

Aalto University  
School of Electrical Engineering  
Master's Programme in ICT Innovation

Jacopo Losi

# Structured light assisted real time stereo photogrammetry for robotics and automation

## Novel implementation of stereo matching

Master's Thesis  
Espoo, June 15, 2020

DRAFT! — June 15, 2020 — DRAFT!

Supervisors: Professor Juho Kannala, Aalto University  
Professor Daniele Fontanelli, University of Trento  
Professor Nicola Conci, University of Trento  
Advisor: Sami Ruuskanen M.Sc. (Tech.)

<b>Author:</b>	Jacopo Losi					
<b>Title:</b>						
	Structured light assisted real time stereo photogrammetry for robotics and automation Novel implementation of stereo matching					
<b>Date:</b>	June 15, 2020	<b>Pages:</b>	vi + 71			
<b>Major:</b>	Autonomous Systems	<b>Code:</b>	ELEC3055			
<b>Supervisors:</b>	Professor Juho Kannala Professor Daniele Fontanelli Professor Nicola Conci					
<b>Advisor:</b>	Sami Ruuskanen M.Sc. (Tech.)					
The abstract provides goal, motivation, background, and conclusions of the work. It has to fit to one page together with the bibliographical information.						
If the thesis is in English and the language of school education is Finnish or Swedish, the abstract is written in English and in Finnish or in Swedish. If the language of school education is other than Finnish or Swedish, the abstract is written in English only.						
The thesis example file ( <code>thesis-example.tex</code> ), all the chapter content files ( <code>1introduction.tex</code> and so on), and the Aalto style file ( <code>aalto-thesis.sty</code> ) are commented with explanations on how the Aalto thesis works. The files also contain some examples on how to customize various details of the thesis layout, and of course the example text works as an example in itself. Please read the comments and the example text; that should get you well on your way!						
In the thesis template, you can find the text of the abstract in the abstract in the <code>thesis-example.tex</code> file, together with the bibliographical information of the abstract tables. <b>FIXME This is an example how to use fixme: add your abstract here.</b> <b>FIXME!</b> Fixme is a command that helps you identify parts of your thesis that still require some work. When compiled in the custom <code>mydraft</code> mode, text parts tagged with fixmes are shown in bold and with fixme tags around them. When compiled in normal mode, the fixme-tagged text is shown normally (without special formatting). The draft mode also causes the “Draft” text to appear on the front page, alongside with the document compilation date. The custom <code>mydraft</code> mode is selected by the <code>mydraft</code> option given for the package <code>aalto-thesis</code> , near the top of the <code>thesis-example.tex</code> file.						
<b>Keywords:</b>	stereo vision; matching cost; census transform; hamming distance; binary pattern; semi-global matching					
<b>Language:</b>	English					

# Acknowledgements

I wish to thank all students who use L<sup>A</sup>T<sub>E</sub>X for formatting their theses, because theses formattted with L<sup>A</sup>T<sub>E</sub>X are just so nice.

Thank you, and keep up the good work!

Espoo, June 15, 2020

Jacopo Losi

# Abbreviations and Acronyms

2k/4k/8k mode	COFDM operation modes
3GPP	3rd Generation Partnership Project
ESP	Encapsulating Security Payload; An IPsec security protocol
FLUTE	The File Delivery over Unidirectional Transport protocol
e.g.	for example (do not list here this kind of common acronyms or abbreviations, but only those that are essential for understanding the content of your thesis.
note	Note also, that this list is not compulsory, and should be omitted if you have only few abbreviations

# Contents

<b>Abbreviations and Acronyms</b>	<b>iv</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Problem statement . . . . .	1
1.1.1 Stereo geometry . . . . .	3
1.1.2 Rectification . . . . .	4
1.1.3 Stereo methods and dense correspondence . . . . .	5
1.2 Structure of the Thesis . . . . .	9
<b>2 Theoretical Background and Related Work</b>	<b>10</b>
2.1 Epipolar geometry and Rectification . . . . .	10
2.1.1 Triangulation . . . . .	10
2.1.2 Epipolar geometry . . . . .	11
2.1.3 Rectification . . . . .	13
2.2 Stereo methods and dense correspondence . . . . .	14
2.2.1 Stereo geometry based methods . . . . .	14
2.2.2 Deep learning based methods . . . . .	14
2.2.2.1 Confidence measures . . . . .	17
2.3 Image processing techniques . . . . .	19
2.3.1 Point operators . . . . .	19
2.3.2 Neighborhood operator . . . . .	20
2.4 Edge detection and segmentation algorithms . . . . .	24
2.5 Data and estimations error types . . . . .	28
<b>3 Environment</b>	<b>29</b>
3.1 Middlebury Dataset . . . . .	29
3.2 Stereo Photogrammetry Ladimo device . . . . .	32
3.3 Software Environment . . . . .	34
3.3.1 MATLAB computing environment . . . . .	35
3.3.2 Microsoft Visual Studio integrated development environment . . . . .	36

3.3.3	C++ programming language . . . . .	36
3.3.4	OpenCV cross-platform library . . . . .	37
3.3.5	CloudCompare 3D editing and processing software . . . . .	38
<b>4</b>	<b>Methods</b>	<b>39</b>
4.1	Standard disparity estimation methods . . . . .	39
4.1.1	Census transform . . . . .	41
4.1.2	Rank Transform . . . . .	43
4.1.3	Conventional intensity based methods . . . . .	44
4.2	Deep-learning based methods . . . . .	44
4.3	Pure derivative based data estimation algorithm . . . . .	45
4.4	SGM based data estimation algorithm . . . . .	47
4.5	Pre-processing techniques . . . . .	48
4.6	Post-processing techniques . . . . .	49
<b>5</b>	<b>Implementation</b>	<b>50</b>
5.1	Preparatory and testing phase on MATLAB . . . . .	50
5.2	Derivative based method implementation . . . . .	53
5.2.1	General features of the method and simulated grid specifications . . . . .	53
5.2.2	Edge cases and penalty values discussion . . . . .	57
5.2.3	Derivative computation and sanity check . . . . .	58
5.3	Semi-Global Matching based method implementation . . . . .	59
5.4	OpenCV built-in Semi-Global Box Matching algorithm . . . . .	60
5.5	Pre-processing testing and analysis . . . . .	60
5.5.1	Look-up table and point grid pre-processing . . . . .	61
5.5.2	Image undistortion and rectification . . . . .	61
5.6	Post-processing testing and analysis . . . . .	61
5.7	Preparation to future improvements . . . . .	61
<b>6</b>	<b>Evaluation</b>	<b>62</b>
<b>7</b>	<b>Discussion</b>	<b>63</b>
<b>8</b>	<b>Conclusions</b>	<b>64</b>
<b>A</b>	<b>First appendix</b>	<b>71</b>

