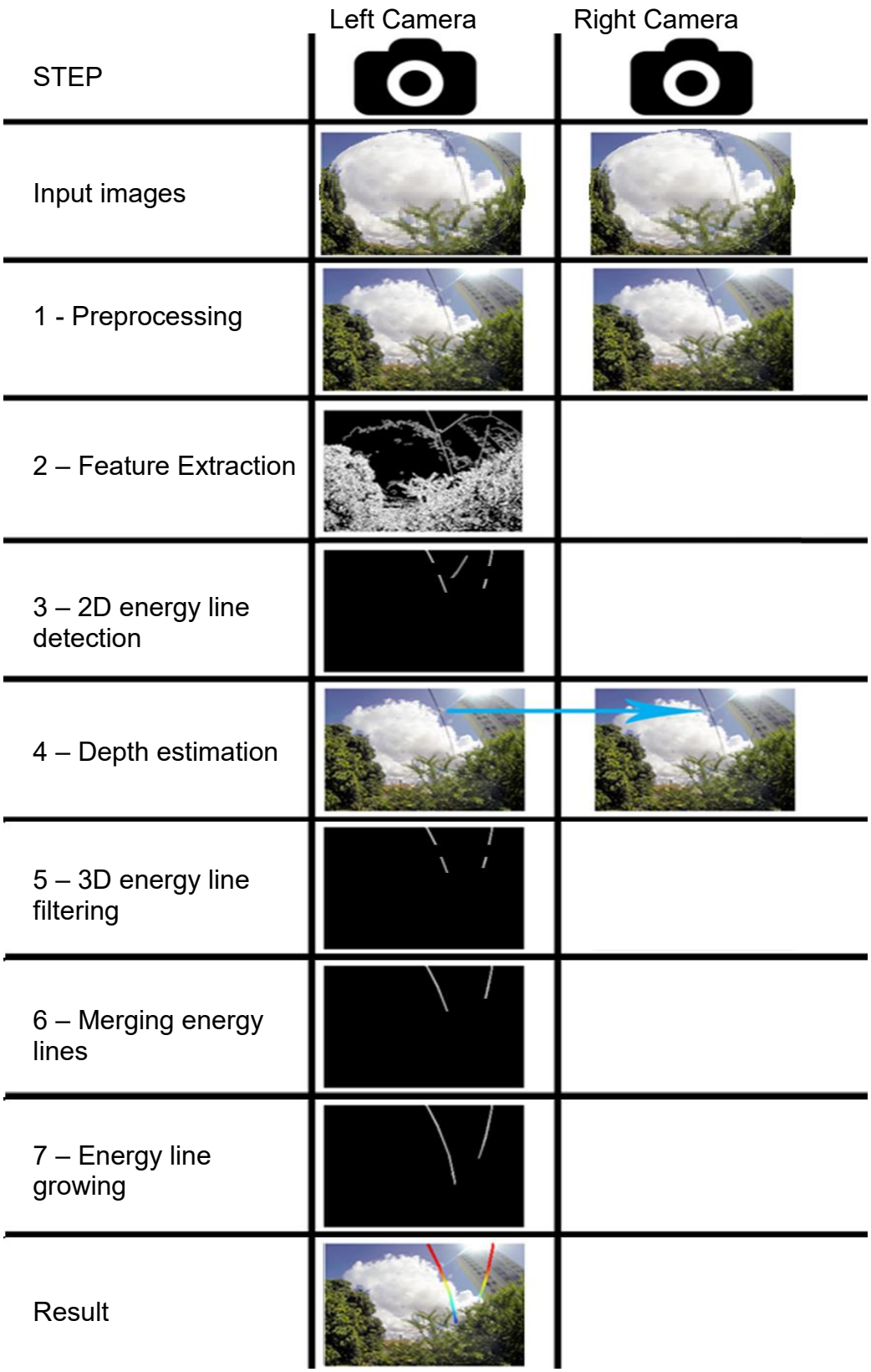


FIGURE 63 - Steps of the proposed method.
SOURCE: The author (2016).

3.2.1 Pre-processing

During this phase, images obtained by the cameras are adjusted in order to become valid inputs to the algorithms applied in the next steps. Taking this into account, rectified undistorted image pair, color transformation and intensity images are created.

To obtain the preprocessed image pairs, it is necessary to perform the steps shown in FIGURE 64.

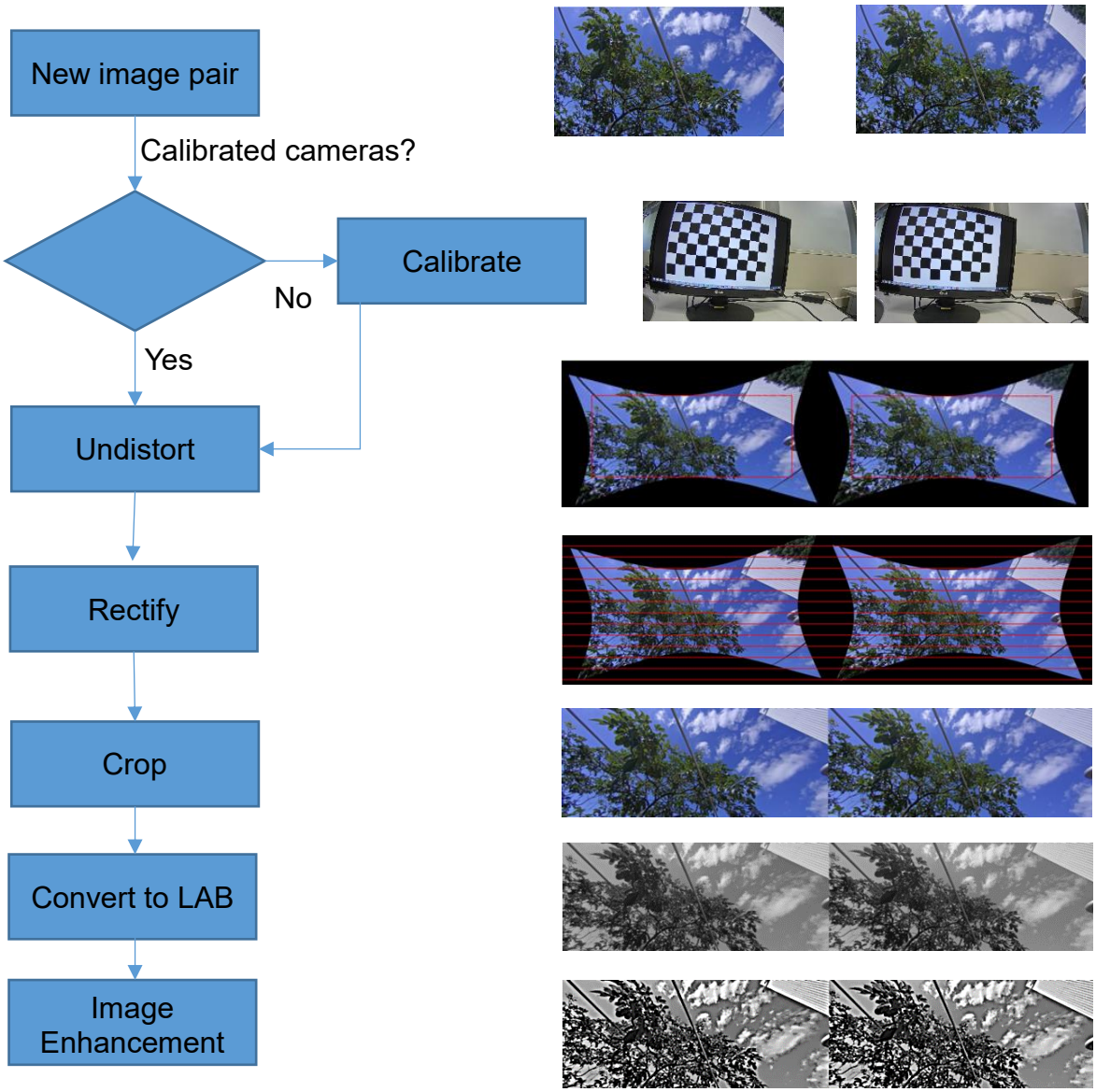


FIGURE 64 – Pre-processing steps.
SOURCE: The author (2016).

One can perceive that it is not necessary to repeat the calibration step as long as the stereo rig parameters (baseline, camera vergence, etc.) are not

modified. Therefore, if the stereo rig is fixed, the other steps may be repeated freely for every new stereo pair. The steps are detailed in the next subsections.

3.2.1.1 Single Camera Calibration

The Calibration step seeks to relate real world measurements with camera measurements (*e.g.* centimeters with pixels) and to correct deviations while applying the pinhole camera model in the present system.

As already commented, different methods may be used to perform the calibration process. Here, the OpenCV implementation of BOUGUET (1999) method is used. The robustness and the numerous available functions of the OpenCV library may make the calibration procedure look simple, what in fact it is not. Therefore, seeking to give a better understanding of what is performed here, it is necessary to detail the functions used to enlighten the underlying implementation. Figure 65 gives an overview of the whole calibration process including the stereo part.

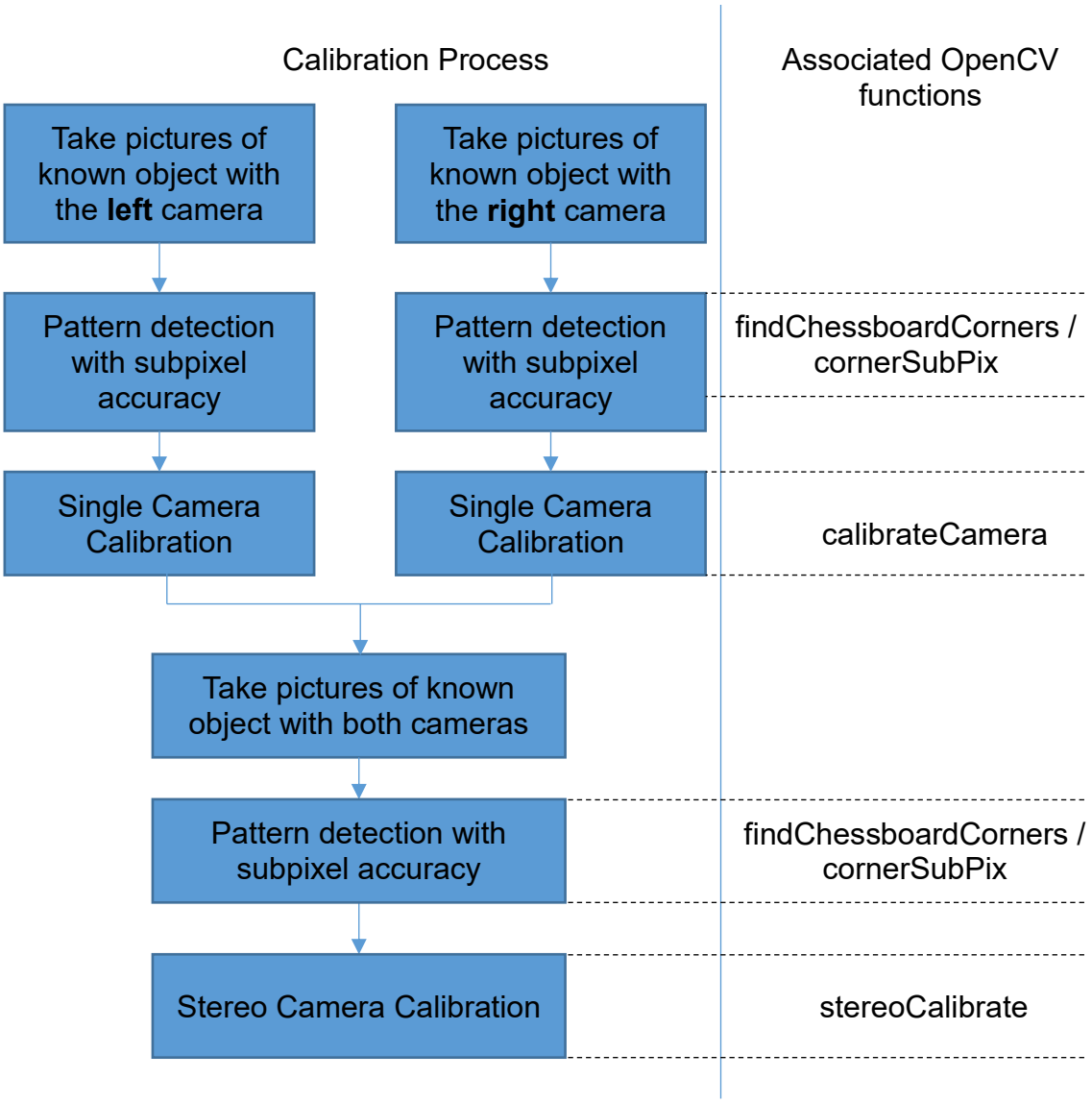


Figure 65- Calibration procedure with OpenCV
Source: The author(2016)

In short, the method uses an object with known dimensions (3D points) and its projection into multiple pictures (2D points) from different viewpoints to find the camera parameters; distortion coefficients; and, as two cameras are used in the present work, the matrices that correlates the views of each camera.

OpenCV allows the use of some predefined calibration patterns. Among them, the calibration object used here is a chessboard pattern with 7 rows and 10 columns, with each square side measuring 3.4mm. The pattern was printed in A4 paper and then it was fixed to a monitor with pivot standing, which allows the pattern orientation to be modified freely and ensured that it would be steady during the image capture. Two sets, each one containing 20 pictures of the

chessboard in different positions, were created using each camera as shown in FIGURE 66.

(a) (b)

FIGURE 66 - (a) Set of pictures shot with the left camera. (b) Set of picture shot with the right camera.
SOURCE: The author (2016).

The points used in the chessboard pattern are the intersection of the black and white squares. For each image, a specific intersection is considered to be origin point (see FIGURE 67), with coordinates X, Y and Z equal to zero. The chessboard pattern in each picture is considered coplanar with Z plane, consequently, all points present Z coordinate equal zero. Therefore, the list of 3D points is composed by points with coordinates $(X, Y, Z) = (3.4k, 3.4l, 0)$ where $k \in \mathbb{N} \mid 0 \leq k \leq 8$ and $l \in \mathbb{N} \mid 0 \leq l \leq 5$.

Now, it is necessary to identify the 2D coordinates (pixels positions) of the intersections in each picture of the chessboard pattern. This is performed using the function *findChessboardCorners()* that finds each chessboard corner automatically as shown in FIGURE 67.

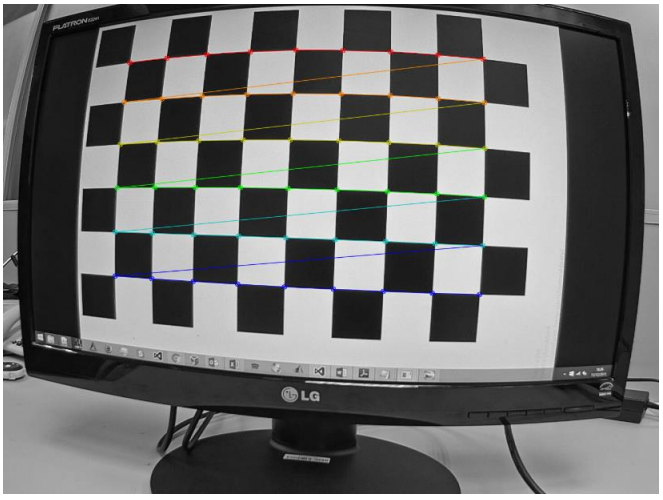


FIGURE 67 – Corners of the chessboard pattern found automatically. Each row of corners is marked with a different color. Where the (0,0,0) is the leftmost topmost red point.
SOURCE: The author (2016).

Using directly the pixel positions may cause undesired errors during the calibration, as the pixels position are limited to the minimum camera sensor measuring capability (1 pixel). Fortunately, OpenCV owns a function called *cornerSubPix()* that allows to use the known properties of the chessboard pattern in order to reach subpixel localization, which improves the results of the whole calibration process.

With the patterns points found (2D) and the list of 3D points it is possible to start the calibration process. This is performed using the function *calibrateCamera()* one time for each set of images obtained with each camera. The function returns the distortion coefficients and the matrices and vectors that relate the 3D points (chessboard coordinates) and its 2D projection (image coordinates).

Among the returned matrices/vectors, there are the rotations matrices and translation vectors called R and T respectively. The R matrices are used to rotate and the T vectors are used to translate 3D points in chessboard coordinates system (unique to each picture) to the 3D camera coordinate system (common to all pictures). Therefore, the matrices R and T are different for each image of the chessboard. Another matrix returned is the camera matrix K, which contains information about the camera geometry, as focal length and center of projection position in pixels, that allows one to project 3D points in 3D camera coordinates to 2D image coordinates. This transformation is given by:

$$s\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{20}$$

Or in a more compact notation,

$$s[p] = [K][R|T][P] \tag{21}$$

where s is a scaling factor, K the camera matrix, R the rotation matrix, T the translation vector, p the projection points in pixels (image system) and P the world points (chessboard coordinate system), where p and P are in homogeneous coordinates.

The calibration function start by solving equation 21 using all the p and P and ends finding K, R and T, but notice that until now the lens distortions are not taken into account. Using the already presented equation 12 and 13 it is possible to get a single equation considering both tangential and radial distortion:

$$\begin{bmatrix} x_p \\ y_p \end{bmatrix} = (1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \begin{bmatrix} x_d \\ y_d \end{bmatrix} + \begin{bmatrix} 2p_1 x_d y_d + p_2(r^2 + 2x_d^2) \\ 2p_2 x_d\, y_d + p_1(r^2 + 2{y_d}^2) \end{bmatrix} \qquad (22)$$

where x_p and y_p are the points coordinates without distortion and x_d and y_d are the opposite. In fact, the model presented in equation 22 might not be able to handle great distortion, mainly introduced by cameras with wide field of view. To bypass this issue, OpenCV allows the use of the flag CALIB_RATIONAL_MODEL that introduces additional radial distortion parameters, called rational model and presented in equation 23.

$$\begin{bmatrix} x_p \\ y_p \end{bmatrix} = \frac{(1 + k_1 r^2 + k_2 r^4 + k_3 r^6)}{(1 + k_4 r^2 + k_5 r^4 + k_6 r^6)} \begin{bmatrix} x_d \\ y_d \end{bmatrix} + \begin{bmatrix} 2p_1 x_d y_d + p_2(r^2 + 2x_d^2) \\ 2p_2 x_d\, y_d + p_1(r^2 + 2{y_d}^2) \end{bmatrix} \qquad (23)$$

Using the initial estimation of the matrix K the points $[x_p, y_p]^T$ are obtained from

$$\begin{bmatrix} x_p \\ y_p \end{bmatrix} = \begin{bmatrix} \frac{f_x X}{Z} + c_x \\ \frac{f_y Y}{Z} + c_y \end{bmatrix} \qquad (24)$$

Using the points obtained with equation 24 as the undistorted points and the points original coordinates p, as the distorted points, the distortion coefficients are estimated using the rational model and finally all these parameters are refined using the global Levenberg-Marquardt optimization algorithm.

Besides the distortion coefficients and the matrices, the function *calibrateCamera()* returns the root mean square of the reprojection error as well. That is, it sum the squared distances between the real corner location (position

in the image) and the reprojected location (position estimated projecting the 3D points using the parameters found). The reprojection error is used to measure the quality of the obtained calibration.

3.2.1.2 Stereo camera calibration

During the previous section only the parameters to undistort the image and the geometry of each camera were obtained. However, as a stereo setup is used here, it is necessary to find the geometrical relation between both cameras as well. In OpenCV, this is performed using *stereoCalibrate()* function. To do so, another set of pictures of the chessboard in different orientations is taken from both views, but now at the same time. This results in pairs of images of the chessboard in the same position but observed from a different viewpoint as shown in FIGURE 68

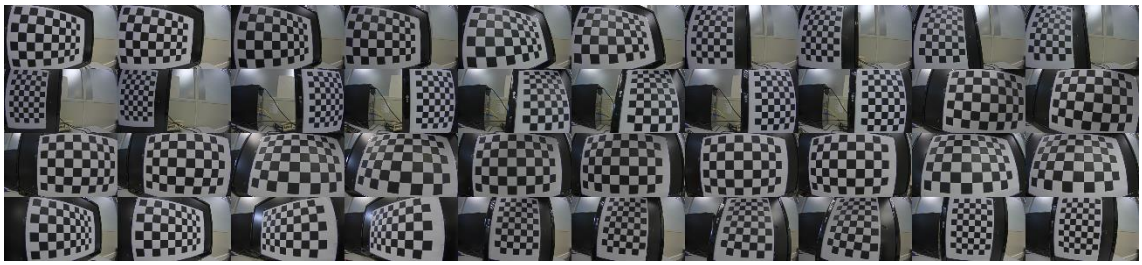


FIGURE 68 – 20 image pairs used to calibrate the cameras. Left view and right view are shown sequentially.
SOURCE: The author (2016).

Then, as in the single camera calibration case, the 2D and 3D coordinates of the intersections are extracted and refined. The camera calibration K and the distortion coefficients of each camera can be found with a single call to *stereoCalibrate()*. However, this makes OpenCV optimize the parameters of each camera and the stereo parameters in a single step. To avoid this, the function *stereoCalibrate()* is used taking into account the parameters already found by calibrating each camera individually using *calibrateCamera(),* which makes it calculate only the stereo parameters, as recommended by Brahmbhatt (2013).

From the stereo calibration the useful outputs not calculated during the single camera calibration phase that will be used afterwards are the $R_{r\to l}$ and $T_{r\to l}$ matrices. These matrices are used to relate one camera view with the other.

Choosing the left camera as the main coordinate system the relation between the cameras coordinates system is described as:

$$P_l = R_{r\to l}^{T}(P_r - T_{r\to l}) \tag{25}$$

where P_l and P_r are 3D points using the left and right camera as reference respectively. Notice that $R_{r\to l}$ and $T_{r\to l}$ are the same for every pair of images as the cameras position are fixed. With the calculated information during this step, now it is possible to undistort and rectify the images as explained in the next section.

3.2.1.3 Undistortion and rectification

With the known geometry of both cameras, the geometry relating them and the distortion parameters now it is possible to undistort and rectify the images. In OpenCV, this can be performed using *StereoRectify*(). This algorithm seeks to maximize the overlap area between both views while minimizing the change caused by the reprojection. To minimize the reprojection instead of rotating one view while letting the other fixed, the algorithm distributes the amount of rotation between both views. This makes both image planes coplanar but not rectified, what is accomplished by projecting the epipoles of each view to the infinity. At the end of the process the function *StereoRectify()* returns R_{rectL} and R_{rectR} rotation, projection matrices P_l and P_r used to project a world 3D point to a 2D image point in left and right image respectively. The projection matrices are given by:

$$P_l = \begin{bmatrix} f & 0 & c_{xl} & 0 \\ 0 & f & c_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$
$$P_r = \begin{bmatrix} f & 0 & c_{xr} & Bf \\ 0 & f & c_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{26}$$

On the other hand, the reprojection matrix Q is used to project a 2D image point with known disparity to the 3D world.

$$Q = \begin{bmatrix} 1 & 0 & 0 & -c_{xl} \\ 0 & 1 & 0 & -c_y \\ 0 & 0 & 0 & f \\ 0 & 0 & -\frac{1}{B} & \frac{(c_{xl} - c_{xr})}{B} \end{bmatrix} \quad (27)$$

The *StereoRectify()* function accepts flags, here the flag CALIB_ZERO_DISPARITY is used, which ensures that the center of projection of each camera have the same coordinates after the rectification step. Allowing the disparity to be calculated directly, without the need to take into account c_{xl} and c_{xr}

With the additional parameters obtained from the *StereoRectify()* function, any given pair of images obtained by the stereo rig used in this work can be undistorted and rectified. This can be achieved by using the OpenCV function *initUndistortRectifyMap()*. This function calculates a lookup table used to rectify and undistort new image pairs based on the parameters obtained during the previous steps: the camera matrix K; the distortion coefficients; the rotation matrix R_{rect} and the projection matrix P, returning as output maps to convert new images from each camera view.

Finally, with the obtained maps any given image pair is converted to an undistorted rectified image in a single call to *remap()* function that, as the name says, remaps a set of points to new coordinates.

3.2.1.4 Image crop

Due to the lack of linearity during undistortion and rectification some pixels are projected outside the image limits, this makes the image loose its rectangular shape. Therefore, to simplify future steps of the algorithm it is necessary to discard these pixels by recovering an area with rectangular shape. Fortunately, the already used function *stereoRectify()* returns as well the coordinates of a rectangle with only valid pixels inside for each view, this region is called region of interest (ROI).

With the ROI of each image, the common ROI is calculated as their intersection area and then the image is cropped using the ROI as reference.

Figure 69 shows an example of the rectified images and their original ROIs and the common ROI.

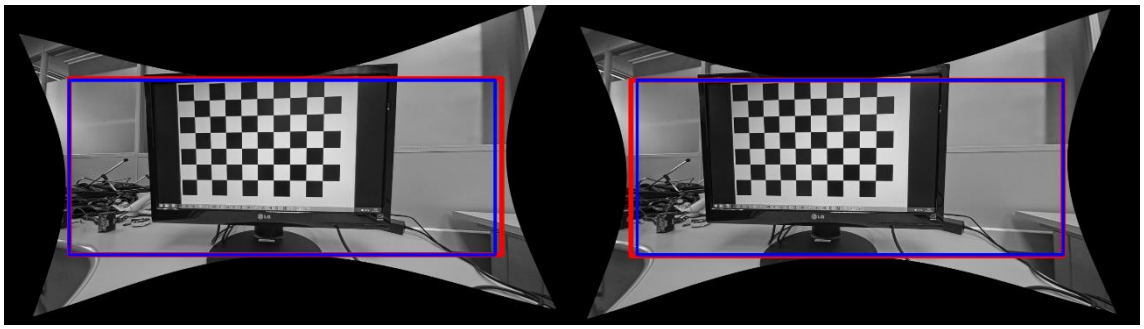


Figure 69 - Left and right view. In red the original ROI of each view. In blue the common ROI.

Source: The author (2016).

3.2.1.5 Color space Conversion

The way used to represent an image may play an important role on the overall result of a given image processing algorithm (FANTONI et al., 2014; LUKAC; PLATANIOTIS, 2006). Different color spaces may be used to represent an image such as RGB, HSV, L*a*b* and so on.

As future steps of the presented methodology uses edges, L*a*b* color space was chosen due its luminosity channel L* (intensity) be separated from the chroma channels a* and b* (color channels), in opposite with RGB where luminosity is distributed along all color channels. This allows enhancements in the lightness of an image without taking into account its color information. FIGURE 70 shows the RGB and L*a*b* color spaces.

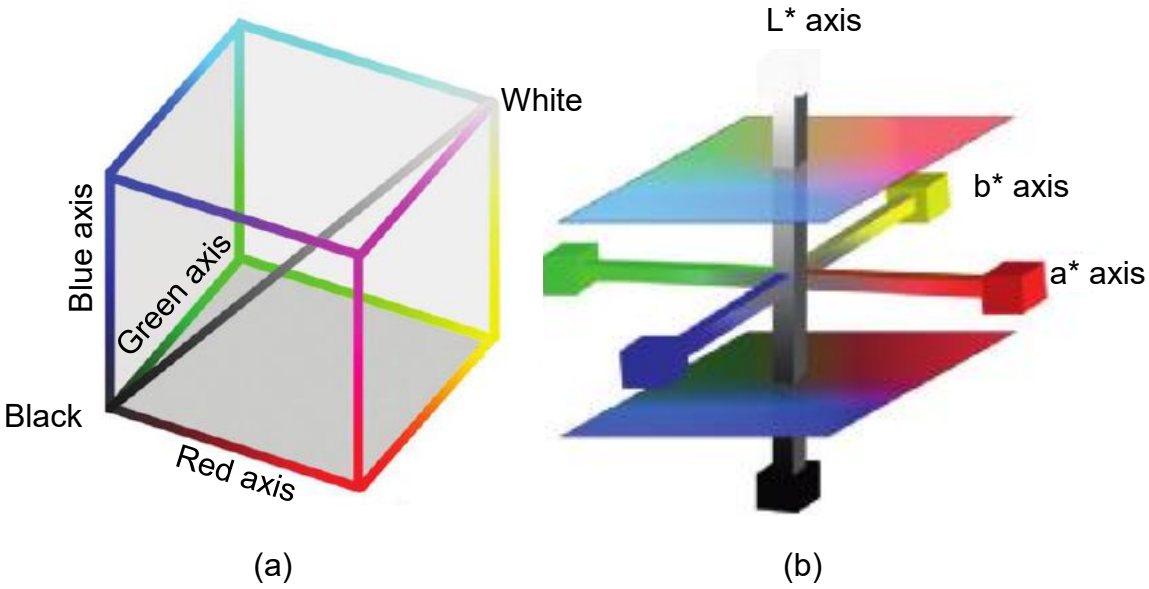


FIGURE 70- Geometric representation of (a) RGB color space and (b) L*a*b* color space.
SOURCE: (a) Russ (2016). (b) Nixon and Aguado (2012).

The conversion from input RGB image to L*a*b* is performed using *rgb2lab* function of Matlab 2015a. The function is simple as it looks, takes a RGB image as input and convert it to L*a*b* using CIE 1976 L*a*b* standard and was set to consider CIE D65 illumination standard, which is intended to represent average daylight (Joint ISO/CIE Standard, 2006; Joint ISO/CIE Standard, 2008).

3.2.1.6 Image enhancement

As this methodology will be applied outdoors and next to vegetation that might block sunlight, it is expected that it faces different illumination conditions along each scene. To decrease the impact of these factors imposed by the non-structured scene the filter proposed by Yang (2013) is applied over the luminosity channel L (from now we will drop the "*") obtained in the previous step. This filter applies a combination of masks that enhance the sharpness of the image while adjusting the illumination as well. The mask used to enhance the sharpness is described as:

$$\nabla^2 L(m,n) = B\ X \sum_{i=-1}^{1} \sum_{j=-1}^{1} \left[\frac{L(m,n) - L(m+i,n+j)}{255} \right]^{\frac{3}{5}} \quad (28)$$

where $(i,j) \neq (0,0)$ and B is a gain.

The mask $\nabla^2 L$ helps to increase the magnitude of the whole image, however, pixels with smaller magnitude receives a higher increment than the pixel with greater magnitude as shown in FIGURE 71.

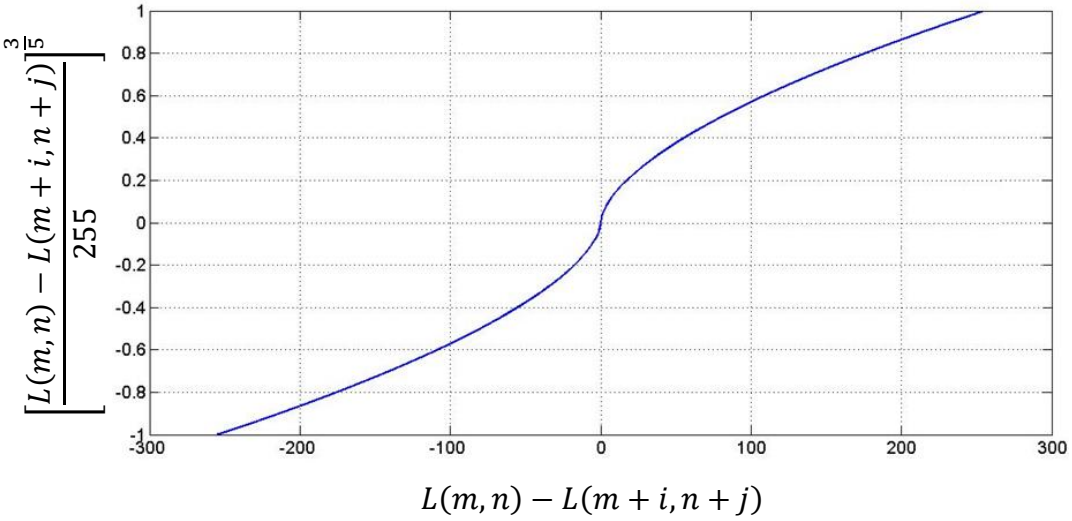


FIGURE 71- Transfer function of equation 28 considering that L is defined within the range 0 to 255.
SOURCE: Adapted from Yang (2013).

A second mask is used to adjust the brightness distribution of the image:

$$\bar{L}(m,n) = A\,X \sum_{i=-1}^{1} \sum_{j=-1}^{1} \left[\frac{L(m,n)+L(m+i,n+j)}{2 \times 255}\right]^{\frac{3}{2}} \quad (29)$$

where $(i,j) \neq (0,0)$ and A is a gain.

The mask $\bar{L}$ helps to smooth regions with similar magnitude, mainly if the magnitude is low. The equation transfer function is shown in FIGURE 72.

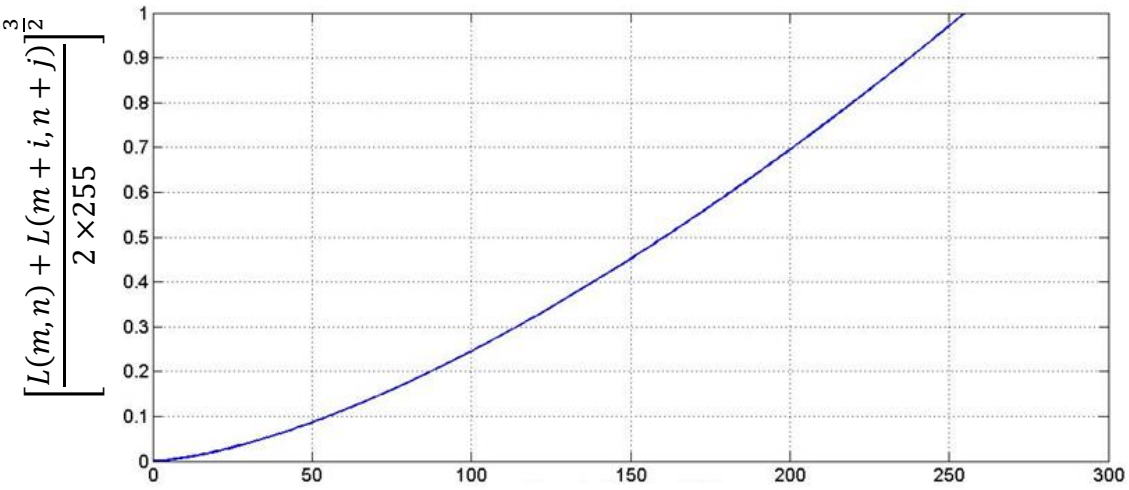


FIGURE 72 - Transfer function of equation 29 considering that L is defined within the range 0 to 255.

SOURCE: Adapted from Yang (2013).

The enhanced image is then obtained by a combination of the two masks and the original image as shown in:

$$G(m,n) = 2 \times L(m,n) - \bar{L}(m,n) + \nabla^2 L(m,n) \quad (30)$$

Here the contrast enhanced image is obtained by using the same gains as in the original article (YANG, 2013), where $A = 35$ and $B = 12$.

3.2.2 Feature extraction

The main feature used by the proposed algorithm are specific edges belonging to a connected component (CC) with a smooth shape. The overview of the current step is shown in FIGURE 73.

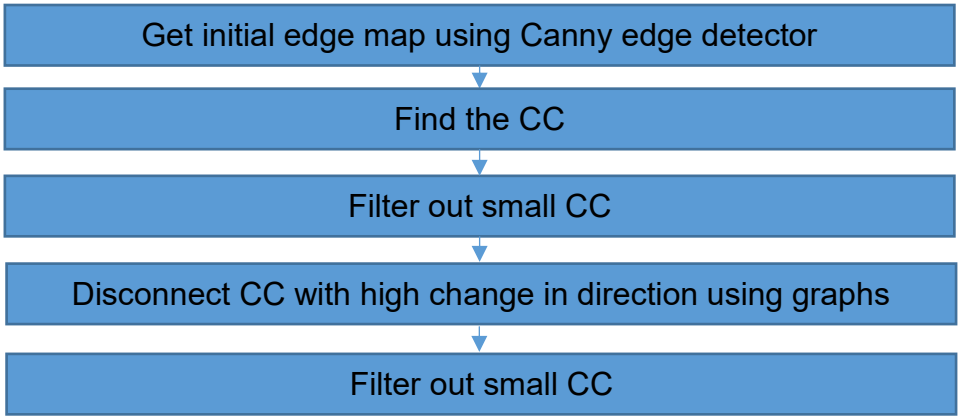


FIGURE 73 - Overview of the feature extraction step.
SOURCE: The author (2016).

To obtain the initial edge map, Canny edge detector is applied over the intensity image with enhanced contrast G obtained during the preprocessing step. The thresholds used in Canny edge detector are defined automatically using p-tile (DOYLE, 1962; PARKER ,2010). To do so, the gradient magnitude ∇G of G is calculated using Sobel. Then, the histogram h_{mag} of the magnitude image ∇G is calculated using k intervals, finally the bins are summed up from the first bin (with lowest magnitude) until reach the bin where the sum represents $x\%$ of the total number of pixels. Therefore, x represent the amoung of pixels that are not edge points. Then the Canny high threshold is defined as:

$$threshold_{high} = \frac{\underset{t}{\operatorname{argmin}}\left(\sum_{i=1}^{t} h_{mag}(i) \geq xmn\right)}{k} \tag{31}$$

where m and n are the number of rows and columns of the magnitude image. Finally, the low threshold is defined as:

$$threshold_{low} = w\ threshold_{high} \tag{32}$$

where w is a percentage.

In Matlab 2015a, this is implemented in the *edge()* function, which returns an image with ones and zeros, where ones (white pixels) are edges and zeros (black pixels) are not edge pixels. As already stated, here the desired features are specific edges with smooth shape. As black pixels are not part of the edge pixels, connected component labelling is then applied to the white pixels considering 8-connected pixels and, in order to remove noise, connected components smaller than $MinCCSize$ are removed. This process is illustrated in FIGURE 74.