# Chapter 1

## Introduction

### 1.1 Problem statement

Dense and accurate disparity maps are the key factor for obtaining correct depth estimations for many computer vision applications such as autonomous driving, 3D reconstruction, object detection and robotics. Therefore, stereo matching and disparity estimation can be identified as fundamental problems in the current developments of computer vision [1].

Multiple methods for disparity estimation has been developed for many years [1]. Specifically, older strategies are focused on local-based or global-based methods. On the contrary, deep learning based strategies applied to local or global methods has been recently proposed. The latter approach aims to a precise local correspondence exploiting deep learning and applying Semi-global matching (SGM) as the regularization step of the pipeline. Therefore, deep learning techniques such as FlowNet and DispNet [1] are used as the end-to-end part of the pipeline. According to the current benchmark database ranks for stereo matching algorithms, e.g. the one published in the KITTI website, the state of the art implementations are based on deep learning methods. However, these strategies lack in accuracy compared to the standard pipelines. This is probably due to the difference between real environment and the training database as underlined in [1] [2].

As aforementioned, the state of the art methods to recover dense disparity maps from stereo pairs are focused on deep convolutional neural networks trained end-to-end [3]. Most of these techniques, which will be subsequently described, exploit as regularization phase the Semi-global matching (SGM) method. Actually, among local and global approaches, the Hirschmuller's algorithm [4] appears to be the best performing in terms of computational cost and accuracy. For this reason, it is the preferred trade-off for most real

time applications.

Considering the multiple algorithm for stereo correspondence, they can be conventionally classified [5] into two general categories, local and global approaches. Specifically, the local-based methods tend to estimate the disparity image through a comparison of the matching cost from left and right views of the scene. In order to recover from low accuracy proper of the previous strategy, global-based methods try to calculate the disparity values by minimizing an energy function. In this context, Semi-Global Matching (SGM) combines strong factors of global and local approaches allowing to obtain a good trade-off between computational cost and accuracy.

Technically speaking, SGM applies a pixelwise, Mutual Information (MI) based matching cost for analysing pixel intensity value differences of input images [4]. Moreover, pixelwise matching is enhanced with a smoothness constraint, which leads to a global cost function. Then, post-processing techniques are applied to remove outliers and filter the image.

Referring to the analysis performed by Scharstein and Szeliski [5], SGM carries out four main steps, as well as most of the stereo matching algorithms. These are defined as: matching cost computation, cost aggregation, disparity computation and disparity refinement.

Considering the former, it is usually based on absolute, squared or sampling insensitive difference between pixel intensities [4]. Although those methods allow to reach a reliable accuracy, they are sensitive to radiometric difference. Thus, cost based on image gradients or window-based methods, such as rank and census transform [6], became an optimal choice. Furthermore, Mutual Information results as a good trade-off for dealing with complex radiometric relationships between images.

In the second phase, cost aggregation collects the matching costs considering multiple directions and the disparity levels. Following, disparity evaluation is defined for each pixel, as the one with the lowest cost. This is the approach typically used for local methods. Global algorithms, rather, used to get rid of the aggregation step and define a global energy function. Over that function, pixel similarity and disparity smoothness are enforced with different strategies. In this latter case, the best disparity is identified finding the minimum of the cost function. This is achieved with multiple techniques such as: Dynamic Programming (DP) [7], Belief Propagation [8] or Graph Cuts [9].

Disparity refinement tends to differ more among the different methods. Usually, post-processing techniques such as filtering, outlier removal and consistency check are in general applied.

As anticipated above, among the top-ranked stereo matching algorithms, SGM results to be the best performing in terms of computational time and

accuracy. Its benefits stand in the hierarchical computation of the matching cost, which exploit Mutual Information. Cost aggregation is achieved taking into account a global energy function and a pathwise pixel optimization. The final disparity is chosen with a winner takes all strategy. Disparity refinement is completed by consistency check between left and right disparity images. Besides the challenge of building up the optimal algorithm for recovering a disparity image from a stereo image pair, it is necessary to develop an analysis of the basis of stereo correspondence and its importance for multiple applications such as: autonomous driving, robotics, object detection and 3D reconstruction.

First of all, stereo matching is defined as the process of estimating a 3D model of a scene, starting from two or more images. Therefore, the matching pixel between the images are found and their 2D positions are converted into 3D depths. Thus, how this operation of building a dense depth map, assigning relatives depth to the input image pixels, is achieved. This is based on the disparity, defined as the amount of horizontal motion between two properly configured images of a stereo pair. This one is then inversely proportional to the distance from the observer, i.e. the camera. Although this concepts are relatively simple to understand, the challenging task within this process stands in establishing dense and accurate inter-image correspondences[10]. As already underlined, stereo matching is one of the most widely studied topic in computer vision from years and it continues to be one of the most active research in that field. In fact, modelling of human visual systems, robotic navigation and manipulation and autonomous driving [2] and 3D model building are some of the possible applications.

The explanations of the fundamental principles of stereo matching, such as epipolar geometry, rectification and disparity map, follows.