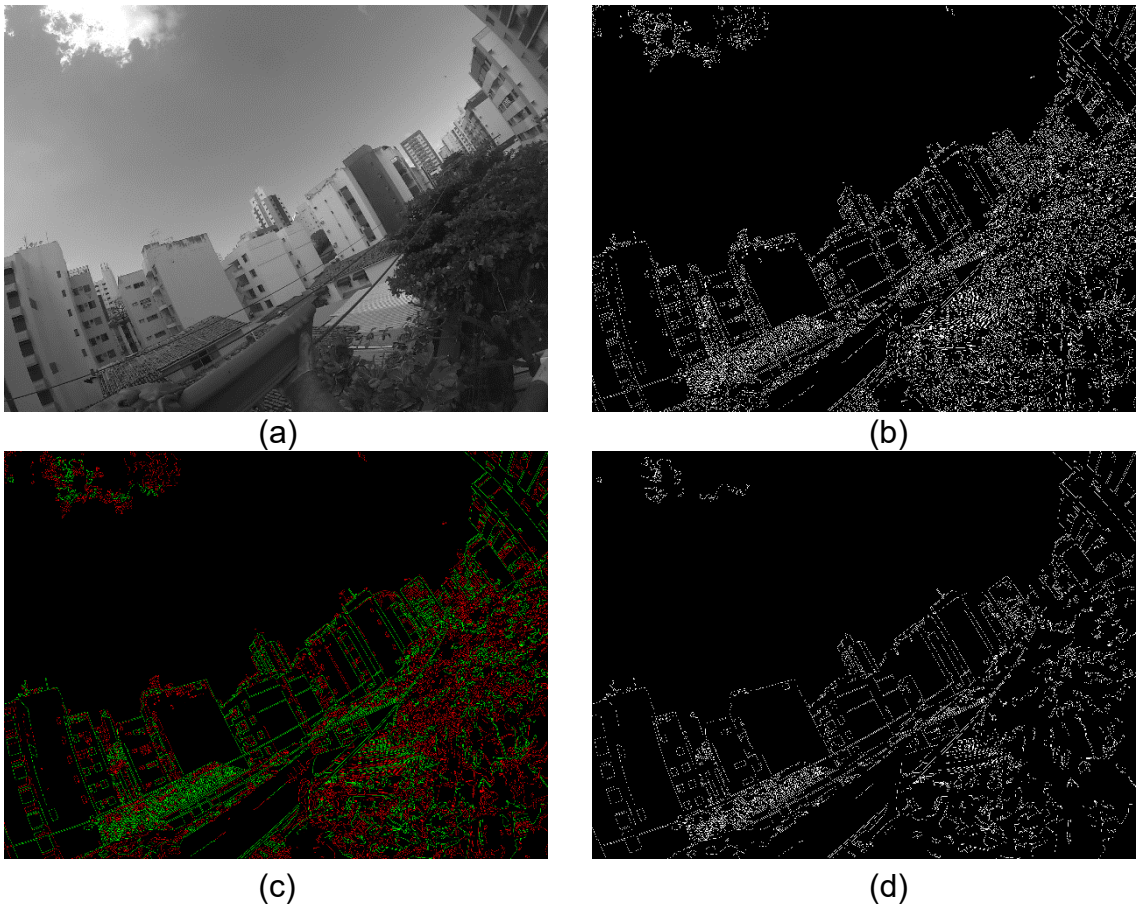


FIGURE 74 - Filtering out connected components smaller than a threshold. (a) Input image. (b) Edge map by Canny edge dectector. (c) In green CC greater than $MinCCSize$. In red CC smaller than $MinCCSize$. (d) Only CC greater than $MinCCSize$.
SOURCE: The author (2016).

This step continue by removing connected components that does not shape like a power line. At first, ellipse eccentricity was used as in Candamo et al. (2009) and Song and Li (2014). The problem is that using these approaches removed true positives when a power line seems connected to any close object, as the branch of a tree. This happened frequently testing it in our dataset, as there are vegetation close to the power lines.

Therefore, it was necessary to develop a new approach that can deal with objects close to energy lines edges. The approach proposed here is to disconnect CC when change in edge direction is greater than $MaxDirectionChange$ threshold. To find the direction, at first, it was used gradient direction, but noise showed that this was not a valid option, causing incorrect disconnections. Finally, to overcome the commented issues a new approach using graph search was developed. It starts by growing a tree in every CC using the algorithm shown in Figure 75.

root = ***Depth First Search(****CC****)***

```
1.  For each pixel p in the CC
2.      p.father = null
3.  Get any pixel p and mark it as not null
4.  root = p      S = empty set
5.  Add root to the stack S
6.  While S is not empty
7.      Remove one pixel v from the top of S
8.      For each pixel u in v neighborhood
9.          If u.father = null
10.             u.father = v
11.             Add u to the top of S
```

Figure 75 - Depth First Search Algorithm.
Source: Adapted from Cormen (2009)

After the tree is grown all leafs are removed to ensure that only real bifurcations remains. After the pruned tree is ready, it is created what was called a discretized tree. Starting from the root and going to the leafs, only one vertex of k vertices is kept, for example the root vertex is kept, the next $k-1$ vertices are eliminated and the next vertex continues repeating this process. With the discretized tree build, it is possible to measure the angle in a specific vertex by measuring the slope of the lines formed by it with its father and by it with its son. If the difference between the angles measured with the father and son is bigger than $MaxDirectionChange$ than it must be disconnected. FIGURE 76 shows an example that helps to explain the idea and FIGURE 77 succinctly shows the pseudocode.

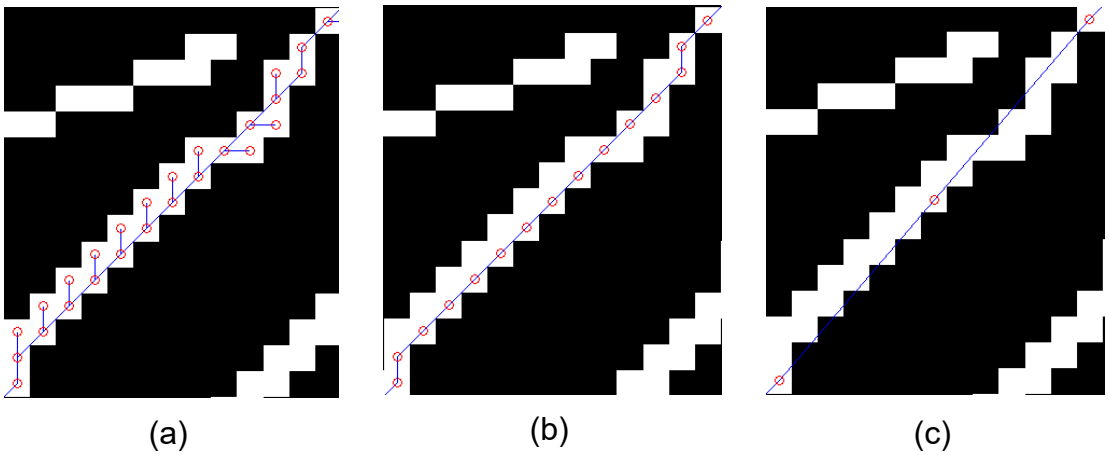


FIGURE 76 - Building a discretized tree. (a) Original tree over the CC. (b) The same tree after removing all leafs. (c) New tree keeping only the *kth* vertex (7 in this example).
SOURCE: The author (2016).

```
cutPoints  = GetCutPoints(startVertex, k)
root = startVertex; q = root; source = startVertex;
cutPoints = empty list
distantce = 0
While q is not empty
    distance = distance + 1
    Remove one pixel v from the top of q
    If v has only one child
        Add v.son to the top of q
        If distance >= k
            v.discreteFather = source
            v.angle = angle formed by v and v.discreteFather
            If source is not the root
                diff = abs(v.angle - v.discreteFather.angle)
                if diff is bigger than MaxDirectionChange
                    cutPoint = RefineCutPointPosition(v)
                    Add cutPoint to the top of cutPoints
                    Remove one pixel from the top of q
                    Add cutPoint to the top of q
            distance = 0
            source = v
    Else if v has more than one child
        For each son of v.children
            Add GetCutPoints(son, k) to the top of cutPoints
        Add GetCutPointsNotCloseTo180(v, k) to the top of cutPoints
```

FIGURE 77 - Pseudocode to cut connected components with edges with high change in direction.
SOURCE: The author (2016).

The function *RefineCutPointPosition* simply shift the vertices $k - 1$ steps forwards and backwards in order to maximize the angle difference, increasing the precision of the cut location as shown in FIGURE 78.

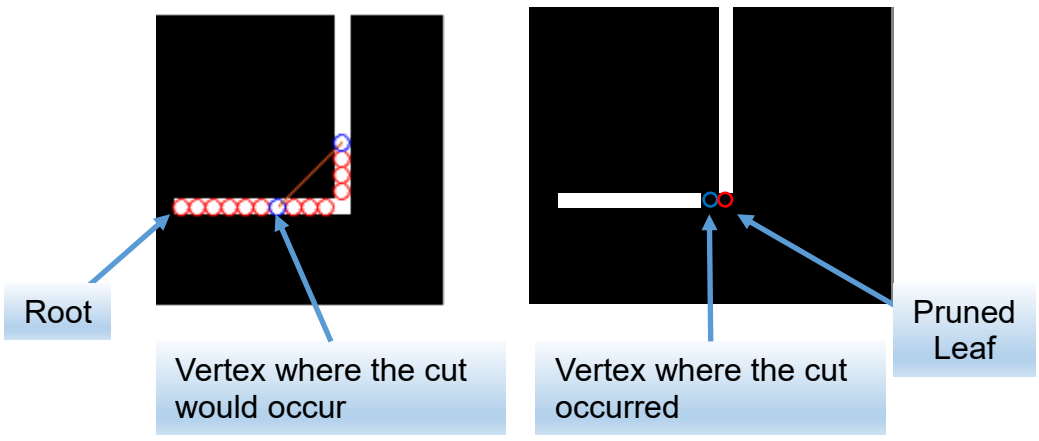


FIGURE 78 - Refinement of the cut point location.
SOURCE: The author (2016).

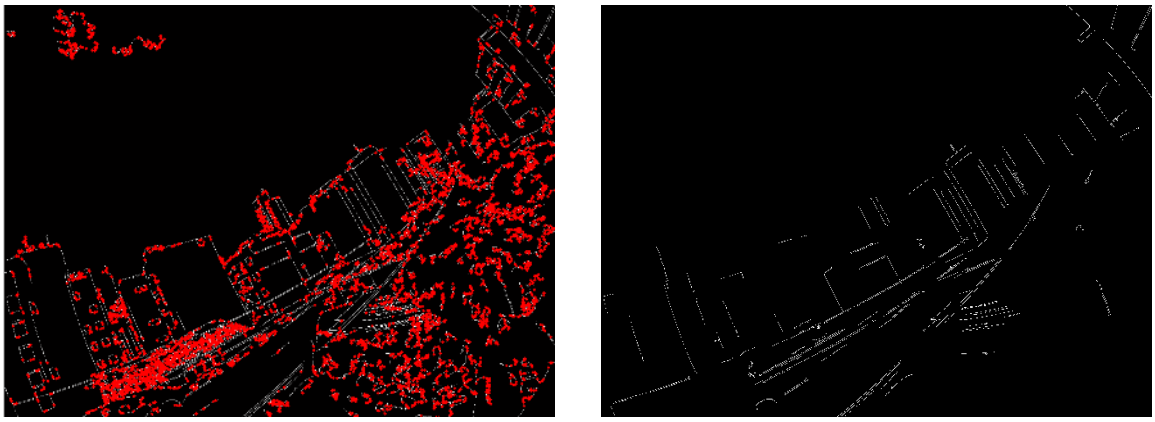
If a bifurcation is found during the execution of the function *GetCutPoints*, the analysis continues for each path. If more than one path have a valid direction, only the path with the angle closest to 180º is kept, while the others are disconnected. This is shown in the pseudocode in FIGURE 79.

PointsToCut = **GetCutPointsNotCloseTo180**(source, k)

```
source.angle = angle formed by source and source.discreteFather
q and s are empty lists
add source to the top of q
add source to the top of s
While q is not empty
    Remove a vertice v from the top of q
    For each son u of v
        u.distance = v.distance +1
        If u.distance < k
            add u to the top of q
        Else
            u.angle = angle formed by u and source
            Add u to the top of s
w,z = Recover the two vertices in s with the most similar angle
diff = abs(w.angle - z.angle)
If diff is bigger than MaxDirectionChange
    PointsToCut = all points connected to the source
Else
    pathWZ = vertices that pass throught the path from w to z
    PointsToCut = Points connected to the source that do not belong to pathWZ
```

FIGURE 79 - Pseudocode to decide the cut location when a bifurcation is found.
SOURCE: The author (2016).

As different CCs were disconnected, CCs smaller than $MinCCSize$ are removed again to get a more filtered edge map. FIGURE 80 shows where the CCs were disconnected and the remaining CCs after filtering by size generating the output edge image $E_{filtered}$.



(a) (b)

FIGURE 80 - (a) In red point where the discretized depth first search broke the CC. (b) Remaining connected components after filtering by size again.
SOURCE: The author (2016).

3.2.3 2D energy line detection

The presented method was designed to perform with high resolution images and to consider that close range energy lines are more important than distant energy lines. Therefore, a CC is considered energy line candidate only if it present a symmetrical CC, in other words where there is one connected component for each energy line border. FIGURE 81 shows an example of power line candidate in green.

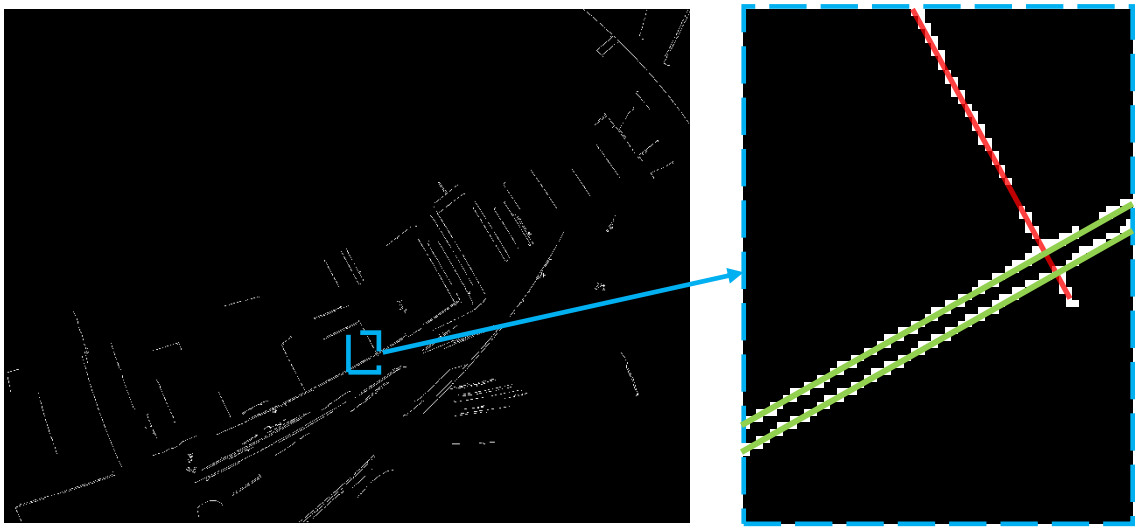


FIGURE 81 – In the left a filtered edge map. In the right zoom of the region in blue. Highlighted in red CC without a symmetrical CC. Highlighted in green, two symmetrical CC.
SOURCE: The author (2016).

In fact, only getting CCs with parallel edges is not sufficient, as many objects can present this geometrical feature (as tree branches), therefore the

smoothness of both curves is considered as well. To retrieve the pairs of borders shown in FIGURE 81 the steps shown in the FIGURE 82 are performed.

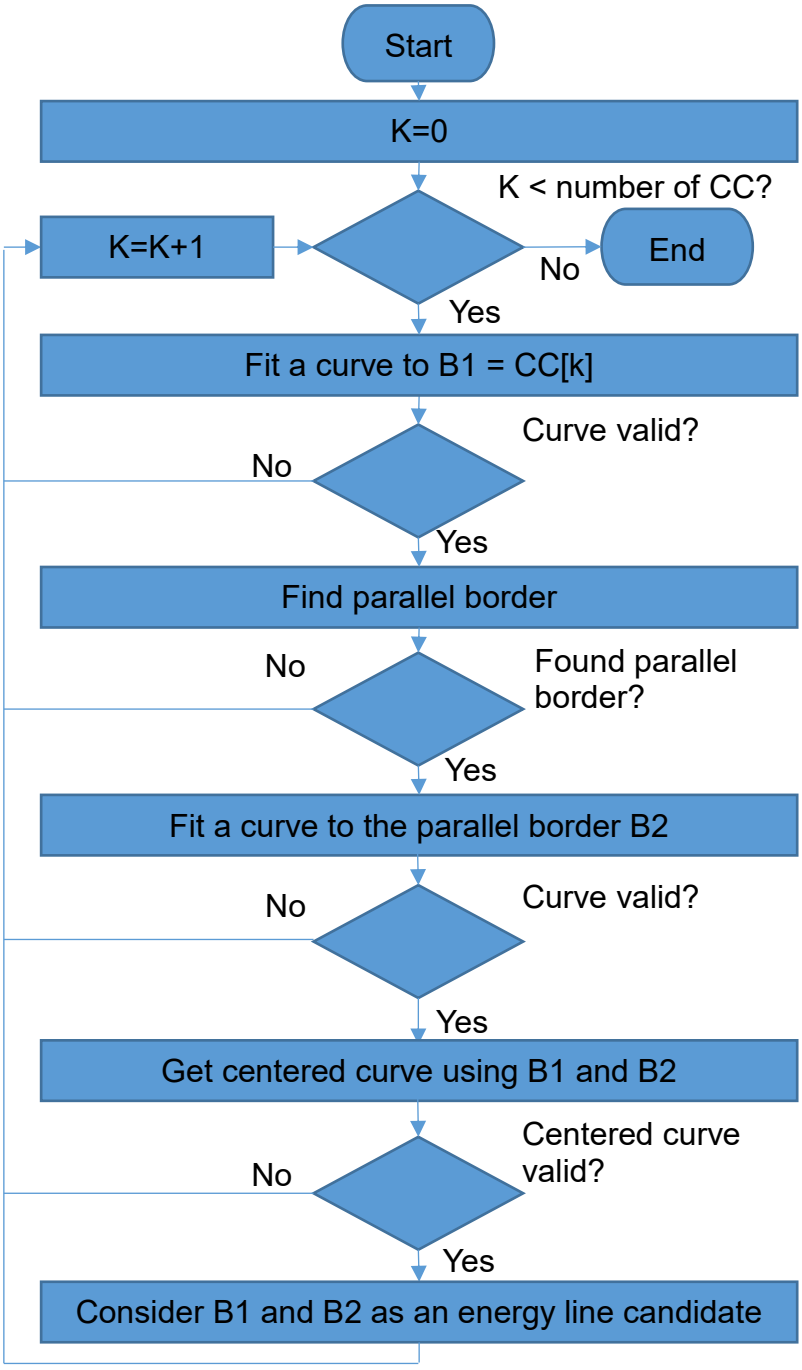


FIGURE 82 - Methodology used to detect energy line candidates.
SOURCE: The author (2016).

In short, for each connected component, a curve is fitted, and then a parallel curve is searched for in the normal direction. If enough points are found, then another curve is fit to these neighbor points. If these curves are considered valid, then the method search for edge pixels in the center between both curves (centered curve), if only few pixels are found then the both borders are considered

an energy line. Each step described here is further explored in the next subsections.

3.2.3.1 Curve fitting

Before explaining the method used here to fit a curve, it is necessary to explain the reason because the traditional methods were not applied. As seen in chapter 2, different authors use Hough/Radon transform to detect straight lines in the edge map. As this system must detect power lines seen from any direction, straight line Hough/Radon would not work, as power lines only appear straight when seen from the top. However, it is possible to implement a Hough/Radon transform to search for a parabola of equation (VENTURI, 2003):

$$x^2 = 2py \tag{33}$$

where x and y are coordinates and p is the distance from the focus to the directrix. The energy line may be shifted in any direction, therefore it is necessary to use a more complete equation, a shifted parabola (VENTURI, 2003):

$$(x - x_0)^2 = 2p(y - y_0) \tag{34}$$

where x, y and p are the same as in (33) and x_0 and y_0 are the coordinates of the shifted parabola vertex. In addition to the shift, the energy line may be rotated, therefore the equation must allow rotation as well. Rotation in Cartesian coordinates is represented by (VENTURI, 2003):

$$\begin{aligned} x &= x' \cos(\theta) - y' \sin(\theta) \\ y &= x' \sin(\theta) + y' \cos(\theta) \end{aligned} \tag{35}$$

Applying the rotation equations over equation 34:

$$(x' \cos(\theta) - y' \sin(\theta) - x_0)^2 = 2p(x' \sin(\theta) + y' \cos(\theta) - yo) \tag{36}$$

where x' and y' are the pixel coordinates, x_0 and y_0 the parabola vertex coordinate and θ the rotation.

As (36) owns four parameters, it means that the Hough/Radon transform would require a four dimensional space. Considering for each parameter an interval of size T, it would be necessary to allocate T^4 bins to perform the transform. If a 4 bytes bin is considered, and $T = 400$, it would be necessary $4{\times}400^4$ bytes, or 95,3GB of memory, what shows that applying the transforms directly to find a parabola impractical.

A general quadratic equation is represented by equation (VENTURI, 2003):

$$Ax^2 + Bxy + Cy^2 + Dx + Ey + F = 0 \tag{37}$$

where the value of the coefficients (A, B, C, D, E, F) determine the type of conic. If $B^2 - 4AC = 0$ with $B \neq 0$ or $A \neq C$ the conic is a parabola. With this in mind, constrained optimization was used to fit parabolas to components. However, it did not always converge to an answer, sometimes resulting in different types of conics.

As both methods shown impractical to fit curves to CCs, an approximation technique was developed. Instead of trying to fit all parameters and instead of rotating the equation, the points were rotated from 0 to π in discrete intervals.

For each rotation, the parabola of equation (34) is fit using least square fitting (WEISSTEIN, 1999), but to do so; it is rewritten it in a slight different way:

$$(x - x_0)^2 = 2p(y - y_0) \Rightarrow y = \frac{x^2 - 2x_0x + x_0^2 + 2py_0}{2p} \tag{38}$$

Considering $a_0 = \frac{(x_0^2 + 2py_0)}{2p}$, $a_1 = \frac{-2x_0}{2p}$, and $a_2 = 1/2p$ it is obtained:

$$y = a_0 + a_1x + a_2x^2 \tag{39}$$

The residual of the points fitted to equation 39 can be calculated as:

$$R^2 = \sum_{i=1}^{n} [y_i - (a_0 + a_1 x_1 + a_2 x_i^2)]^2 \tag{40}$$

Calculating the partial derivative of R^2 equals zero, the parameters a_0, a_1, and a_2 are found. Because these derivatives are for sure a minimum, as it is noticeable that there is no maximum, then the partial derivatives are calculated using chain rule:

$$\begin{aligned} \frac{\partial R^2}{\partial a_0} &= -2 \sum_{i=1}^{n} [y_i - (a_0 + a_1 x_i + a_2 x_i^2)] = 0 \\ \frac{\partial R^2}{\partial a_1} &= -2 \sum_{i=1}^{n} [y_i - (a_0 + a_1 x_i + a_2 x_i^2)]\, x = 0 \\ \frac{\partial R^2}{\partial a_2} &= -2 \sum_{i=1}^{n} [y_i - (a_0 + a_1 x_i + a_2 x_i^2)]\, x^2 = 0 \end{aligned} \tag{41}$$

What can be rewritten as:

$$\begin{aligned} a_0 n + a_1 \sum_{i=1}^{n} x_i + a_2 \sum_{i=1}^{n} x_i^2 &= \sum_{i=1}^{n} y_i \\ a_0 \sum_{i=1}^{n} x_i + a_1 \sum_{i=1}^{n} x_i^2 + a_2 \sum_{i=1}^{n} x_i^3 &= \sum_{i=1}^{n} x_i y_i \\ a_0 \sum_{i=1}^{n} x_i^2 + a_1 \sum_{i=1}^{n} x_i^3 + a_2 \sum_{i=1}^{n} x_i^4 &= \sum_{i=1}^{n} x_i^2 y_i \end{aligned} \tag{42}$$

Rewriting them in matricial form:

$$\begin{bmatrix} 1 & 1 & \cdots & 1 \\ x_1 & x_2 & \dots & x_n \\ x_1^2 & x_2^2 & \dots & x_n^2 \end{bmatrix} \begin{bmatrix} 1 & x_1 & x_1^2 \\ 1 & x_2 & x_2^2 \\ \vdots & \vdots & \vdots \\ 1 & x_n & x_n^2 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ x_1 & x_2 & \dots & x_n \\ x_1^2 & x_2^2 & \dots & x_n^2 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} \tag{43}$$

And in matricial notation:

$$X^T X a = X^T Y \tag{44}$$

where the matrix with x values is called X and its transpose as X^T, the matrix with the coefficients as a and the matrix with the y values as Y, now solving for a:

$$a = (X^T X)^{-1} X^T Y \tag{45}$$

After finding a for each rotation using equation 45, the parabola with the smaller residual is considered the right parabola. As one knows by which angle α the points with the smaller residual were rotated, now it is possible to rotate the parabola of equation (34) and get a rotated shifted parabola of (36).

3.2.3.2 Recovering the parallel curve

To recover two parallel borders, called here B1 and B2, first a CC is considered to be a border B1 and if it is valid (more on this in the next subsection), then parallel pixels are searched for. These pixels can make part of different CC, even if only part of each CC is part of the border B2. Therefore, this search procedure starts from the coordinates of the curve fitted to the valid border B1 and search for its matching curve using the pixels in the edge map $E_{filtered}$.

First, the connected components are sorted by size descending. Therefore, the first CC is the bigger one and is considered to be B1. A match for each pixel of B1 is searched for in the normal direction of the curve found in the previous step. The normal direction of the curve is given by:

$$\theta = \left(\tan^{-1}\frac{dp}{dx}\right) - \alpha \pm \frac{\pi}{2} \tag{46}$$

Where p is the polynomial fitted to B1 and α is the angle by which the points were rotated.

Then using θ it is possible to calculate possible coordinates of a matching pixel using equation 47.

$$X_{B2} = X_{B1} + \lfloor \cos(\theta(X_{B1}, Y_{B1})) \times offSet \rceil \tag{47}$$

$$Y_{B2} = Y_{B1} + \lfloor \sin(\theta(X_{B1}, Y_{B1})) \times offSet \rceil$$

where (X_{B1}, Y_{B1}) are the pixels coordinates of $B1$, (X_{B2}, Y_{B2}) are the pixels coordinates of $B2$ and the $offSet$ determines the distance between the matching pixels, going from two pixels until the maximum of $MaxThickness_{pixels}$. This range is tested for every pixel, if an edge pixel is found in the coordinates of (X_{B2}, Y_{B2}) while the $offSet$ grows the process stops and the next pixel in the curve has its match searched for.

The pixels are processed sequentially from one border extreme to the other. This ordered set of pixels is obtained using the curve fitted in the previous section. Using the angle α that was used to rotate the pixels to perform the fit and sorting them by the rotated x coordinate and rotating them back it is possible to obtain this set of sorted pixels. FIGURE 83 shows the start, the end pixel and allowed search range.

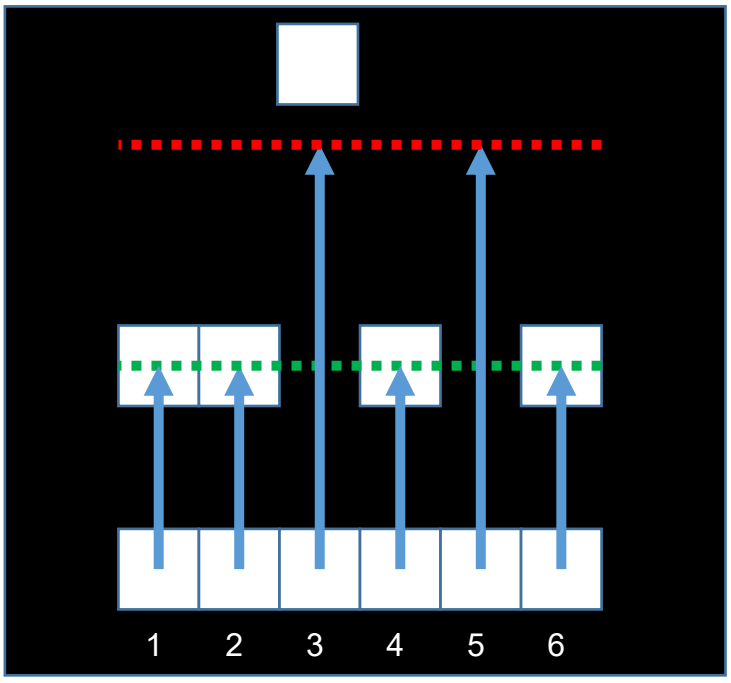
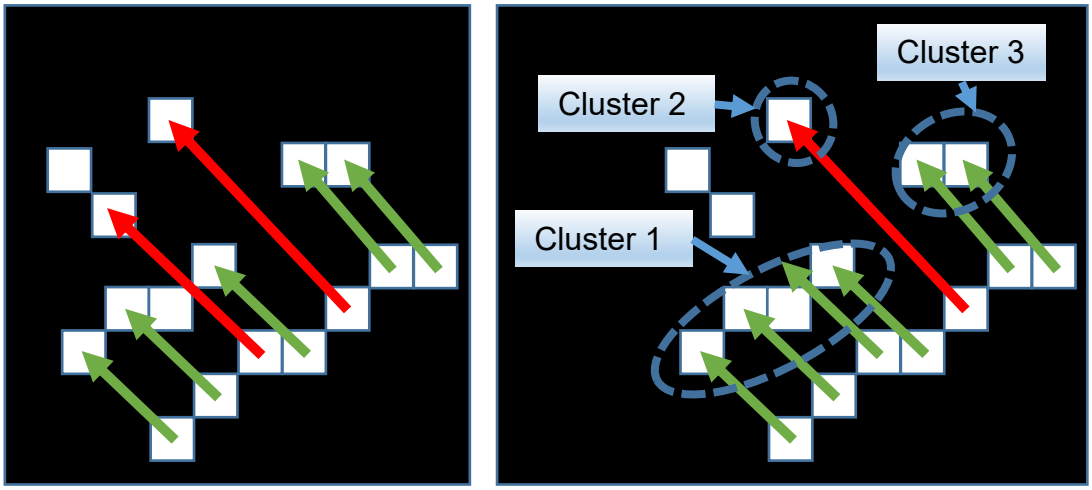


FIGURE 83 – The arrows tail are positioned over the pixels of border B1. 1 is the position where the search starts and 6 is the position where it ends. The green dashed line indicates the minimum value of $offSet$ and the red dashed line indicates the maximum value of $offSet$.
SOURCE: The author (2016).

Different factors can make a pixel be misinterpreted as a matching pixel. For example, discontinuities in the parallel border can cause a false match with a pixel that is further away than the real border and diagonal pixels can allow the

border to be passed through unnoticed. FIGURE 84 (a) shows the false matches caused by discontinuities and diagonal.



(a) (b)

FIGURE 84 - (a) Correct matching pixels in green and incorrect in red. (b) Clusters formed during the search for maching pixels.
Source: The author (2016).

The issue with the diagonal is easily circumvented analyzing neighboring pixels as well, therefore instead of only looking for an edge in $E_{filtered}(X_{B2}, Y_{B2})$, edges in $E_{filtered}(X_{B2} - sign(\cos(\theta(X_{B1}, Y_{B1}))), Y_{B2}))$ and $E_{filtered}(X_{B2}, Y_{B2} - sign(\sin(\theta(X_{B1}, Y_{B1}))))$ are observed as well.

The issues with matches found in gaps requires a more elaborate strategy. Sequential pixels are put in the same cluster if their $offSet$ difference is at most 1 pixel, otherwise a new cluster is created. An example of generated clusters is shown in FIGURE 84 (b).

After all the pixels are processed and with their clusters built, the clusters are analyzes sequentially, fitting neighbor clusters, and if the curve is valid (explained in the next section) these clusters are merged and the procedure continues. At the final of the procedure only the pixels belonging to the biggest cluster are considered as B2. If the size of B2 is at least half of the size of B1 then B2 is considered a parallel curve.

3.2.3.3 Curve validation

The validation intends to filter the borders that are really part of an energy line. Two different types of validation are performed for each of the two types of already introduced curves, the border curves and the centered curve.

The validation used for the border curves requires that:

1. The curve fitting mean square error must be smaller or equal to 1 pixel;
2. The first derivative of the curve in all points in the interval of the fitted curve must be smaller than 1.5.

These requirements basically ensures that a polynomial was capable to describe the set of pixels well and that it is not growing too fast, as the energy lines are curve with smooth slopes.

A different validation is performed to the centered curve, using the assumption that there must be no edge between parallel borders of an energy line and that the slope of both border curves must be similar. The validation used for the centered curve requires that:

1. The amount of edge pixels among the centered curve pixels must not be greater than 5%;
2. $diff$ must not be greater than the threshold $MaxDiffCurvature$.

where $diff$ is defined as:

$$diff = \frac{\sum_{i=1}^{n} |d_{B1}(x_i) - d_{B2}(x_i)|}{n} \quad (48)$$

and n is the number of matching points, d_{B1} and d_{B2} are the first derivatives of the curves associated with B1 and B2 respectively.

Finally, two parallel borders that passes all these requirements are considered an energy line candidate.

3.2.4 Depth estimation

The method used to obtain the disparity map is a simplification of the proposed by Bleyer and Gelautz (2005) to create an initial disparity map, except that here the segmentation part is not applied and at the end of the process the disparity is filtered using median filter. FIGURE 85 gives an overview of the procedure.

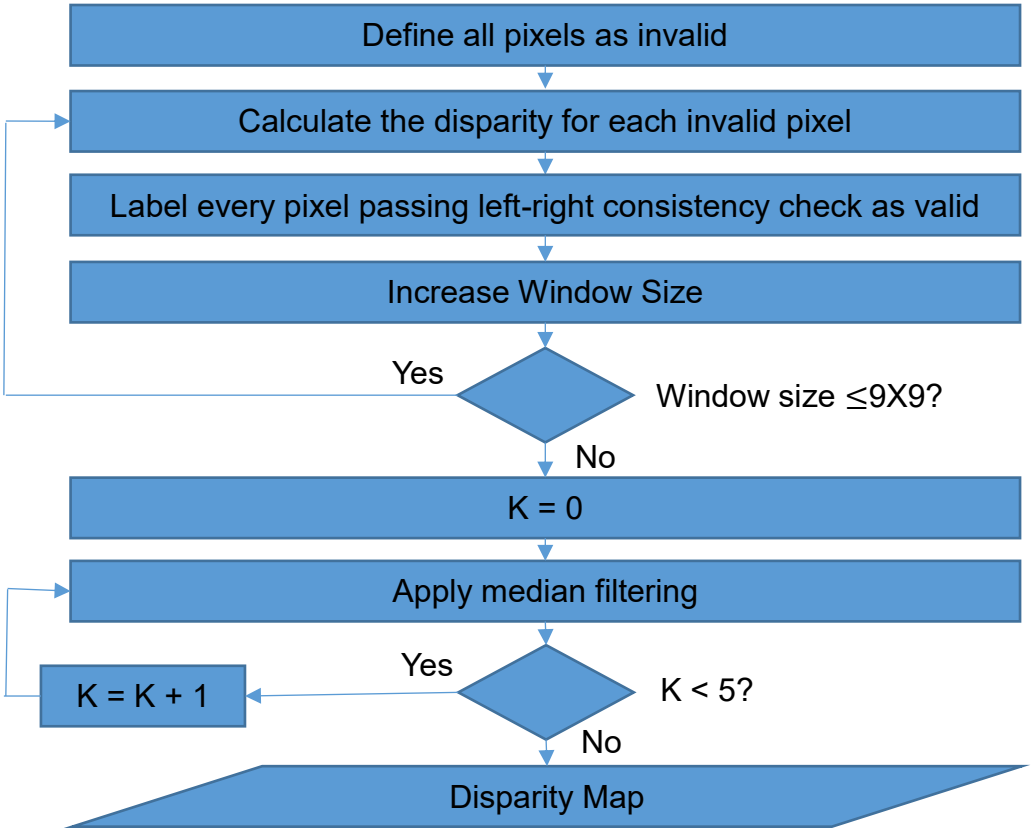


FIGURE 85 - Methodology used to obtain the disparity map.
SOURCE: Adapted from Bleyer and Gelautz (2005).

The method uses a local window with variable size, starting with 3 x 3 window and resizing it until 9 x 9 growing by two each dimension. The method starts calculating the disparity map using the left image as reference and the smaller window, while the costs are computed using sum of absolute differences over the RGB color image, as shown in equation:

$$Cost_l(x, y, c, d) = |I_l(x, y, c) - I_r(x - d, y, c)| \tag{49}$$

where x and y are pixels coordinates, c is the color channel and d is the disparity.

The left disparity map calculation is shown in:

$$d_l(u, v) = \underset{0 \le d \le d_{max}}{\operatorname{argmin}} \left(\sum_{i=-w}^{w} \sum_{j=-w}^{w} \sum_{c=1}^{3} Cost_l(u + i, v + j, c, d) \right) \tag{50}$$

where u and v are pixels coordinates, d is the disparity, d_{max} is the maximum disparity, $w = \left\lfloor \frac{window\ size}{2} \right\rfloor$.

The same equation 50 is used to calculate the disparity map of the right image $d_r(u,v)$, but with a small modification in the cost function:

$$Cost_r(x,y,c,d) = |I_l(x+d,y,c) - I_r(x,y,c)| \quad (51)$$

where x and y are pixels coordinates, c is the color channel and d is the disparity.

The two obtained disparity maps d_l and d_r are then used to perform a left-right consistency check, verifying if the disparities are coherent. The left right consistency check is expressed as:

$$CrossChecked_l(u,v) = \begin{cases} valid, & d_l(u,v) = d_r(u - d_l(u,v), v) \\ invalid, & otherwise \end{cases} \quad (52)$$

All the pixels passing the left-right consistency check are considered as valid pixels, while all the others as invalid. The invalid pixels have their disparity recalculated again, but using a bigger window size until the size limit is reached.

Finally, as unfortunately errors might pass the left right consistency check, a median filter is applied to reduce its effect. A 3 x 3 window is used and the median filter is applied 5 times.

Now, using the disparity map d_l and the pixel coordinates (u,v) it is possible to calculate the 3D world points using the Q matrix obtained along the stereo calibration. For each pixel the 3D coordinates in homogeneous coordinates are:

$$\begin{bmatrix} X \\ Y \\ Z \\ W \end{bmatrix} = Q \begin{bmatrix} u \\ v \\ d(u,v) \\ 1 \end{bmatrix} \quad (53)$$

Therefore, the 3D world points coordinates in Cartesian coordinates are $(\frac{X}{W}, \frac{Y}{W}, \frac{Z}{W})$.

Equation 53 is applied for every pixel creating a depth map called $WorldPoints_{xyz}(u,v)$, where u and v are pixels coordinates. Then, the 3D coordinates can be recovered later for any pixel just by accessing it thorough pixels coordinates.

3.2.5 3D energy line filtering

As the image are acquired in urban environment, many manmade constructions are present, what can cause false edges to be taken as an energy line candidate during the 2D energy line detection. Furthermore, trees present branches that may look like energy lines as well. To decrease the amount of false positives the selected energy lines are now filtered using 3D features. To accomplish this, two different approaches are used: one using known properties of the energy line geometry and a second measuring the 3D line thickness, these are detailed in the next subsection.