### 1.1.1 Stereo geometry

Main goal of epipolar geometry is the computation of pixels correspondences among the input images. Neighbouring pixels information, cameras positions and their calibration data are fundamental to achieve that. Figure 1.1 demonstrate a pixel in one image  $\mathbf{p}_1$  projected to its correspondent epipolar line segment in the other image, which is lower bounded by the projection of the first camera center into the second camera plane, i.e. the epipole  $\mathbf{e}_2$ . Projecting the epipolar line in the second image back to the first, another line would be obtained, bounded by the correspondent epipole  $\mathbf{e}_1$ . The extensions to infinity of these two segment are identified as the epipolar lines, which are defined by the intersection of the two image planes with the epipo-

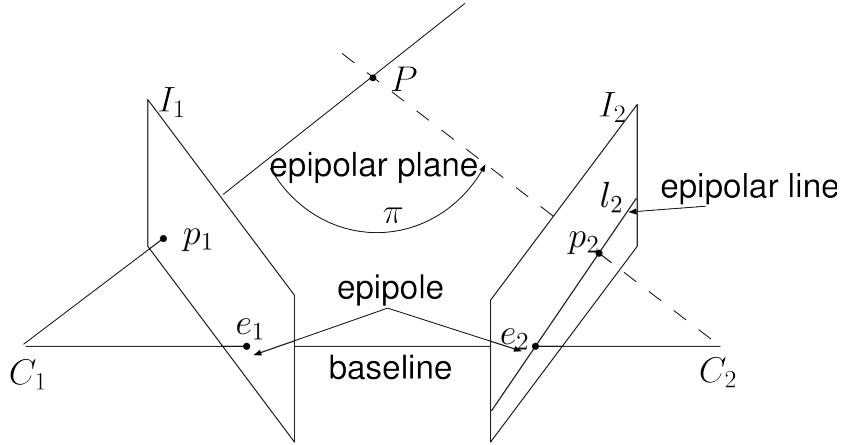


Figure 1.1: Epipolar geometry

lar plane. A fundamental property is that the epipolar plane passes through both camera centers  $C_1$  and  $C_2$ , as well as point  $P$ . Therefore, they lie in the same plane.

### 1.1.2 Rectification

Epipolar geometry for a pair of cameras is relative to pose and calibration of the camera and can be computed using the fundamental matrix, which can be obtained applying the eight point algorithm [11]. Computing this geometry allows, then, to find the correspondent pixels between the two images using the constraint of the epipolar lines. This is possible, because, as explained in 1.1.1, considered a pixel in one image, the correspondent one lies on the relative epipolar line.

Beside that, pixels correlations can be more efficiently performed by rectifying the input images [11]. In Figure 1.2 is clearly visible the outcome of this process and its advantages. As shown, corresponding horizontal scanlines are epipolar lines. The essential importance of this standard rectified geometry is clearly explained by the following equation,

$$d = f \frac{B}{Z} \quad (1.1)$$

that leads to a linear relationship between 3D depth  $Z$  and disparity  $d$ , where  $f$  is the focal length (in pixel) and  $B$  the baseline. Moreover, the relationship between the corresponding pixels in the left and right images

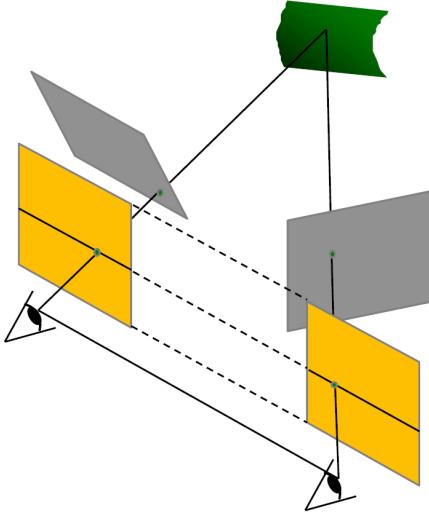


Figure 1.2: Image rectification – Source: L. Lebezniak

can be defined as follows:

$$x' = x + d(x, y), \quad y' = y \quad (1.2)$$

Thus, the main step for recovering a depth image of a scene is the estimation of the disparity map  $d(x, y)$ .

As introduced at the beginning, the best disparity map is estimated after the rectification process. This is performed by comparing the similarity of corresponding pixels, as defined in equation 1.2, and storing them in a disparity space image  $C(x, y, d)$ , which is then processed with multiple algorithms.

### 1.1.3 Stereo methods and dense correspondence

In this section a brief delineation of the general pipeline implemented in most of the stereo matching method is presented. Moreover, as a theoretical completion of what introduced above, some generic algorithm are further explained.

Stereo algorithm follows in general a subspace of the following methods: matching cost computation, cost aggregation, disparity computation and optimization and disparity refinement [5].

A preliminary distinction, based on those phases, separates stereo methods between local or window-base global methods.

In local methods, the disparity computation in a certain region depends on the pixel intensities within a limited window.

On the contrary, global algorithms, are based on an energy function. In this one smoothness assumptions are defined and then a global optimization problem is solved. These algorithm are mainly distinguished considering the minimization strategy, such that, simulated annealing, graph cuts or belief propagation.

Between these two classes there are iterative and hierarchical algorithms. The latter aim to constraint gradually the disparity estimation from the coarser to the finer levels [4].

Considering the first general step of stereo matching algorithms, the matching cost, there are multiple measures to define it. Among the most prevalent pixel-based algorithm can be included square intensity differences, absolute intensity differences, mean-squared error and mean absolute difference.

Other common matching cost comprehend normalized cross-correlation, which is similar to sum-of-squared-difference (SSD), and binary methods. However, the latter tend to not be used any longer.

On the other hand, lately, more robust algorithms are used for their insensitivity to non-stationary exposure and illumination changes. Entropy measures and non-parametric functions such as, rank and census transform [12], sampling insensitive difference[7] and hierarchical mutual information [4], are some examples. In particular, they allow to obtain accurate performance when considerable exposure or appearance variations are present.

Drawing up some conclusion regarding the local methods, the core steps are the matching cost calculation and the aggregation phase. Disparity estimation, then, becomes trivial. Each pixel takes the disparity levels whose cost value is the minimum. This approach is said to be a local *winner-take-all* optimization. A drawback of this approach is that the matches are imposed for the reference image. While points in the support image might have multiple correct matches. For this reason, cross-checking and post-processing become more relevant here.

Summarizing the general pipeline of global stereo matching methods, they often get rid of the aggregation step. They usually perform sort of optimization steps after disparity estimation, exploiting the smoothness constraints as aggregation part.

Goal of this approach is to find the solution to a global energy function, i.e. the disparity  $d$  that minimizes the energy,

$$E(d) = E_d(d) + \lambda E_s(d) \quad (1.3)$$

where  $E_d(d)$  is the data term and  $E_s(d)$  the smoothness term. Adopting the aforementioned disparity space image (DSI) matching cost, the data energy

is calculated as:

$$E_d(d) = \sum_{(x,y)} C(x, y, d(x, y)) \quad (1.4)$$

where  $C$  is the DSI. Then, the smoothness term is usually defined as:

$$E_s(d) = \sum_{(x,y)} \rho(d(x, y) - d(x + 1, y)) + \rho(d(x, y) - d(x, y + 1)) \quad (1.5)$$

where  $\rho$  is some monotonically increasing function of disparity difference. For some implementations, the smoothness energy term can also be based on intensity differences,

$$\rho_d(d(x, y) - d(x + 1, y)) \cdot \rho_I(\|I(x, y) - I(x + 1, y)\|) \quad (1.6)$$

where  $\rho_I$  is a monotonically decreasing function, which depends on the intensity differences and makes the smoothness costs lower at high-intensity gradients.

After the energy function has been clearly identified, different categories of algorithms can be exploited to recover a (local) minimum. Graph cut, belief propagation and Markov random field (MRF) based methods have been proved to give the most accurate results.

Mentioning some hybrid methods, there are cooperative algorithms and others based on coarse to fine incremental steps. Cooperative algorithms were some of the earliest proposed for disparity estimation. They are influenced by human stereo vision processing models. They operate by iteratively improve disparity evaluations using non-linear calculations, leading to a result similar to the one of the global methods. Iterative algorithms are, instead among the current best algorithms. The main idea is to successively choose the best disparity among all of the possible ones. A coarse-to-fine framework is usually used to speed up the computations.

Dealing with global optimization methods, it is worth to mention dynamic programming (DP) technique. Unlike solutions based on equation 1.3, dynamic programming allows to reach global minimum exploiting independent scanlines. This approach works over a slice of the DSI, i.e. the matching cost cube, finding the path associated to the minimum cost. The generic implementation of DP along a scanline  $y$  and for each input state in a 2D cost matrix  $D(m, n)$  leads to combine its DSI value with its previous cost values as follows,

$$C'(m, n) = C(m + n, m - n, y) \quad (1.7)$$

Correct cost selection in presence of occluded pixels and difficulties with scanline consistency are some of the weakness of DP. Multiple algorithms have

been proposed to recover from these problems. Scharstein and Szeliski [5] proposed an alternative to standard DP, improving recursively independent scanlines in the global energy function,

$$D(x, y, d) = C(x, y, d) + \min_{d'} \{D(x - 1, y, d') + \rho_d(d - d')\} \quad (1.8)$$

An upgrade of this scanline optimization is the aggregation cost approach used in semi-global matching [4]. In this case, a cumulative cost function is evaluated from at least eight directions. Intuitively, this approach accesses accurate results and it is highly efficient.

Considering more recent improvements to stereo matching, segmentation-based techniques hold a prominent position. In this case, an initial segmentation of the reference image is performed. Then, disparities are estimated pixelwise using local methods. Citing couple of recent approaches, Klaus, Sormann and Karner [8] segment the image with mean shift, to get initial disparity estimations. Then they apply re-fitting with global planes, and perform final MRF with loopy belief propagation.

Wang and Zheng [13] built a similar top ranked algorithm. They segment the image with local plane fits. Then run cooperative optimization of neighbouring plane fit parameters. Others developed similar approaches exploiting color correlation and left-right consistency check for occlusion detection [14] or focusing on alpha mattes fractional pixel extraction [15].

As explained in section 1.1 the area of stereo matching and disparity estimation is one of the most extensively researched topic in computer vision. Nowadays approaches based on convolutional neural networks and deep learning are going to be the highly ranked in the standard database. Although, their performance in accuracy tends to decrease a lot when moving from dataset images to real ones.

Therefore, novel strategies based on standard stereo geometry algorithms could reach consistent accuracy even in real time [16]. Thus, as described above, after the structure of the cost volume or DSI has been delineated, the actual pixelwise photoconsistency measures are computed. Multiple methods to achieve this has been proposed during years and already explained in the previous paragraphs. Then, depth computation is obtained with different form of optimizations. These ranges among local, global or hybrid frameworks.

Consequently, starting from classical stereo-based methods and building up a novel pipeline, accurate real time depth map estimations can be achieved.

## **1.2 Structure of the Thesis**

You should use transition in your text, meaning that you should help the reader follow the thesis outline. Here, you tell what will be in each chapter of your thesis. Often the thesis does not have as many chapters as is in this template. For example, environment and implementation can be combined as well as chapters of evaluation and discussion. The rest of this thesis is organized as follows. Chapter 2 gives the background, etc.

# Chapter 2

## Theoretical Background and Related Work

In chapter 1 a brief general analysis of stereo geometry and methods has been provided. In this chapter a more precise revision of the theoretical tools that stereo matching methods exploit is presented. Epipolar geometry, camera calibration and disparity estimation algorithms are specifically described. Starting from the necessary mathematical basis, the discussion moves on the disparity estimation algorithms. Then, the chapter focus on the main benefits and drawbacks of standard and novel approaches in depths computation. Comparison between stereo-geometry based and deep learning based algorithms is proposed, to provide a clear explanation of the decisions implemented.

### 2.1 Epipolar geometry and Rectification

Fundamental problem of stereo vision is the estimation of 3D locations of points from at least two corresponding input images. This process, which comprises concurrent computation of both 3D geometry and camera pose, is generally known as structure from motion [10].

In the explanation of these topics it is necessary to start discussing about the triangulation. Then, the concept of epipolar geometry is outlined and after that the notions of camera calibration and rectification.