3.2.5.1 Filtering comparing the expected depth and stereo depth

As the energy line is a known object from the pinhole camera model, it is possible to estimate its depth. Taking into account the focal length obtained during the calibration step and the thickness measured in pixels it is possible to estimate the depth expected for a line of a certain thickness. Equation 54 shows how to obtain the depth and FIGURE 86 demonstrates the concept. This equation is given by:

$$Z = \frac{fW}{w} \tag{54}$$

where Z is the depth in centimeters, W is the diameter of the line in centimeters, w is the distance between matching pixels and f is the focal length in pixels.

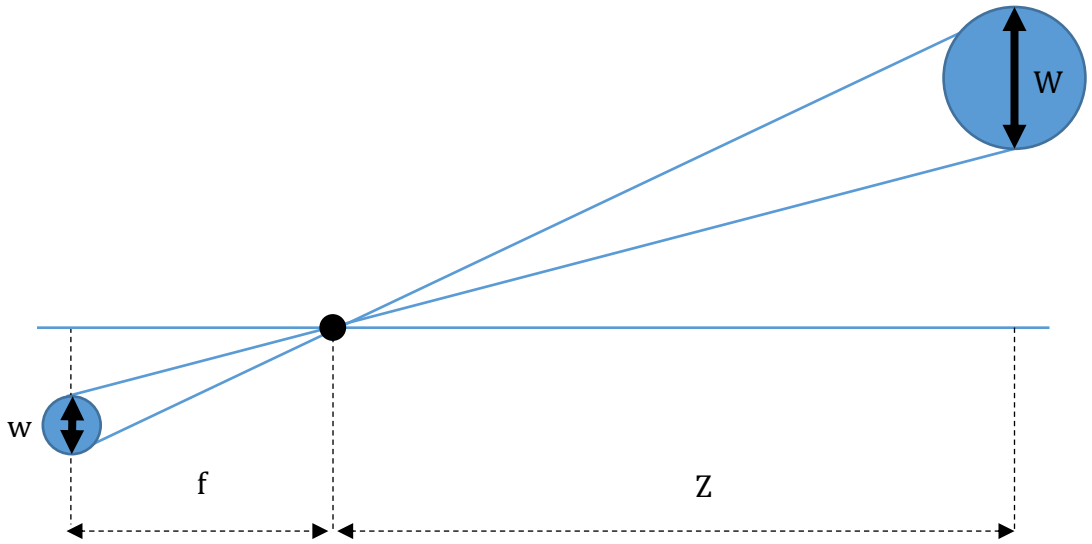


FIGURE 86- Projection of a line with known diameter using the pinhole camera model.
SOURCE: The author (2016).

As the depth information is available as well from the stereo measurements it is possible to compare it with the depth expected using the energy line thickness in pixels. To do so, first it is necessary to find the thickness in pixels of each line. This is performed by finding matching pixels one more time (similar to what was performed in 3.2.3.2), but now using the edge positions of B2 set by the polynomial used to approximate it, therefore without any discontinuities. Once again, the pixels are looked for in the direction of the normal of the polynomial used to approximate each border as shown in equation 55:

$$\theta = \left(\tan^{-1}\frac{dp}{dx}\right) - \alpha \pm \frac{\pi}{2} \qquad (55)$$

where p is the polynomial used to approximate the border and α is the angle by which the points were rotated.

The application of equation 55 is shown in the pseudo code of FIGURE 87 and an example of matching pixels are shown in FIGURE 88

pairs = **Find parallel pixels** (B1,B2)

```
1    BW = binary image with all pixels equal to zero
2    BW(border1)  = 1
3    Side = -1;
4    pairs = empty list
5    For each pixel p in B2
6        offSet = 2
7        While offSet is smaller than MaxThickness_pixels
8            offsetX = round(cos(θ(p)) * offSet)
9            offsetY = round(sin(θ(p))  * offSet)
10           parallelPixel = (p.x+offSetX,p.y+offSetY)
11           If BW(parallelPixel) is equal to 1
12               add parallelPixel and p to pairs
```

FIGURE 87 - Find continuous Matching pixels.
SOURCE: The author (2016).

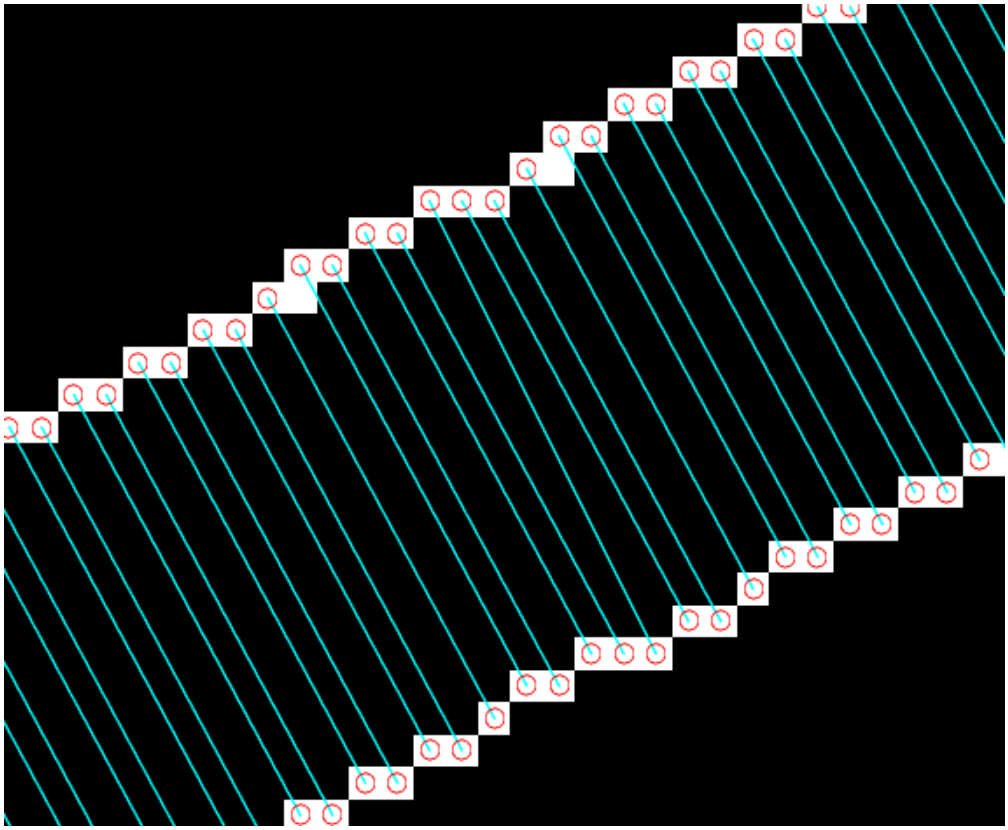


FIGURE 88 - Example of matching pixels.
SOURCE: The author (2016).

The width in pixels is then obtained by calculating the Euclidean distance between matching pixels. Considering a minimum real world thickness and a maximum real world thickness it is possible to calculate one maximum expected

depth (upper bound, equation 57) and one minimum expected depth (lower bound, equation 56) using equation 54.

$$Z_{min} = \frac{fW_{min}}{w_{pixels}} \tag{56}$$

$$Z_{max} = \frac{fW_{max}}{w_{pixels}} \tag{57}$$

where W_{min} and W_{max} are the minimum and maximum allowed thickness of an energy line in real world units respectively, f is the camera focal length in pixels and w_{pixels} the thickness in pixels.

Notice that Z_{min} and Z_{max} must bound the depth of the same energy line measured using stereo vision if it really is an energy line. The stereo measure Z_{stereo} is acquired by reading the world points map Z coordinate at the mean coordinate of a pair of parallel pixels, as:

$$\begin{aligned} x_{stereo} &= \left\lfloor \frac{(x_1 + x_2)}{2} \right\rfloor \\ y_{stereo} &= \left\lfloor \frac{(y_1 + y_2)}{2} \right\rfloor \end{aligned} \tag{58}$$

where (x_1, y_1) are points that belong to B1 and $(x_2 y_2)$ are points that belong to B2. Finally, the Z_{stereo} is defined as:

$$Z_{stereo} = WorldPoints_z(x_{stereo}, y_{Stereo}) \tag{59}$$

As already shown in section 3.1.1.1, the stereo measurements lie within a range. Therefore, this lack of precision is included in the boundaries as well. As shown in equations 60 and 61.

$$Z_{upper} = Z_{max} + \frac{rZ_{upper}^2}{fb - rZ_{upper}} \tag{60}$$

$$Z_{lower} = Z_{min} - \frac{rZ_{stereo}^2}{fb - rZ_{stereo}} \quad (61)$$

where r is the pixel size, f is the focal length and b is the baseline.

FIGURE 89 shows an example of the boundary Z_{upper} limiting a stereo measurement Z_{stereo} from above and Z_{lower} limiting it from bellow.

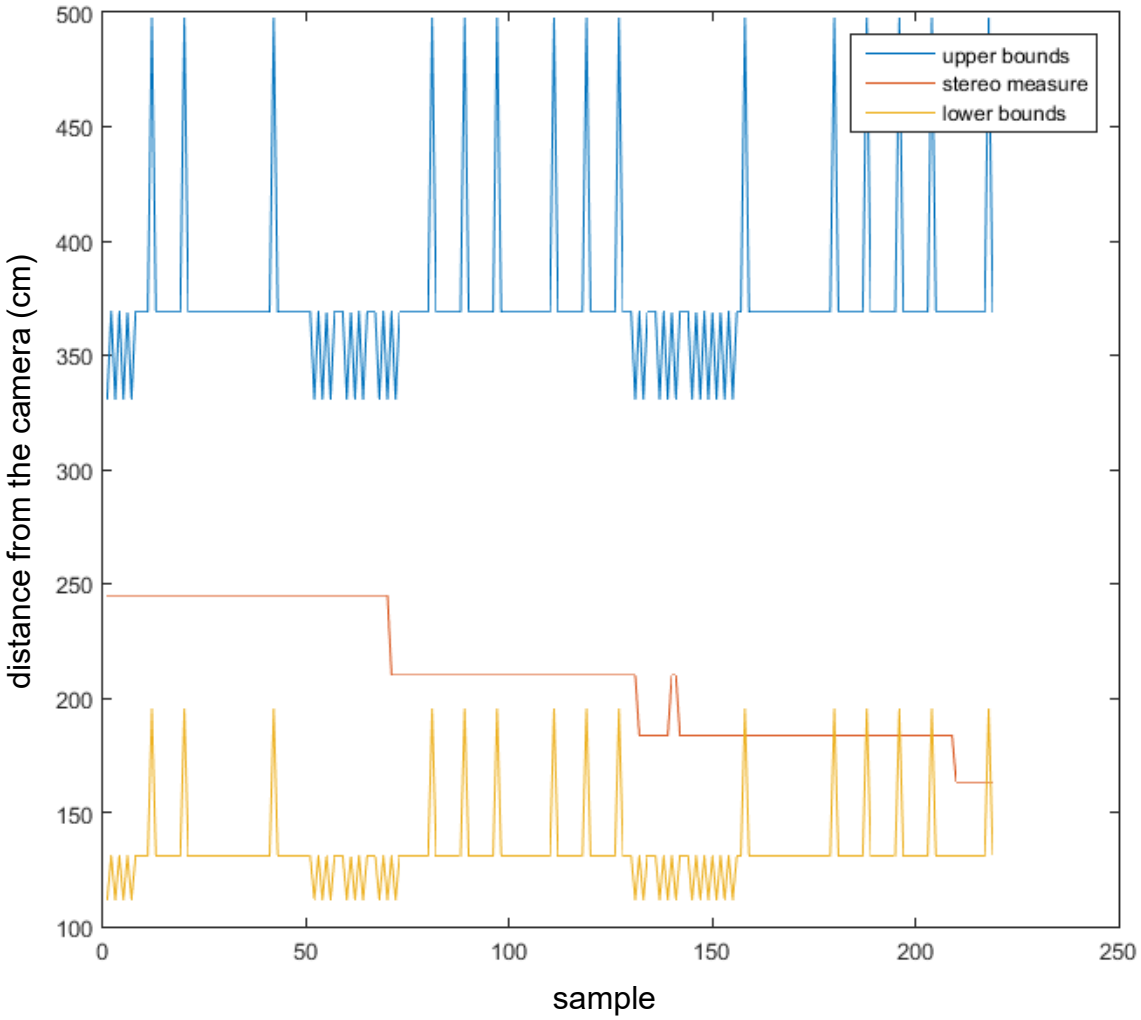


FIGURE 89 - Example of stereo measurement of energy line points bounded by expected distance for a line of 2cm (upper bounds) and 1cm (lower bounds) of thickness.
SOURCE: The author (2016).

Finally, if more than 75% of the stereo measurements are bounded by Z_{upper} and Z_{lower}, then it is considered a real power line, otherwise it is filtered out.

3.2.5.2 Filtering measuring 3D thickness

An energy line is expected to have its thickness measurement laying within a certain range. As a small variation in depth might cause wrong measurements as the points would be lying in depths with different resolutions it is considered here that matching pixels found in the previous section present the same depth. In this case, the depth used for each pair of matching pixels is the depth obtained in the center of the energy line, therefore with coordinates defined in the same way as in the previous section in equation 58. As the center depth might be different from the depth found in the border of the line, the X and Y 3D coordinates must be recalculated. Before recalculating these values, the disparity values from the center of the energy line are filtered using 1D median filtering.

Using the Q matrix and the disparity of the energy line center the 3D points of each border are calculated as:

$$\begin{bmatrix} X \\ Y \\ Z \\ W \end{bmatrix} = Q \begin{bmatrix} u_{border} \\ v_{border} \\ d_{center} \\ 1 \end{bmatrix} \tag{62}$$

where u_{border} and v_{border} are the coordinates of the border pixel, d_{center} is the disparity at the center, W is used to convert the homogenous coordinates to Cartesian coordinates and then $(\frac{X}{W}, \frac{Y}{W}, \frac{Z}{W})$ are the 3D points coordinates.

Now using equation 62, the world coordinates of each parallel point are recalculated. With the 3D points of both borders, finally the thickness is calculated using Euclidean distance.

The so far obtained values of thickness for each matching point might contain huge values due disparities close to zero, which makes the distance tend to infinity. Therefore, using mean to get a unique thickness measurement would not be appropriate. Consequently, the median value is used.

To check the validity of the obtained thickness using a static upper threshold would be incoherent as with the distance the depth resolution decreases, what might not allow one to measure such thin structures as lines with enough precision. Therefore, a different upper threshold is used for each pair of

parallel pixels according to their depth. The minimum perceived distance for each pair of matching points is defined as:

$$\Delta dist = |P_1 - P_2| \tag{63}$$

where P_1 and P_2 are 3D points calculated using equation 62 and 2D points with coordinates (u, v) and $(u + 1, v)$ with the same disparity.

Therefore, the upper threshold $\mathrm{Thd}_{\mathrm{upper}}$ must hold the inequation

$$\mathrm{Thd}_{\mathrm{upper}} = K.\Delta dist \geq W_{max} \tag{64}$$

where W_{max} is the maximum allowed thickness for an energy line and as the measurements are discrete, K must be integer, therefore:

$$K \geq \left\lceil \frac{W_{max}}{\Delta dist} \right\rceil \tag{65}$$

The lower threshold $\mathrm{Thd}_{\mathrm{lower}}$ is fixed to W_{min} as, in the worst case, the lack of resolution makes the stereo measure bigger than it is, not smaller.

The thresholds are calculated for every pair of matching points and if more than 75% of the points respect the minimum and maximum thresholds the line is kept, otherwise it is filtered out.

3.2.6 Merging energy lines

Among the energy lines that passed by the filtering procedures might exist energy lines that are segments of a unique energy line. In order to recover the energy line gaps some checking is performed to find energy lines candidates that describe the same energy line. FIGURE 90 shows the methodology adopted here.

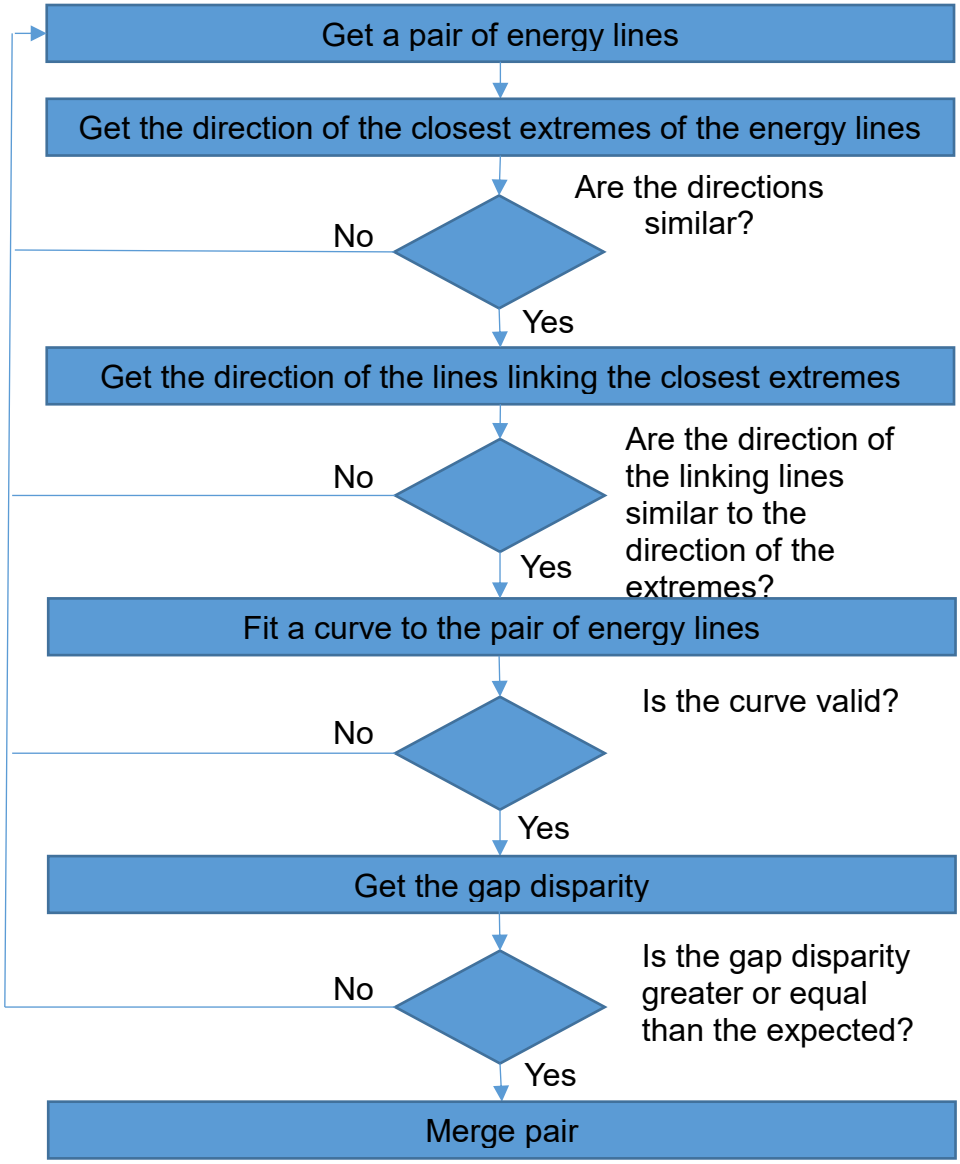


FIGURE 90 - Overview of the merging line procedure.
SOURCE: The author (2016).