#### 2.1.1 Triangulation

Triangulation is the problem of detecting 3D points positions from a collection of corresponding 2D image locations, assuming that camera positions

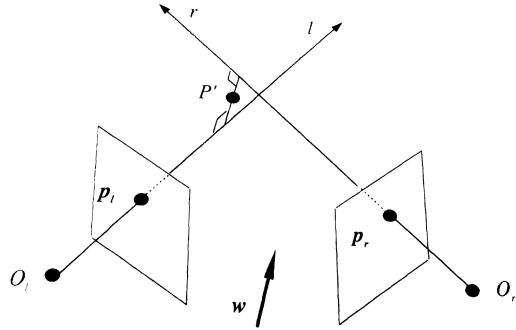


Figure 2.1: 3D triangulation by finding point  $P'$  that lies nearest to all of the optical rays

are known. Figure 2.1 shows one of the easiest methods to tackle this problem. Objective is to evaluate the 3D position of  $P'$  that have the smallest error to all of the 3D optical rays coming from the camera centers, which identify the 2D point locations in the image plane, i.e.  $P_r$  and  $P_l$ . As shown in Figure 2.1, the rays starts from the camera centers,  $O_j$  and go in direction of  $r$  and  $l$ , which can be defined using the camera matrix  $\{P_j = K_j[R_j|t_j]\}$ . The closest point to  $P$  on this ray minimizes the distance,

$$\|O_j + d_j \hat{v}_j - P\|^2 \quad (2.1)$$

Therefore, because of the minimum is  $d_j = \hat{v}_j \cdot (p - c_j)$ , the nearest points are calculated as:

$$q_j = O_j + (\hat{v}_j \hat{v}_j^\top)(P - O_j) = O_j + (P - O_j)_\parallel \quad (2.2)$$

Hence, the optimal value for  $P$ , obtained solving a least square problem, becomes,

$$P = \left[ \sum_j (I - \hat{v}_j \hat{v}_j^\top) \right]^{-1} = \left[ \sum_j (I - \hat{v}_j \hat{v}_j^\top) O_j \right] \quad (2.3)$$

### 2.1.2 Epipolar geometry

The intrinsic projective geometry between two views is known as epipolar geometry. It is only dependent on the cameras' internal parameters and pose. The  $3 \times 3$  rank 2 matrix that defines this geometry is the fundamental matrix  $F$ .

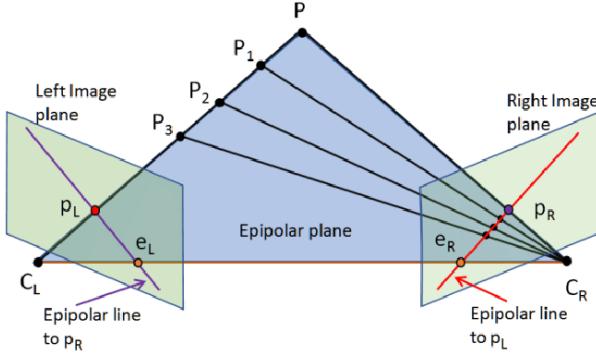


Figure 2.2: Epipolar geometry. Image point  $m$  back-projectes to a ray in a 3D space defined by  $C$  and  $m$ . This ray becomes a line  $l'$  in the second view. The image of  $X$  must lie on  $l'$

The epipolar geometry is the basis for finding corresponding points in stereo matching. It is basically defined by the intersection between image planes and the one on which the cameras baseline lies.

A fundamental property, that makes this geometry extremely useful, is that image points, space point and camera centers are coplanar. Assuming that only  $m_l$  is known, that geometry allows to constraint the corresponding point  $m_r$ . The epipolar plane is defined by the baseline and the ray that comes from  $m_l$ . Hence, knowing that  $m_r$  lies on the same plane, that point belongs to  $l_r$ , i.e. the intersection between the epipolar and the second image plane. Therefore, exploiting this property, the searching of corresponding points is constrained to only one line inside the image.

Mathematical definition of the epipolar geometry is the fundamental matrix  $F$ . As already demonstrated through Figure 2.2, for each point  $p_L$  in one image, the corresponding epipolar line  $e_R$  to that point belongs to the other image plane. Moreover, any point  $p_R$  in the second image, which is related to point  $p_L$ , lies on  $e_R$ . Hence, the epipolar line is described as the projection in the second image of the ray that comes from the point in the first image, passing through its camera center. This defines a map,  $p_L \rightarrow e_R$ , which relates the points in one image with the corresponding epipolar lines in the second image. This correlation, between points and lines, is represented by the fundamental matrix  $F$ .

Considering the aforementioned map  $p_L \rightarrow e_R$  described by  $F$ , an important property of the fundamental matrix is defined,

$$p_R^\top F p_L = 0 \quad (2.4)$$

Therefore, assuming two corresponding points  $P_L$  and  $p_R$ , it is known that  $p_R$  lies on the epipolar line  $l_R = Fp_L$ . Thus, the mathematical correlation is,

$$0 = p_R^\top e_R = p_R^\top Fp_L \quad (2.5)$$

Reciprocally, if image points comply the relation 2.4, then the rays identified by these points are coplanar. For point corresponding this is a necessary condition. Equation 2.4 is extremely important because it allows to characterize the fundamental matrix without reference to the camera matrices [11]. Thus, using at least 7 correspondences, it is possible to recover the fundamental matrix  $F$ . This estimation is known as *weak calibration*.

#### Maybe add 8-point algorithm description

### 2.1.3 Rectification

Image rectification is defined as the process of obtaining a pair of *matched epipolar projections* from a pair of stereo images, which are taken from generally differing viewpoints. In the rectified projections the epipolar lines become parallel with respect to the x-axis. Thus, they match between the stereo pair and so the disparities are in the x-direction only.

In order to obtain a rectified stereo pair, 2D projective transformations are employed to the images, so that the epipolar lines can match. Using this method, the transformations are built up in a way that the corresponding points have almost the same x-coordinate. Actually, this strategy leads to a minimal distortion on the images, being the two transformations arbitrary. However, working on rectified images, the matching problem is highly simplified, being correlated only to epipolar geometry and near-correspondence. Core problem of this section is to find the appropriate projective transformation  $H$ . Indeed, to get epipolar lines parallel with x-axis, the epipole should be mapped to an infinite point. This, has to be done correctly, otherwise intensive projective distortion of the image can happen. For this reason, constraints are put on the definition of  $H$ .

First of all, restricting  $H$  to be a rigid transformation in the neighbourhood of a given point<sup>1</sup>, the errors are reduced.

Once the epipole has been mapped to infinity, it is then necessary define a map to match the corresponding epipolar lines. This resampling is build up in such a way that, being  $e_L$  and  $e_R$  any pair of epipolar lines, then,

$$H^{-\top} e_L = H'^{-\top} e_R \quad (2.6)$$

---

<sup>1</sup>this means that to first-order, the neighbourhood of the point may be subjected to rotation and translation only

Satisfying the condition above, a matched pair of transformations is recovered.

Specifically, at first  $H'$  is chosen, so that it can map the epipole  $e_R$  to infinity. Then the matching transformation  $H$  is defined minimizing the sum-of-square distances,

$$\sum_i d(Hp_{L_i}, H'p_{R_i})^2 \quad (2.7)$$

Therefore, the full algorithm can be summarized as follows.

The outcome of this resampling process is a pair of stereo images whose epipolar lines are horizontal. Hence, the disparities are calculated along the epipolar lines. First of all, at least seven corresponding matches are defined. This allows to compute the fundamental matrix  $F$ , applying the so called eight-point algorithm, and after that the two epipoles are found. After that, there is the selection of the projective transformation  $H'$ , that maps the epipole of the support image to infinity. The corresponding transformation  $H$  is found solving the least-square problem. Finally both of the input images are resampled according to  $H$  and  $H'$ .

## 2.2 Stereo methods and dense correspondence

Take the part already written in the introduction  
Refactoring needed between the 2 chapters

### 2.2.1 Stereo geometry based methods

### 2.2.2 Deep learning based methods

Considering that disparity estimation from a rectified stereo pair is still one of the most important tasks in computer vision, the latest years have seen an important development of deep learning based methods.

Usually these approaches comprises a main pipeline, based on a standard local or global method, whose parameters are finely tuned exploiting Convolutional Neural Networks (CNNs). Therefore, in most of the cases, Semi Global Matching (SGM) is used as regularization method, because of its accuracy and low computational time. Then, deep learning based methods are used for tuning the penalty parameters, which are related to smoothness and discontinuity of the disparity map. As described above, those penalties are empirically adjusted in the standard methods. Therefore, the CNNs based approach aims at learn the penalties, in correlation with the 3D structure of

the objects in the scene, to achieve an improvement in the disparity map. As aforementioned, several of these deep learning approaches has been proposed in the latest years, and some of them has been able to reach state of the art level of accuracy on the KITTI datasets.

Even though these method suffer drop in accuracy when shifting from synthetic to real images, it is worth to describe some of them, which are ranked among the state of the art methods in the KITTI benchmark.

Seki and Pollefeys proposed SGM-Nets[1], a CNNs based method for penalty estimation, which exploit SGM as regularization technique in the main pipeline. They used small image patches and their locations as inputs to the network to predict penalties for the 3D object structures. Actually, they developed a novel loss function for training the networks, which inputs are small image patches and their position. Moreover, they managed to separate positive and negative disparity changes, thus to get object structures more discriminatively.

More in detail, SGM-Net produces  $P_1$  and  $P_2$  for each pixel. This is achieved though a training and a testing phase. In the former, the network is iteratively trained by minimizing a *Path cost* and a *Neighbor Cost*. In testing, the standard SGM pipeline is run using the penalties estimated by the network. Actually, the *Path Cost* is how the authors of the paper called the loss function that they developed, which they minimized using forward and back propagation. Moreover, they introduced the *Neighbor cost* function for removing the ambiguous disparities traversed along along the path, which can lead to wrong penalties.

Another approach similar to the one just described is the method developed by Kuzmin et al. [17]. The core of their method is the prediction of the local parameters of cost volume aggregation process, which they perform exploiting a deep convolutional network. Thus, as in [1], they avoid to apply deep learning of pixel appearance descriptors, using classical matching scores instead. Thus, they refuse learning high dimensional descriptors and matching them, as it used to happen in most of the first deep learning based algorithm. On the contrary, they focus the learning on the cost-aggregation step. Thus, they defined the overall matching cost using a combination of census transform and sum-of-absolute-differences. Then, they carry out the cost-aggregation phase, which is peculiar for smoothing the general matching cost and correct the wrong matches, applying the domain transform [18],[19]. Basically, they develop a deep convolutional neural network to estimate on a pixel basis the cost-aggregation parameters and make them spatially varying. In this way, they were able to have smoother disparities on the same object and avoid smoothing across object boundaries. Therefore, combining standard methods for the matching cost and applying end-to-end deep learning

process to the cost-aggregation step, they obtain state-of-the-art accuracy in the KITTI 2015 dataset. Differently from [20], [21] and similarly to [22], they build up an end-to-end learning method that comprises all the part of depth map computation in it. Furthermore, unlike [22], their approach exploits classical stereo matching techniques as modules within a more complex neural network structure. Their process encompasses the definition of a general cost volume, the cost-aggregation step over that volume and the final winner-takes-all label selection. Then, left-right consistency and filling of occluded pixels are performed as post-processing.

A different approach with respect to the previous one stands in the implementation developed by Žbontar and LeCun [20]. They focus their attention on the matching cost computation, which is usually the first step of a stereo matching algorithm. They developed their method using a convolutional neural network to learn similarity measures on small image patches. Specifically, they structure the training in a supervised manner building up a binary classification data set with examples of similar and dissimilar pairs of patches. Moreover, they carry out two different architectures, one adjusted for speed, and another one for accuracy. Thus, the output of the network handles the initialization of the stereo matching cost. After that, post processing steps are applied: cross-based cost aggregation, semi-global matching and finally disparity refinement techniques such as, left-right consistency check, subpixel enhancement, median and bilateral filters.