The first test performed to test line similarity takes into account the direction of the extremes of both borders of each energy line and compares it against the other energy lines. The direction is calculated as shown in equation 66.

$$\theta = \tan^{-1}(y_{extreme} - y_{inner})/(x_{extreme} - x_{inner}) \quad (66)$$

where θ is the arctangent of the slope of a straight line with points (x, y) with two different subscripts. The subscript $extreme$ represents the point that belongs to the extreme of the line and $inner$ the point that is M pixels away from the extreme.

If the difference of direction of any border is greater than $MaxExtremeDiff$, the energy lines are not the same. Otherwise, the angles of the two lines linking the both closest borders extremes of the energy lines is compared with the angle of the extremes, if its difference is greater than a $MaxExtremeDiff$, then again the energy lines are not the same. FIGURE 91 shows an example of energy lines segments and the linking lines with the respective angles.

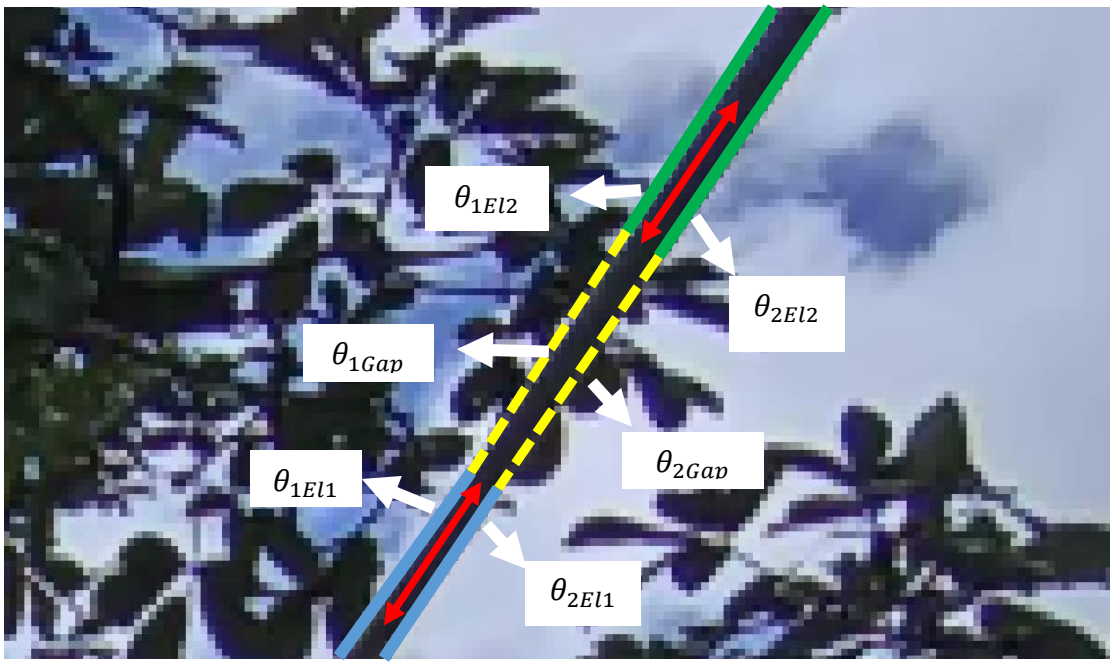


FIGURE 91 - Lines segments in blue and green. In yellow, the line connecting the closest extremes of both energy lines. In red, the M distance that separates the extreme pixels from the inner pixels.
SOURCE: The author (2016).

In the case where two power lines were considered the same one by the angle test, then a new energy line is fitted to the points that describe both energy lines. If the each curve fitted to each border is not valid considering the validation presented in subsection 3.2.3.3, then the segments are not merged.

The merged lines that passed the angle test and the fitting test, passes now through a last check, which verifies the depth of the gap between the segments of energy lines. Two scenarios are considered as valid: the first when the depth is the expected, where the energy line were not detected as a whole due problems during the detection in the previous stages; a second when there is an object blocking the view of the energy line, therefore showing a closer depth than expected. The expected depth is obtained by fitting a second-degree polynomial to the depth of the centered curve disparities of each energy line

segment, while the real depth is obtained directly from the coordinates of the centered curve over the disparity map. If the line passes this test, then both are merged and become one single line.

3.2.7 Energy line growing

After linking all the segments that belong to the same energy line in the previous subsection, there still might be energy lines that were not detected in their full extend. To recover these parts of the lines this step seeks to extend them as much as possible. This is performed in an alternate fashion, trying to grow a certain extension to each extreme of the line one time, refitting it again using the new points and then the process repeats. Basically, the growing is divided in two steps, one two check if it is possible to grow and another to perform the growing respecting the borders position. The logical sequence is shown in FIGURE 92 and the main steps are divided in the next subsections.

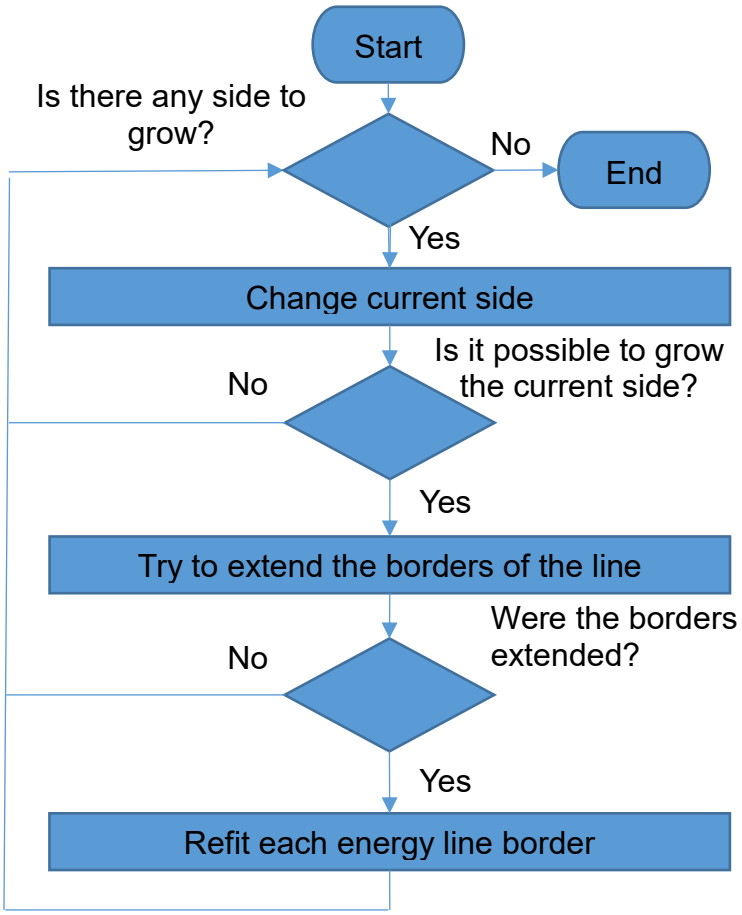


FIGURE 92 - Overview of the energy line growing method.
SOURCE: The author (2016).

3.2.7.1 Checking if the line can grow

The method proposed here assumes that the center of each line does not present edges, or if it does, it does not block the path along both energy line borders. Therefore, a centered curve is calculated using both energy lines borders as previously defined in equation 58.

Using the last M points of each extreme, a straight line is fitted to get a prediction of possible points. Using the equation of the fitted straight line a prediction of size $\frac{M}{2}$ is made. This prediction must no collide with any pixels, because in this case the algorithm consider that it is taking the wrong direction or the energy line end was reached. The idea is similar to a catheter passing through a vein where the energy line border are the vein walls.

The extended line receives offsets starting from 0 that linearly increases/decreases until ± 3. Different conditions are evaluated for each offset. In case of failure of any condition the range is reduced to ± 2 and in case of failure again the procedure continues reducing the range until the offset is 0. If none of the possible extended line passes thought the tests, than it is considered that is not possible to grow anymore in that line extreme. FIGURE 93 shows the offsets

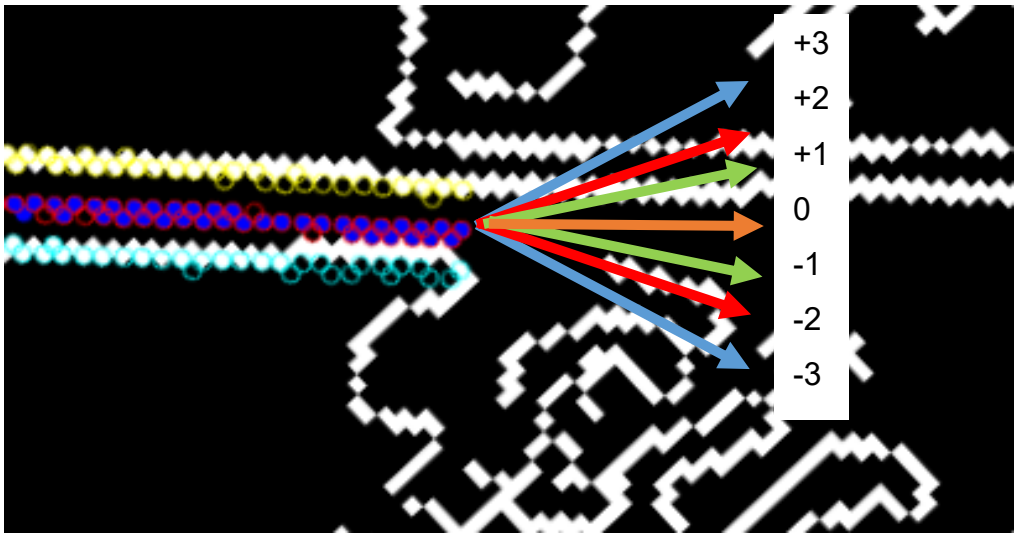


FIGURE 93 - Energy line borders in yellow and cyan. Centered curve in blue and red dots. Predicted centered curve with 0 offset in orange, ± 1 in green, ± 2 in red and ± 3 in blue.
SOURCE: The author (2016).

The first condition verifies if the line is “colliding”, that is, if it is passing over any edge pixel. If it does, the extended line is invalid, otherwise, another tests is performed evaluating the disparity.

The disparity of the extended line is compared against the previous disparities along the centered curve. The previous disparities are calculated using M points from the line extreme to its interior, this interval is divided in two with size $\frac{M}{2}$ each. The median of the each interval is considered the previous disparities. FIGURE 94 helps to understand the concept. The difference between both medians is the delta disparity calculated as:

$$\delta d = |d_n - d_{n-1}| \tag{67}$$

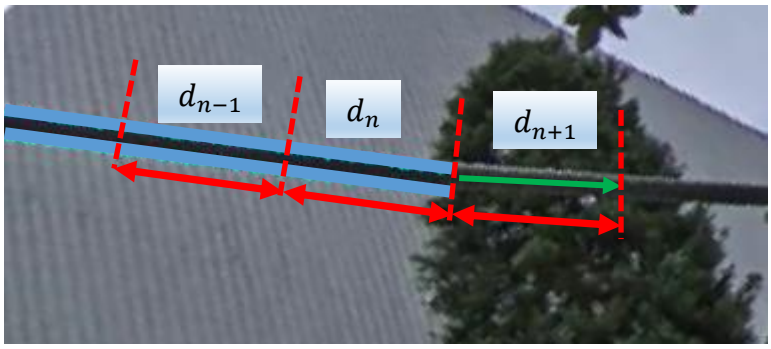


FIGURE 94 - Previous disparities. Each red arrow represents M pixels. The green arrow is the extended range.
SOURCE: The author (2016).

If the median of the disparity of the extended line is within the expected range calculated as:

$$d_{insideRange} = \begin{cases} 1, if\ d_n - \delta d - tol \leq d_{n+1} \leq d_n + \delta d + tol \\ 0, \quad otherwise \end{cases} \tag{68}$$

where the tolerance tol is 1, then it is considered a valid extension, otherwise invalid.

If both offsets (*e.g.* +3 or -3) are valid then a score is used to decide the right path. This score is calculated using k-nearest neighbor (KNN) classifier, which classifies new data (test data) using the k nearest neighbors obtained from known data (training data), where k is an integer. The training data are vectors with class labels, where each vector coordinate is a feature. The test data is similar, but lacks the class label, which is obtained by assigning to it the class of the majority of the k nearest neighbors in the training data.

Here, the already known energy lines points close to the current extreme are used as training data. The centered points are enlarged by ± 1 and are labeled as energy line points. The not energy line points are points outside the energy line border with an extra offset to consider the border region as fuzzy, as the border might be badly located. FIGURE 95 helps to understand the concept used to gather the pixels used as training and test.

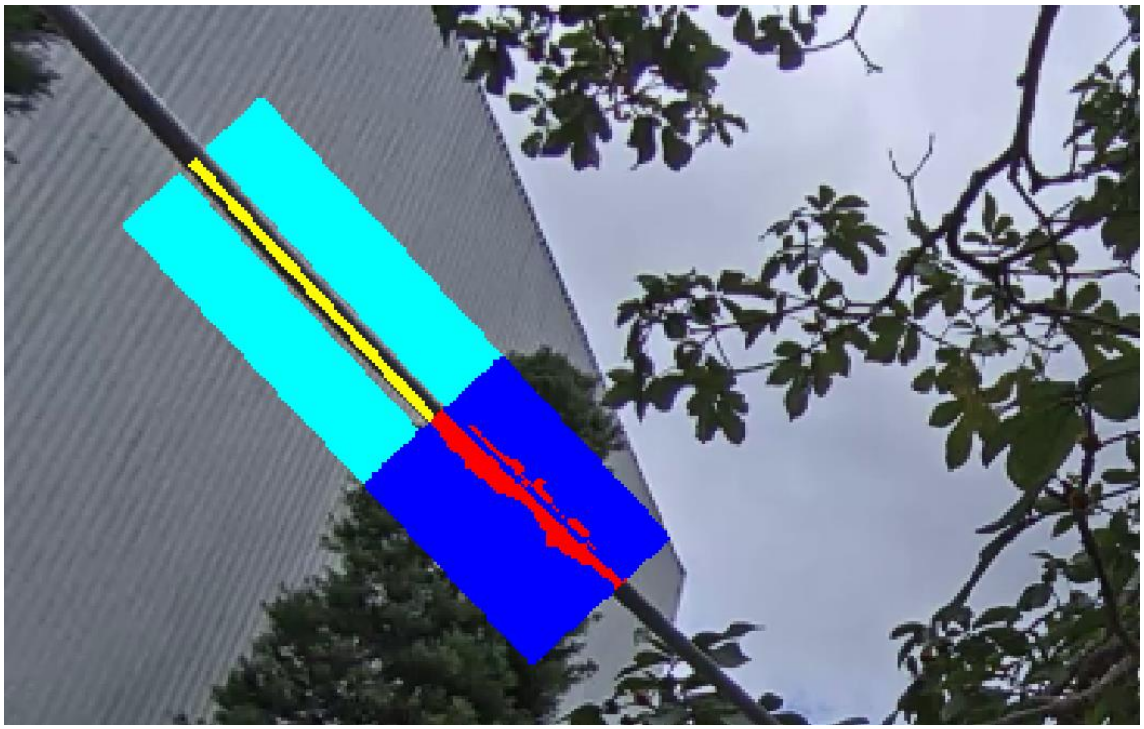


FIGURE 95 - KNN used to predict energy line points. Where yellow and cyan are training points, with yellow meaning energy line points and cyan not energy line points. The red and blue are validation points, where the red points were classified as energy lines and the blue as not energy lines.
SOURCE: The author (2016).

The features used for the classification are each color channel in RGB format, the pixels coordinates and the disparity. All the features are normalized values ranging from 0 to 1. FIGURE 95 and the correspondent class.

TABLE 2 shows 10 samples of the data extracted from FIGURE 95 and the correspondent class.

TABLE 2 - Samples of data used to train the KNN classifier

Column	Row	R	G	B	Disparity	Class
0.00	0.37	0.15	0.15	0.16	0.16	energy line
0.32	0.37	0.14	0.15	0.15	0.16	energy line
0.32	0.36	0.15	0.15	0.16	0.16	energy line
0.32	0.36	0.15	0.15	0.16	0.16	energy line
0.32	0.36	0.14	0.15	0.15	0.16	energy line
0.29	0.21	0.48	0.50	0.55	0.14	other
0.28	0.20	0.45	0.47	0.52	0.14	other
0.28	0.20	0.45	0.47	0.52	0.14	other
0.28	0.20	0.45	0.47	0.52	0.14	other
0.28	0.20	0.44	0.46	0.51	0.14	other

The training is performed using 10 closest neighbors and Euclidean distance as metric.

Finally, the KNN classifier is used to evaluate the extended ranges pixels, the one with more pixels classified as energy line is considered the one with highest score and then considered the right path.

This “catheter” approach using the centered curve is used to flag if the energy line is allowed to grow in the extreme under processing and is used to avoid energy line border crossing over it (more on this soon). In positive case, the borders extend one time and then the catheter approach repeats in the other extreme. In negative case, the range M is reduced by five until it reaches zero, when it is confirmed that the border cannot more grow in the current extreme. In every iteration M has its original value restored. The borders extension is detailed in the next subsection.

3.2.7.2 Extending borders

As with the centered curve, two extended range candidates are built using straight line fitting of the last M points for each border. However, now the M is fixed to the value set by the catheter approach.

Similar to what was performed in the previous subsection, the borders receive possible offsets, linearly distributed. However, this time the offsets starts as well from -1/0/+1 to correct possible miss positioned borders and goes to -2/0/+2 respectively, as shown in FIGURE 96.

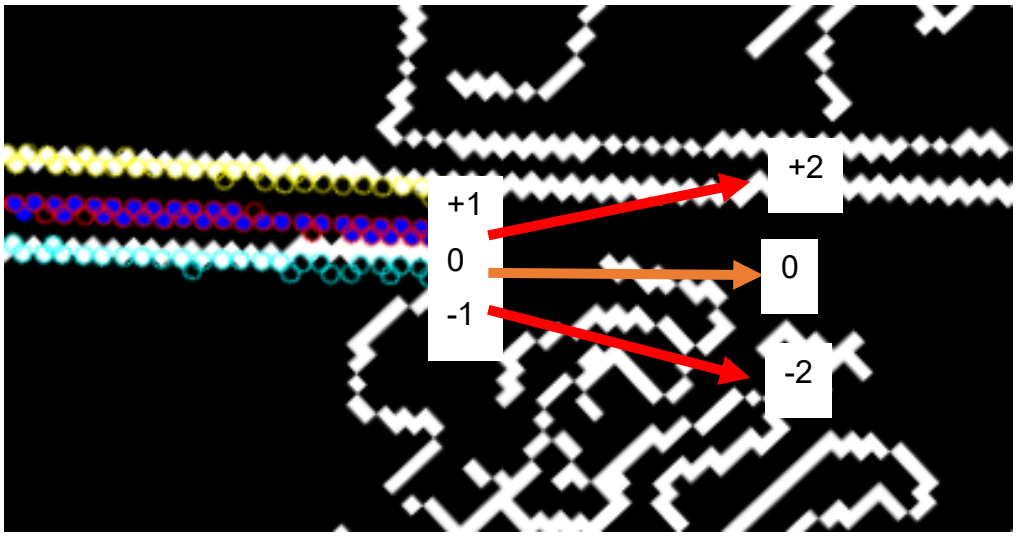


FIGURE 96 - Offsets used to extend the border.
SOURCE: The author (2016).

Among the commented possible borders position, one of each border are selected using the magnitude and the distance between them (thickness) in pixels as a metric.

The expected thickness in pixels for the extended range is obtained considering the real world thickness of the energy line and the depth at the extended range coordinates. The real world thickness is obtained by measuring the distance between all matching pixels of a given energy line, the depth at the center of the line and the focal length in pixels obtained during the calibration process. These measures a real world thickness RWT for every pair of parallel borders, therefore the real world value is considered the median of all measurements.

$$RWT = median\left(z_{centered}\frac{w}{f}\right) \quad (69)$$

where w is the Euclidean distance in pixels between each pixel pair, f is the focal length in pixels and $z_{centered}$ is the depth value measured using stereo at the center of the line.

As the real world thickness RWT is constant for an energy line, it must be the same even for the extended range. Therefore, using the equation 69 the other way around, it is possible to estimate the expected thickness in pixels for the extended range:

$$w_{predicted} = RWT \frac{f}{z_{centered}} \tag{70}$$

The magnitude score of each extended border is obtained as the mean of the magnitudes along the extended range filtered using median filter.

Each pair of possible borders receive a $score$ that is the sum of the individual magnitude scores weighed by the squared difference between the predicted width in pixels and the mean of the current widths defined as:

$$K = \begin{cases} 1, if \left(w_{precited} - w_{measured}\right)^2 \leq 1 \\ \dfrac{1}{\left(w_{precited} - w_{measured}\right)^2}, otherwise \end{cases} \tag{71}$$

$$score = K \left(\sum_{p_1 \in border_1} \nabla G(p_1) + \sum_{p_2 \in border_2} \nabla G(p_2) \right) \tag{72}$$

where $border_1$ and $border_2$ are extended range borders and ∇G is the magnitude image.

Notice that as the borders position get closer or afar from expected, K decreases quadratically, but as long as the deviations from the expected measurement are not bigger than one, only the magnitudes affect the score.

This weight aids the algorithm to find the right combination of borders even when there is a false border candidate with greater magnitude available.

Combinations of borders in which any of them passes through the centered curve are eliminated from the possible border pool. The remaining borders with the highest score are considered the real borders. Finally, the process repeats until no energy line is allowed to grow.

4 RESULTS

The results obtained using the presented method over the built stereo dataset were measured using error rates. The results is divided in two parts, one for 2D energy line detection and another one including the filtering steps using 3D methodology.

4.1 2D ENERGY LINES DETECTION RESULTS

To evaluate the detected lines using 2D energy line detection three different metrics are used: true positive rate (TPR), the mean of false positives (FP) per image and per detected energy line. The FP are objects classified as energy lines, when in fact they are not. The TPR is obtained as:

$$TPR = \frac{TP}{P} \tag{73}$$

where the TP (true positive) are the energy lines detected that are really an energy line, while P is the set of positive instances, therefore the number of real energy lines or:

$$P = TP + FN \tag{74}$$

where FN are real energy lines that were not detected.

An energy line is considered a correct detection if 90% of its pixels passes over the ground truth (even if partially), where the ground truth was defined manually and then, using morphological operations, were dilated 5 pixels to add some tolerance regarding errors with border locations. Multiple 2D segments over the same energy line are counted as a single TP.

TABLE 3 summarize the results obtained with 2D energy line detection over the stereo dataset described in subsection 3.1.2.

TABLE 3 – Results using 2D energy line detection

Image pair	P	TP	FP
1	3	3	121
2	3	3	135
3	3	3	135
4	2	2	279
5	3	3	79
6	3	3	99
7	2	2	237
8	2	2	236
9	3	3	28
10	3	3	54
11	2	2	190
12	3	3	99
13	3	3	124
14	3	2	155
15	3	3	84
16	3	2	132
17	3	3	126
18	3	3	186
19	3	3	216
20	3	3	181
21	3	3	160
22	3	3	162
23	3	3	150
24	3	3	212
25	2	1	236

SOURCE: The author (2016).

From TABLE 3 the TPR using only 2D methodology is:

$$TPR_{2D} = \frac{67}{70} \approx 95.7\% \tag{75}$$

Despite the good results of TPR_{2D}, the FP are excessively high. The false positive per image $FPPI$ is:

$$FPPI_{2D} = \frac{\sum_{i=1}^{n} FP_i}{n} \approx 152.6 \tag{76}$$

where n is the number of images.

And the false positives per energy line $FPPEL$ is:

$$FPPEL = \frac{\sum_{i=1}^{n} FP_i}{\sum_{i=1}^{n} TP_i} \approx 56.9 \quad (77)$$

where n is the number of images.

The $FPPEL$ of 56.9 means that for each energy line detected another 56.9 false energy lines are detected. Alternatively, using $FPPI$, a mean of 152.6 false energy lines per image. These poor results are mainly caused by diverse manmade structures that resemble an energy line, an example is the left view from the first image pair shown in FIGURE 97.

FIGURE 97 – Energy lines detected during 2D energy line detection. In yellow, branches; in orange, fence; in green, energy line like structure; in blue, wall texture; in purple, real energy lines.
SOURCE: The author (2016).

These results were refined using 3D filtering as explained in the next subsection.

4.2 3D FILTERING RESULTS

The adopted criteria to evaluate the results after 3D filtering is similar to the used by Song and Li (2014), that used false positives, false negatives, true negatives, and true positives. This is summarized by a confusion matrix, which is used to evaluate the performance of a classifier, contrasting the real class of a given object with the class assigned by the classifier (SAMMUT; WEBB, 2011). Here, the real instances are represented in columns, while the classified instances in rows. The confusion matrix is show in TABLE 4, where elements on

the top-left and bottom-right cells indicate correct decisions and elements on the bottom-right and top-right indicate wrong decisions.

TABLE 4 - Confusion matrix

Classification	Energy line present	Energy line absent
Energy line detected	True positive (TP)	False positive (FP)
Energy line not detected	False negative (FN)	True negative (TN)

SOURCE: Song and Li (2014).

The results of this step are based on the energy lines obtained in the 2D detection phased. Considering this, a true positive is an energy line detected during 2D stage that is really an energy line. A false positive is anything else detected as an energy line but that in fact it is not. The false negatives are real energy lines that were not detected or that were later removed with 3D filtering. Finally, the true negatives are energy lines detected during the 2D stage that were correctly removed later during 3D filtering stages.

The criteria used to consider that an energy line is a correct detection is the same one adopted during the presentation of the 2D results.

Before moving to the used metrics, it is necessary to define the sets P and N that are the positive instances and negative instances respectively, defined as:

$$\begin{aligned} P &= TP + FN \\ N &= TN + FP \end{aligned} \tag{78}$$

Due the intrinsic risks associated with energy lines, the consequences of missing one are much more severe than over detecting it, this was already respected by the results obtained using only 2D information. However, this came at the cost of detecting false energy lines, what was reduced with the 3D filtering. To evaluate the amount of FP removed with the 3D filtering, another ranking is used, false positive rate (FPR):

$$FPR = \frac{FP}{N} \tag{79}$$

To evaluate the overall performance of the classification process the accuracy metric is used:

$$accuracy = \frac{TP + TN}{P + N} \tag{80}$$

The results of the method with 3D filtering over the stereo dataset is shown in TABLE 5.

TABLE 5 - Results using 3D features

Image pair	P	N	TP	FP	TN
1	3	121	3	4	118
2	3	135	3	18	132
3	3	135	3	19	132
4	2	279	1	5	278
5	3	79	3	5	76
6	3	99	3	16	96
7	2	237	2	11	235
8	2	236	2	5	234
9	3	28	3	8	25
10	3	54	3	7	51
11	2	190	2	16	188
12	3	99	3	41	96
13	3	124	2	23	122
14	3	155	2	30	153
15	3	84	3	42	81
16	3	132	2	23	130
17	3	126	3	18	123
18	3	186	3	52	183
19	3	216	3	42	213
20	3	181	3	53	178
21	3	160	3	33	157
22	3	162	3	14	159
23	3	150	2	15	148
24	3	212	3	30	209
25	2	236	1	32	235

SOURCE: The author (2016).

From the results shown in TABLE 5 the FPR considering 3D filtering is:

$$FPR_{3D} = \frac{FP}{N} \approx 14.7\% \quad (81)$$

The TPR considering 3D filtering is:

$$TPR_{3D} = \frac{TP}{P} \approx 91{,}4\% \quad (82)$$

What resulted in an accuracy of 98%.

Clearly, the detection rate decreased when comparing TPR_{3D} with TPR_{2D}, but this happened in exchange for a much lower amount of FP. This becomes clear when recalculating $FPPI$ and $FPPEL$ for the 3D case as shown in the comparative TABLE 6.

TABLE 6 - Comparative of $FPPI$ and $FPPEL$ from 2D and 3D

	2D	3D	$\frac{3D}{2D}$
$FPPI$	152.6	22.3	6.8
$FPPEL$	56.9	8.7	6.5

SOURCE: The author (2016).

From TABLE 6 it is noticeable that the false positives from the 2D detection were reduced by a factor of about 7 after the 3D filtering.

In general, the FN cases happen in two situations. The first, when the energy line is not detected during 2D detection, caused by the lack of enough edge pixels. The second, when the stereo algorithm fails to calculate correctly the disparities, normally due repetitive patterns, what can happen when the energy lines are close to horizontal when viewed by the cameras. One example of each case of failure is shown in FIGURE 98 and FIGURE 99.

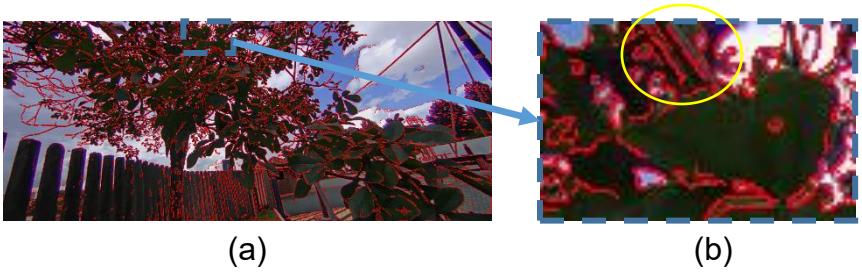


FIGURE 98 - Not enough edges to detect an energy line candidate. None of the CCs have enough pixels, in other word, more than $MinCCSize$.
SOURCE: The author (2016).

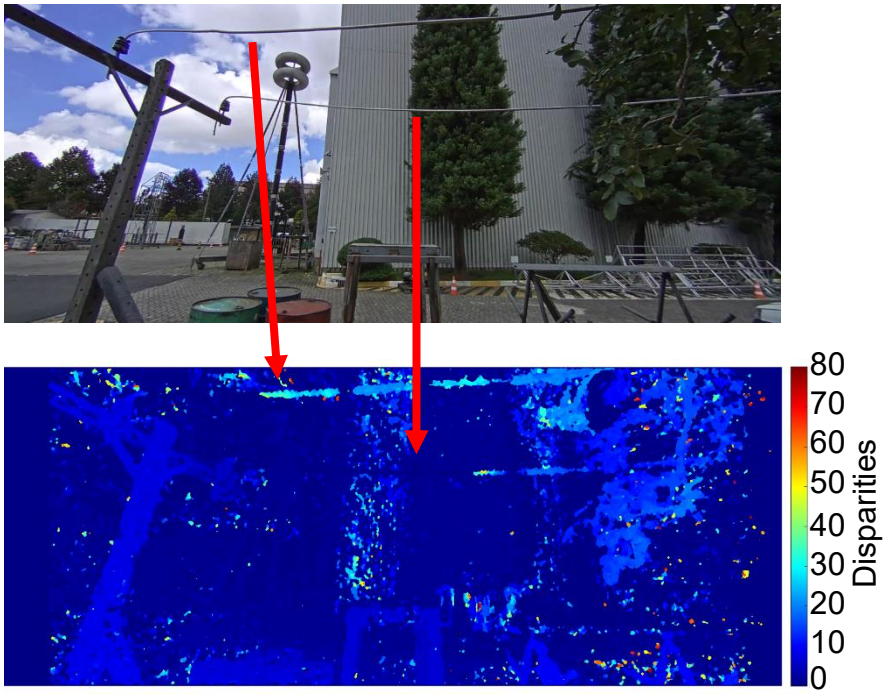


FIGURE 99 - Incorrect calculation of the energy lines disparities.
SOURCE: The author (2016).

During each run of the algorithm, the same parameters were used for all images, and using an AMD FX-8150 3.600 GHz with 16 GB of ram, the processing time for the detection of energy line in each image pair was about 100 seconds using Matlab 2015a. The rectification process using OpenCV can be considered real time, as it was possible to rectify and undistort even videos using the calculated rectification maps without any perceivable lag. A concise list with all the static parameters is shown in TABLE 7.

TABLE 7 - Parameters used to run the algorithm

Algorithm Step	Parameter	Value
Stereo camera calibration	$Q = \begin{bmatrix} 1 & 0 & 0 & -c_{xl} \\ 0 & 1 & 0 & -c_y \\ 0 & 0 & 0 & f \\ 0 & 0 & -\frac{1}{B} & \frac{(c_{xl} - c_{xr})}{B} \end{bmatrix}$	$\begin{bmatrix} 1 & 0 & 0 & -881.25 \\ 0 & 1 & 0 & -556.68 \\ 0 & 0 & 0 & 491.25 \\ 0 & 0 & 0.3342 & 0 \end{bmatrix}$
Feature extraction	$MinCCSize$	20
	x	0.7
	w	0.4
	$MaxDirectionChange$	$\frac{\pi}{9}$
2D energy line detection	k	7
	$maximum\ offSet$	20
	$MaxDiffCurvature$	0.2
3D energy line filtering	W_{min}	0.75
	W_{max}	2
Merging Energy lines	$MaxExtremeDiff$	$\frac{\pi}{18}$
Energy line growing	M	$\min\left(40, \frac{n}{2}\right)$, where n is the number of elements in each border

SOURCE: The author (2016).

After each run of developed algorithm, it is possible to create a 2D image with a colormap over the detected energy line to indicate the energy line depth (FIGURE 100) or a point cloud of the detected energy lines as shown in FIGURE 101.

FIGURE 100 - Color projected over each detected energy line.
SOURCE: The author (2016).

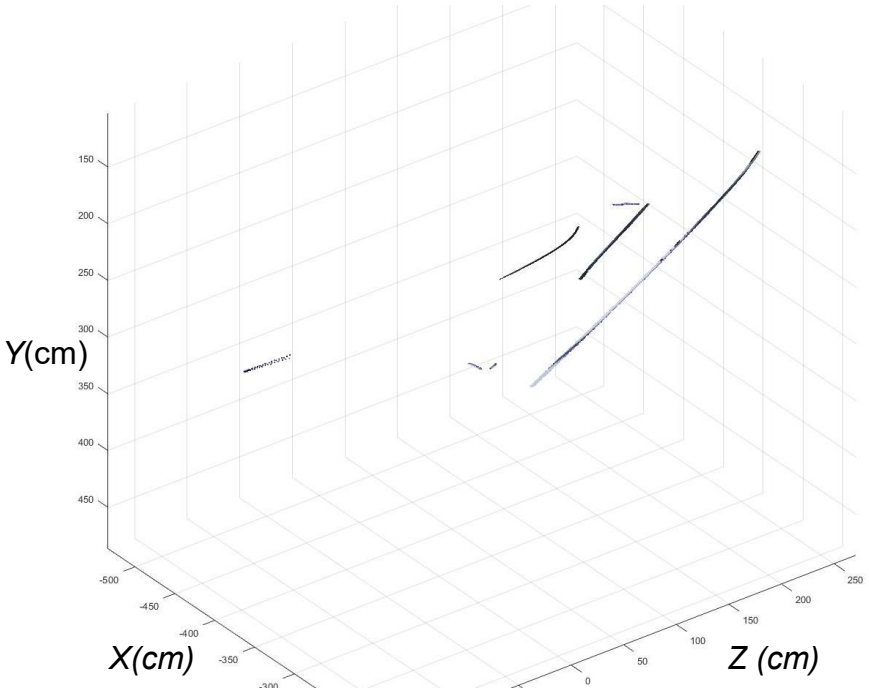


FIGURE 101 - Pointcloud of the detected energy lines from the first image pair.
SOURCE: The author (2016).

5 CONCLUSION

In this work, it was developed a method to detect urban close range energy lines and its 3D positioning in high resolution stereo image pairs. Due to the lack of available energy line datasets in urban environment with stereo information it was not possible to compare the presented method with the current literature. As a consequence, a data set was created using two parallel cameras and will be publicly available. The presented method showed how helpful 3D features might be to decrease false positives, using them as a powerful ally to the common 2D features used along image processing algorithms. The 3D features showed great importance mainly in urban scenario due to the great amount of objects that might resemble an energy line.

Enhancement in image quality and color space transformation played an important role due to the lack of control of outdoor illumination. As the detection of energy lines occurred using geometric features of specific edges, automatic thresholding was necessary as well, to ensure that the technique will work in most of the cases. For the first time, the use of multiple cameras was explored to improve the accuracy of urban energy line detection. With depth information, it was possible to decrease about seven times the amount of false positives. As a result, the method generates color maps with the detected lines indicating its depth information and 3D point clouds as well.

As the proposed method lay its foundations on detected edges, improved methods of edge detection, artificial illumination and high contrast cameras with low noise sensors can be evaluated in future works in order to check the impact in the results.

Another feature not explored in this work is the use of temporal analysis, processing sequential frames, instead of static frames, in order to check if it aids to get stable results in video streams.

As the proposed algorithm was designed as a prototype to evaluate the applicability of computer vision to detect overhead urban energy lines close to vegetation, it does not accomplish real time requirements. Therefore, there is still room for research concerning processing time, where the study to convert the method to parallel architectures, such as graphic processing units (GPUs) or field programmable gate arrays (FPGAs), can increase significantly its applicability.

Regarding the 3D features obtained with stereo vision, future research can evaluate as well what improvements can be achieved by using different stereo matching algorithms, different stereo rigs with different base lines or cameras. Furthermore, other technologies to measure 3D features might be evaluated as well, such as structured light or light detection and ranging (LIDAR), which could constrain even more the range of detected objects, possibly reducing the false positives.

Finally, it is expected that the results obtained in this work will be applied in future studies to aid the development of a system applicable to solve real world problems.

REFERENCES

ADRIAN, R. J.; WESTERWEEL, J. **Particle Image Velocimetry**. book, New York, NY, USA: Cambridge University Press, 2011.

ALEXANDERSON, G. L. About the Cover: Euler and Königsberg's Bridges: a Historical View. **Bulletin (New Series) of the American Mathematical Society**, v. 43, n. 4, p. 567–573, 2006.

ALLEN, E.; TRIANTAPHILLIDOU, S. **The Manual of Photography**. book, Taylor & Francis, 2012.

ARACIL, R.; FERRE, M.; HERNANDO, M.; PINTO, E.; SEBASTIAN, J. M. Telerobotic system for live-power line maintenance: ROBTET. **Control Engineering Practice**, v. 10, n. 11, p. 1271–1281, 2002.

AZIZ, S. Development and verification of ground-based tele-robotics operations concept for Dextre. **Acta Astronautica**, v. 86, p. 1–9, 2013. article, .

BANSAL, A.; KOWDLE, A.; PARIKH, D.; GALLAGHER, A.; ZITNICK, L. Which Edges Matter? 2013 IEEE International Conference on Computer Vision Workshops (ICCVW). **Proceedings...** . p.578–585, 2013. inproceedings, Sydney, NSW, Australia.

BLEYER, M.; GELAUTZ, M. A layered stereo matching algorithm using image segmentation and global visibility constraints. **ISPRS Journal of Photogrammetry and Remote Sensing**, v. 59, n. 3, p. 128–150, 2005. article, .