Technically speaking, they present a convolutional neural network that is trained on pairs of small image patches, for which the disparity value is known. Then, they initialize the matching cost using the output of the network. After that, they propose a series of common post-processing steps, which are though crucial to obtain accurate results. Cross-based cost aggregation is exploited for combining matching cost between neighbouring pixels with similar intensities. Then, constraints are enforced for smoothness and left-right consistency check applied for detect and eliminate errors in occluded regions. The final disparity map is then obtained applying median and bilateral filters, useful for subpixel enhancement.

Related problems to [20] are the works by Haeusler et al. [23] and by Spyropoulos et al. [24]. Using a similar approach, they concentrate their attention on predicting the confidence of the calculated matching cost. Specifically, in the first cited approach [23], the aim was using a random forest classifier to connect multiple confidence measures. Similarly, the authors of [24] focused in estimating the confidence of the matching cost by training a random forest classifier. Then, they employed the predictions as mild constraints in a Markov Random Field (MRF) aiming at reducing the errors of the stereo method.

Focusing on confidence prediction intended at the estimation of dense disparity map, it is worth to cite the work of [25]. They adopted two channels disparity patches as inputs of a Convolutional Neural Network (CNN), predicting the correctness of stereo correspondences, that is the confidence. Using these specific patches, they managed to simultaneously train features and classifiers. Furthermore, they incorporate the predicted confidence into Semi-Global Matching (SGM), adjusting its parameters directly.

Unlike methods based on hand crafted features, which lead to limited accuracy, authors of [25] leverage CNNs to overcome that problem. In fact, CNNs support high performance from low level processing, such as patch based matching, to high level, like scene classification and object detection. Therefore, as previously introduced, they conceive a two channels disparity patch, based on the concept of standard confidence features. Then they apply those patches as input to the network, obtaining a simultaneous training of discriminative features and classifiers. Moreover, they also developed three different types of network structures to manage the trade-off between computational time and accuracy. Finally, the confidence was combined into SGM, so that dense disparity map could be obtained.

### 2.2.2.1 Confidence measures

Considering some of the deep learning based methods described in 2.2.2, becomes necessary to define the main typology of features proposed during the years to estimate the confidence of stereo correspondences.

Because of many features were introduced by different works in computer vision field, Hu and Mordohai [26] developed a broad evaluation of them, leading to the definition of five groups, used for categorize those features.

The first group is correlated to the matching cost. This means that correct correspondence are unlike to be connected to large matching costs. The second group focuses on local properties of the cost curve. That is, confidence measure becomes the curvature around the minimum matching cost. For example, flat curves, which have smaller values, and describe textureless areas express higher ambiguity. Features related to local minima of the cost curve form the third group. As an example, the so called *Peak Ratio (PKR)*, is calculated as the minimum matching cost divided by the second local minima. Then, a probability mass function over disparity defined using the entire cost curve describes the fourth group. The last group includes features that highlight the consistency between left and right disparity map, i.e. correct matches indicate more consistent disparity.

A similar but more recent work is the one carried out in [27] driven by

the deep breakthrough lately happened in computer vision field. In fact, changing such as, availability of bigger and more challenging datasets, novel and more accurate stereo algorithms and confidence measures, leverage techniques based on deep learning. Therefore, the authors of [27] focus on implement a complete and updated quantitative analysis of the state-of-the-art confidence measures.

As a matter of fact, after the analysis performed in [26], major improvements have happened in computer vision field. Among those the most relevant can be summarized as follows:

- enhanced confidence prediction algorithms based on deep learning [28] and on random-forests [24], [29];
- larger dataset providing challenging indoor and outdoor scenes [22];
- more accurate stereo algorithms and novel implementation of the SGM pipeline [25], [20]

Therefore, in [27], the authors extend and update the taxonomy previously performed in [26], especially focusing on machine learning techniques. Then they aim at evaluating the algorithms' performance over the novel and larger datasets, concentrating on the correlation between the availability of training data and the correctness of confidence measure prediction. Moreover, they estimate the accuracy of the methods when handling new data and calculate the effectiveness of the predictions when included in state-of-the-art pipelines.

Considering that among the multiple confidence measures proposed, all of them deal with the cost curve and the relationship between left and right image or disparity map, basing on [26], Poggi et al. [27] grouped confidence measures according to their input cues. Specifically, they considered 76 confidence measures, which were then grouped into 8 categories and evaluate them employing three state-of-the-art stereo algorithms and three ground truth datasets: Middlebury 2014 [30], KITTI 2012 [31] and KITTI 2015 [32]. Performing that exhaustive analysis, the authors carried out the results that learning based approaches seem to be more efficient than conventional ones. Especially, using disparity maps as input cue more accurate results are achieved in terms of correct matches, adaptation to new data and stereo accuracy improvement. Beside that, training remains an issue for those methods though. As a matter of fact, for deep learning based approaches the general amount of training data is still limited and in most case they struggle when dealing with new real data.

## 2.3 Image processing techniques

In almost all the computer vision methods image processing technique are applied in different phases of the algorithm pipeline in order to model the image in a form convenient for subsequent analysis. Image processing stage is a key component of most of the computer vision applications, such as object recognition, stereo matching, computational photography, scene reconstruction, 3D pose recognition or motion flow, in order to achieve reasonable results. Different types of processing operations are usually applied, depending on the type of task demanded. For example commonly applied procedures includes exposure correction, color balancing, noise reduction, smoothness enhancement, sharpness increasing or feature detection.

In relation to this Master's thesis project, it is worth to focus on some of those operations, which result to be most relevant for the designed algorithm. Especially, point operators, area-based and global image transform techniques will be analysed. Particularly, neighborhood (area based) operators will receive specific attention, being the ones specifically employed in the developed algorithm.

### 2.3.1 Point operators

Point operators are defined as the simplest type of image processing transforms [10]. For these operators the output pixel value is entirely related to the corresponding input pixel value. Brightness and contrast adjustment, or color correction and transformation represents some of those techniques.