BLEYER, M.; GELAUTZ, M. Simple but Effective Tree Structures for Dynamic Programming-based Stereo Matching. International Conference on Computer Vision Theory and Applications (VISAPP). **Proceedings...** . p.415–422, 2008. inproceedings, Funchal, Madeira, Portugal.

BONDY, J. A.; MURTY, U. S. R. **Graph Theory**. 1º ed. London: Springer-Verlag London, 2008.

BOUGUET, J.-Y. Complete Camera Calibration Toolbox for Matlab. **Jean-Yves Bouguet's Homepage**, 1999.

BOYKOV, Y.; VEKSLER, O.; ZABIH, R. Fast approximate energy minimization via graph cuts. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, v. 23, n. 11, p. 1222–1239, 2001. article, .

BRADSKI, G.; KAEHLER, A. **Learning OpenCV: Computer Vision with the OpenCV Library**. 1st ed. Sebastopol: O'Reilly Media Inc., 2008.

BRAHMBHATT, S. **Practical OpenCV**. 1st ed. book, New York, NY, USA: Apress, 2013.

BROWN, D. C. Close-range camera calibration. **Photogrammetric Engineering**, v. 37, n. 8, p. 855–866, 1971.

BURGNER, J.; SWANEY, P. J.; RUCKER, D. C.; et al. A bimanual teleoperated system for endonasal skull base surgery. 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems. **Proceedings...** . p.2517–2523, 2011. inproceedings, .

BURKE, M. W. **Image Acquisition: Handbook of machine vision engineering**.

1st ed. book, London, UK: Chapman & Hall, 1996.

CANDAMO, J.; GOLDGOF, D. Wire detection in low-altitude, urban, and low-quality video frames. 19th International Conference on Pattern Recognition, 2008. **Proceedings...** . p.1–4, 2008. inproceedings, Tampa, FL, USA.

CANDAMO, J.; KASTURI, R.; GOLDGOF, D.; SARKAR, S. Detection of Thin Lines using Low-Quality Video from Low-Altitude Aircraft in Urban Settings. **IEEE Transactions on Aerospace and Electronic Systems,** v. 45, n. 3, p. 937–949, 2009. article, .

CANNY, J. A Computational Approach to Edge Detection. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, v. PAMI-8, n. 6, p. 679–698, 1986.

CAO, W.; YANG, X. Power Line Detection Based on Symmetric Partial Derivative Distribution Prior. IEEE International Conference on Information and Automation (ICIA). **Proceedings...** . p.767–772, 2013. Yinchuan, China.

CHEN, H.-H.; CHUANG, W.-N.; WANG, C.-C. Vision-based line detection for underwater inspection of breakwater construction using an ROV. **Ocean Engineering**, v. 109, p. 20–33, 2015. Elsevier.

CHEN, W.-K. **Graph Theory and Its Engineering Applications**. book, Singapore: World Scientific, 1997.

CHEN, Y.; LI, Y.; ZHANG, H.; et al. Automatic power line extraction from high resolution remote sensing imagery based on an improved Radon transform. **Pattern Recognition**, v. 49, p. 174–186, 2016.

CLAUS, D. **High accuracy metrology using low-resolution cameras**. PhD Thesis - Department of Engineering Science, University of Oxford, Oxford, UK, 2007.

CORKE, P. **Robotics, Vision and Control: Fundamental Algorithms in MATLAB**. book, Berlin, Germany: Springer, 2011.

CORMEN, T. H. **Introduction to Algorithms**. 3rd ed. book, United States of America: MIT Press, 2009.

CYGANEK, B.; SIEBERT, J. P. **An Introduction to 3D Computer Vision Techniques and Algorithms**. book, Chichester, UK: John Wiley & Sons, 2009.

DANIELSSON, P.-E.; SEGER, O. Generalized and Separable Sobel Operators. In: H. Freeman (Org.); **Machine Vision for Three-Dimensional Scenes**. p.347–379, 1990. incollection, Academic Press.

DAWSON-HOWE, K. **A Practical Introduction to Computer Vision with OpenCV**. 1st ed. book, Chichester,: John Wiley & Sons, 2014.

DIESTEL, R. **Graph Theory**. 3rd ed. book, Berlin, Germany: Springer Science & Business Media, 2006.

DOYLE, W. Operations Useful for Similarity-Invariant Pattern Recognition. **Journal of the ACM (JACM)**, v. 9, n. 2, p. 259–267, 1962. article, New York, NY, USA: ACM.

DUDA, R. O.; HART, P. . E.; MUNSON, J. H. **Graphical data-processing research study and experimental investigation**. 1967.

DUFF, E.; CARIS, C.; BONCHIS, A.; et al. The Development of a Telerobotic

Rock Breaker. In: A. Howard; K. Iagnemma; A. Kelly (Orgs.); **Field and Service Robotics: Results of the 7th International Conference**. p.411–420, 2010. inbook, Berlin, Heidelberg: Springer.

EULER, L. Solutio problematis ad geometriam situs pertinentis. **Commentarii academiae scientiarum Petropolitanae 8**, p. 128–140, 1741.

FANTONI, A.; FERREIRA, A. J.; BELMAMOUN, A.; HASSOUNI, M. EL; HAMMOUCH, A. Conference on Electronics, Telecommunications and Computers – CETC 2013. On Selection and Combination of Relevant Color Components for Edge Detection. Procedia Technology. **Proceedings...** . v. 17, p.764–771, 2014. article, .

FAUGERAS, O. **Three-dimensional computer vision: a geometric viewpoint**. Massachusetts: The MIT Press, 1993.

FELZENSZWALB, P. F.; HUTTENLOCHER, D. P. Efficient graph-based image segmentation. **International Journal of Computer Vision**, v. 59, n. 2, p. 167–181, 2004.

GONZALEZ, R. C.; WOODS, R. E. **Digital Image Processing**. New Jersey, USA: Prentice Hall, 2008.

GRIMSON, W. E. L. Computational Experiments with a Feature Based Stereo Algorithm. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, v. PAMI-7, n. 1, p. 17–34, 1985. article, .

GROMPONE VON GIOI, R.; JAKUBOWICZ, J.; MOREL, J. M.; RANDALL, G. LSD: A fast line segment detector with a false detection control. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, v. 32, n. 4, p. 722–732, 2010.

GUTHART, G. S.; SALISBURY, J. K. The IntuitiveTM Telesurgery System : Overview and Application. Proceedings of IEEE International Conference on Robotics & Automation. **Proceedings...** . p.618–621, 2000. San Francisco, CA, USA.

GUY, N. K. **The Lens: A Practical Guide for the Creative Photographer**. 1º ed. book, Santa Barbara, CA, USA: Rocky Nook, 2012.

HARPER, D. Online Etymology Dictionary. Available at: <http://dictionary.reference.com/browse/binocular>. Acesso em: 9/2/2016.

HARTLEY, R. I. Theory and Practice of Projective Rectification. **International Journal of Computer Vision**, v. 35, n. 2, p. 115–127, 1999. article, .

HARTLEY, R.; ZISSERMAN, A. **Multiple View Geometry in Computer Vision**. book, New York, NY, USA: Cambridge University Press, 2004.

HAWICK, K. A.; LEIST, A.; PLAYNE, D. P. Parallel graph component labelling with GPUs and CUDA. **Parallel Computing**, v. 36, n. 12, p. 655–678, 2010. article, .

HEIKKILA, J.; SILVEN, O. A four-step camera calibration procedure with implicit image correction. Proceedings of IEEE Conference on Computer Vision and Pattern Recognition. **Proceedings...** . p.1106–1112, 1997. inproceedings, San Juan, Puerto Rico.

HIRSCHMULLER, H.; SCHARSTEIN, D. Evaluation of Cost Functions for Stereo Matching. **IEEE Conference on Computer Vision and Pattern Recognition**,

2007. inproceedings, Minneapolis, USA.

HOPKINS, B.; WILSON, R. J. The Truth about Königsberg. **The Mathematical Association of America**, v. 35, n. 3, p. 198–207, 2004.

HOWARD, I. P.; ROGERS, B. J. **Binocular Vision and Stereopsis**. book, New York, NY, USA: Oxford University Press, 1995.

HUBEL, D. H.; WIESEL, T. N. Receptive fields of single neurones in the cat's striate cortex. **Journal of Physiology**, v. 148, p. 574–591, 1959.

ITSEEZ. OpenCV. Available at: <http://opencv.org/>. .

JOHNSTON, C. T.; BAILEY, D. G. FPGA implementation of a Single Pass Connected Components Algorithm. 4th IEEE International Symposium on Electronic Design, Test and Applications. **Proceedings...** . p.228–231, 2008. inproceedings, Hong Kong.

JOINT ISO/CIE STANDARD. **CIE Colorimetry - Part 2: Standard Illuminants for Colorimetry**. Vienna, Austria, 2006.

JOINT ISO/CIE STANDARD. **Colorimetry -- Part 4: CIE 1976 L*a*b* Colour space**. Vienna, Austria, 2008.

KALENTEV, O.; RAI, A.; KEMNITZ, S.; SCHNEIDER, R. Connected component labeling on a 2D grid using CUDA. **Journal of Parallel and Distributed Computing**, v. 71, n. 4, p. 615–620, 2011. article, .

KAUFMAN, M. **Mars Landing 2012: Inside the NASA Curiosity Mission**. book, Washington, D.C., USA: National Geographic Society, 2012.

KONG, T. Y.; ROSENFELD, A. **Topological Algorithms for Digital Image Processing**. book, Elsevier Science, 1996.

KOVESI, P. Phase congruency detects corners and edges. The Australian Pattern Recognition Society Conference: DICTA 2003. **Proceedings...** . p.309–18, 2003. Sydney, NSW, Australia.

LI, Z.; LIU, Y.; WALKER, R.; HAYWARD, R.; ZHANG, J. Towards automatic power line detection for a UAV surveillance system using pulse coupled neural filter and an improved Hough transform. **Machine Vision and Applications**, v. 21, n. 5, p. 677–686, 2009.

LOOP, C.; ZHANG, Z. Computing rectifying homographies for stereo vision. **IEEE Computer Society Conference on Computer Vision and Pattern Recognition**, 1999. CONF, Fort Collins, CO, USA.

LUKAC, R.; PLATANIOTIS, K. N. **Color Image Processing: Methods and Applications**. book, Boca Raton, FL, USA: CRC Press, 2006.

LUO, B.; HANCOCK, E. R. Structural graph matching using the EM algorithm and singular value decomposition. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, v. 23, n. 10, p. 1120–1136, 2001. article, .

LUO, X.; ZHANG, J.; CAO, X.; YAN, P.; LI, X. Object-aware power line detection using color and near-infrared images. **IEEE Transactions on Aerospace and Electronic Systems**, v. 50, n. 2, p. 1374–1389, 2014. article, .

MANOHAR, M.; RAMAPRIYAN, H. K. Connected component labeling of binary images on a mesh connected massively parallel processor. **Computer Vision, Graphics, and Image Processing**, v. 45, n. 2, p. 133–149, 1989. article, .

MARR, D.; HILDRETH, E. Theory of Edge Detection. **Proceedings of the Royal Society of London. Series B, Biological Sciences**, v. 207, n. 1167, p. 187–217, 1980.

MARTINS, D. A.; SILVA, G. T.; SALAMANCA, H. L.; et al. **Simulações e medições de campo elétrico em redes de 13,8 kV**. Curitiba, PR, Brazil, .

MATTOCCIA, S. Stereo Vision: Algorithms and Applications. Available at: <http://www.vision.deis.unibo.it/smatt/Seminars/StereoVision.pdf>. Acesso em: 18/2/2016.

MERIAN-ERBEN. Preussen Chronik Eines Deutschen Staates. Available at: <http://www.preussenchronik.de/bild_jsp/key=bild_kathe2.html>. Acesso em: 23/11/2015.

MIRALLÈS, F.; POULIOT, N.; MONTAMBAULT, S.; IREQ, H. State-of-the-Art Review of Computer Vision for the Management of Power Transmission Lines. **3rd International Conference on Applied Robotics for the Power Industry (CARPI)**, 2014. Foz do Iguassu, PR, Brazil.

MORRONE, M. C.; OWENS, R. A. Feature detection from local energy. **Pattern Recognition Letters**, v. 6, n. 5, p. 303–313, 1987.

NIXON, M.; AGUADO, A. **Feature Extraction & Image Processing for Computer Vision, Third Edition**. New York, NY, USA: Academic Press, 2012.

OHTA, Y.; KANADE, T. Stereo by Intra- and Inter-Scanline Search Using Dynamic Programming. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, v. PAMI-7, n. 2, p. 139–154, 1985. article, .

ORTIZ, A.; ANTICH, J.; OLIVER, G. A particle filter-based approach for tracking undersea narrow telecommunication cables. **Machine Vision and Applications**, v. 22, n. 2, p. 283–302, 2009.

PARKER, J. R. **Algorithms for Image Processing and Computer Vision**. book, Indianapolis, IN, USA: Wiley, 2010.

PETROU, M.; KITTLER, J. Optimal edge detectors for ramp edges. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, v. 13, n. 5, p. 483–491, 1991. article, .

PICCIGALLO, M.; SCARFOGLIERO, U.; QUAGLIA, C.; et al. Design of a Novel Bimanual Robotic System for Single-Port Laparoscopy. **IEEE/ASME Transactions on Mechatronics**, v. 15, n. 6, p. 871–878, 2010. article, .

PREWITT, J. Object enhancement and extraction. **Picture processing and Psychopictorics**. v. 10, p.15–19, 1970.

PRINCE, S. J. D. **Computer Vision: Models, Learning, and Inference**. book, New York, NY, USA: Cambridge University Press, 2012.

RAO, G. S. **Mathematical Foundations of Computer Science**. book, New Delhi, India: I.K. International Publishing House Pvt. Limited, 2006.

ROBERTS, L. G. **Machine perception of three-dimensional solids**, 1963. Massachusetts Institute of Technology.

ROBLES-KELLY, A.; HANCOCK, E. R. Graph edit distance from spectral seriation. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, v. 27, n. 3, p. 365–378, 2005. article, .

ROSENFELD, A. Connectivity in Digital Pictures. **Journal of the ACM**, v. 17, n. 1, p. 146–160, 1970. article, New York, NY, USA: ACM.

ROSENFELD, A.; PFALTZ, J. L. Sequential Operations in Digital Picture Processing. **Journal of the ACM**, v. 13, n. 4, p. 471–494, 1966.

RUSS, J. C. **The Image Processing Handbook, Sixth Edition**. 6º ed. book, Boca Raton: CRC Press, 2016.

SAMMUT, C.; WEBB, G. I. **Encyclopedia of Machine Learning**. book, New York: Springer US, 2011.

SCHARSTEIN, D.; SZELISKI, R. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. **International Journal of Computer Vision**, v. 47, n. 1–3, p. 7–42, 2002.

SHI, J.; MALIK, J. Normalized cuts and image segmentation. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, v. 22, n. 8, p. 888–905, 2000. article, .

SICILIANO, B.; KHATIB, O. **Springer Handbook of Robotics**. book, Berlin, Germany: Springer, 2008.

SIEBERT, L. C.; BIANCHI FILHO, J. F.; CAREGNATO NETO, Â.; et al. Teleoperação de um sistema robotizado para poda de árvores na proximidade de redes energizadas. **XII Simpósio Brasileiro de Automação Inteligente (SBAI)**, 2015. Natal, RN, Brazil (in Portuguese).

SIEBERT, L. C.; TOLEDO, L. F. R. B.; BLOCK, P. A. B.; et al. A survey of applied robotics for tree pruning near overhead power lines. **International Conference on Applied Robotics for the Power Industry (CARPI)**, 2014. inproceedings, Foz Iguassu, PR, Brazil.

SIMPSON, P.; BOSSUYT, R. VAN. Tree-caused electric outages. **Journal of Arboriculture**, v. 22, n. 3, p. 117–121, 1996.

SONG, B.; LI, X. Power line detection from optical images. **Neurocomputing**, v. 129, p. 350–361, 2014.

SONKA, M.; HLAVAC, V.; BOYLE, R. **Image Processing, Analysis, and Machine Vision**. book, Stamford, CT, USA: Cengage Learning, 2014.

SPACEK, L. A. Edge detection and motion detection. **Image and Vision Computing**, v. 4, n. 1, p. 43–56, 1986. article, .

SUN, J.; ZHENG, N.-N.; SHUM, H.-Y. Stereo matching using belief propagation. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, v. 25, n. 7, p. 787–800, 2003. article, .

SUZUKI, K.; HORIBA, I.; SUGIE, N. Linear-time connected-component labeling based on sequential local operations. **Computer Vision and Image Understanding**, v. 89, n. 1, p. 1–23, 2003. article, .

SZELISKI, R. **Computer Vision : Algorithms and Applications**. London, UK: Springer Science & Business Media, 2010.

THE MATHWORKS INC. MATLAB Version R2015a. Available at: <http://www.mathworks.com/>. .

TORSELLO, A.; HANCOCK, E. R. Computing approximate tree edit distance using relaxation labeling. **Pattern Recognition Letters**, v. 24, n. 8, p. 1089–

1097, 2003. article, .

TSAI, R. Y. A Versatile Camera Calibration Technique for High-Accuracy 3D Machine Vision Metrology Using Off-the-Shelf TV Cameras and Lenses. **IEEE Journal on Robotics and Automation**, v. 3, n. 4, p. 323–344, 1987.

VENTURI, J. J. **Cônicas e quádricas**. 5th ed. Curitiba, PR, Brazil: Artes Gráficas e Editora Unificado, .

WALKER, B. H. **Optical Engineering Fundamentals**. book, Bellingham, WA, USA: SPIE Optical Engineering Press, 1995.

WALTHER, D. B.; CHAI, B.; CADDIGAN, E.; BECK, D. M.; FEI-FEI, L. Simple line drawings suffice for functional MRI decoding of natural scene categories. **Proceedings of the National Academy of Sciences of the United States of America**, v. 108, n. 23, p. 9661–9666, 2011.

WEISSTEIN, E. W. “Least Squares Fitting--Exponential.” **From MathWorld--A Wolfram Web Resource.**, 1999.

WENG, J.; COHEN, P.; HERNIOU, M. Camera calibration with distortion models and accuracy evaluation. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, v. 14, n. 10, p. 965–980, 1992.

WEST, D. B. **Introduction to Graph Theory**. 2nd ed. book, Prentice Hall, 2001.

WU, Z.; LEAHY, R. An optimal graph theoretic approach to data clustering: theory and its application to image segmentation. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, v. 15, n. 11, p. 1101–1113, 1993. article, .

YANG, C.-C. Improving the sharpness of a non-uniformly illuminated image by an integral mask-filtering approach. **Optik - International Journal for Light and Electron Optics**, v. 124, n. 21, p. 5049–5051, 2013. article, .

YANG, T. W.; YIN, H.; RUAN, Q. Q. Overhead Power Line Detection from UAV Video Images. , p. 28–30, 2012.

ZEKI, S. **A vision of the Brain**. Oxford, UK: Blackwell Scientific Publications, 1993.

ZHANG, J.; LIU, L.; WANG, B.; et al. High Speed Automatic Power Line Detection and Tracking for a UAV-Based Inspection. 2012 International Conference on Industrial Control and Electronics Engineering. **Proceedings...** . p.266–269, 2012. Xi’an, China.

ZHANG, Z. Flexible camera calibration by viewing a plane from unknown orientations. The Proceedings of the 7th IEEE International Conference on Computer Vision. **Proceedings...** . v. 1, p.666–673 vol.1, 1999. inproceedings, Kerkyra, Greece.