Basically, an image processing operator is identified, in the continuous domain, by the following function that takes one or more input images and generates a corresponding output:

$$g(x) = h(f(x)) \quad \text{or} \quad g(x) = h(f_0(x), \dots, f_n(x)) \quad (2.8)$$

where  $x$  is in the D-dimensional domain of the functions and  $f$  and  $g$  operate over some range. Specifically, for discrete images, the domain comprises a finite number of pixel locations, i.e.  $x = (i, j)$ , so that:

$$g(i, j) = h(f(i, j)) \quad (2.9)$$

Among these pixel-based operators it is sufficient to cite color transforms, image matting and histogram equalization. Therefore, considering color images as arbitrary vector-valued functions, it is coherent to identify them as

highly correlated signals with strong connections to the image formation process.

Taking into account particular image editing application, where for example the goal is to take a foreground object from a scene and to put it into a different background. The first step of this processing is called matting, that is cut of a specific object from scene. Whereas, the second part, the positioning over a different background is defined compositing.

Histogram equalization is, rather, a more widely exploited technique. This algorithm is based on the histogram of the individual color channels and luminance values. Thus, using information about minimum, maximum and average intensity values of the image it is possible to equalize the pixel intensity value of the whole image, that is identify the intensity mapping function  $f(I)$  such that the resulting histogram is flat.

### 2.3.2 Neighborhood operator

Differently from the previous type of operators, in this case the output pixel value is evaluated on the basis of its neighboring pixel values. This category of operators is usually applied for local tone adjustment, but more specifically for multiple kinds of image filtering, such as blurring, sharpening, feature detection or noise removal. Considering these local transforms, it is important to distinguish between linear and non-linear filtering operators.

Linear operators are the most generally used in terms of are-based operators. They relate weighted combinations of pixels in a neighborhood in order to estimate the output pixel's value. Basically, they can be described by the following function:

$$g(i, j) = \sum_{k,l} f(i + k, j + l)h(k, l) \quad (2.10)$$

where the entries in the weight kernel  $h(k, l)$  are called *filter coefficients*. More simply, the relation 2.10 can be written as:

$$g = f \otimes h \quad (2.11)$$

An important mention has to be done in relation to the separable filters. As a matter of fact, when filters are applied to a 2D image and convolution is taken into account, that requires  $K^2$  operations per pixels, i.e. multiplication and summation, defining  $K$  as the kernel size. In most cases, the filtering process can be accelerated carrying out a one-dimensional horizontal convolution and subsequently a vertical convolution. This procedure requires, in fact, only  $2K$  operations per pixels. However, this cannot be always done,

but it can be applied only to the convolution kernels that are said to be *separable*.

The proof of that property for a convolution kernel can be achieved by inspecting at its analytic form, as shown in [? ]. More precisely that can also be checked handling the 2D kernel as a 2D matrix  $\mathbf{K}$  and taking its singular value decomposition (SVD), as follows:

$$\mathbf{K} = \sum_i \sigma_i \mathbf{u}_i \mathbf{v}_i^\top \quad (2.12)$$

Therefore, it is proved that if only the first singular value  $\sigma_0$  is non-zero, the kernel is separable and  $\sqrt{\sigma_0}u_0$  and  $\sqrt{\sigma_0}v_0^\top$  give the vertical and horizontal kernels respectively [10]. Among linear filters, there are some that are commonly used for pre and post processing operations over images, which were tested in this project too.

The moving average or box filter is regarded as the simplest one. It basically performs a convolution over the image with a kernel of all ones and the result is then scaled. The bilinear kernel is, instead, a specific version of the *Bartlett* filter. Actually, the bilinear filter is the outer product of two linear splines.

Useful kernel for accurate noise removal are the *Gaussian* kernels. As the operators introduced previously, they are example of low-pass kernels, whose effect is softening higher frequencies, which are correlated with the noise components of the signals<sup>2</sup>.

Taking into account the non-linear filters, they are usually exploited when dealing with the need of achieve more accurate results with respect to the outcome given by linear operators. Moreover, on top of that, there are the morphological operators, which are neighborhood kernels working with binary images.

Focusing on the non-linear operators, some of them has been employed in the designed of this project, in particular in the image pre-processing step of the main pipeline and in the post-processing phase. Among all the available non-linear area-based kernels, it worth to mention the applied ones.

The median filter is one of these. Basically, it takes the median value from each pixel's neighborhood. This kind of filter becomes quite performing when dealing with shot noise, which is difficult to be removed if applied a standard Gaussian kernel. However, moderate computational cost and the fact that it modify the values pixel-wise are drawbacks of this filter, which has to be taken into account. For this reason, a *weighted* median filter can be implied, instead. In this case, each pixel of the subregion is counted a different

---

<sup>2</sup>In the Fourier frequency-space the noise appears to be a high frequency signals



Figure 2.3: Original test image for filter comparison. Shot noise is added on top of the image. Credits [10]

number of times depending on its distance from the center. Therefore, this can be equivalently formulated as the minimization of the weighted objective function:

$$\sum_{k,l} w(k,l) |f(i+k, j+l) - g(i, j)|^p \quad (2.13)$$

where  $g(i, j)$  is the requested output value and  $p = 1$  for the weighted median.

Besides its heavier computational cost, compared to the linear filtering, non-linear noise removing operators are usually preferred because of their more accurate *edge preserving* capability. Hence, when cleaning away noisy frequencies, they tend to soft less the edges.

Figure 2.3 shows a test image where shot noise is added. Thus, as visible in Figure 2.4, the Gaussian filter, trying to cancel most of the noise, tends to flatten high-frequency details, which are localized close to strong edges. Contrarily to the result in Figure 2.4(a), the median filter, shown in Figure 2.4(b), is more edge preserving.

Hence, the median filter was chosen in the initial part of the pipeline of this project for the image pre-processing phase, especially for its capability of preserving the edges, without, then, smoothing away the discontinuities.

Bilateral filtering is another type of non-linear operator, which was tested during the development of the designed algorithm. Basically, during the image pre-processing implementation this specific kernel was one of the analysed options. In fact, it is one of the most common option for noise cancelling in computer vision algorithm. technically speaking, in the bilateral filter, the

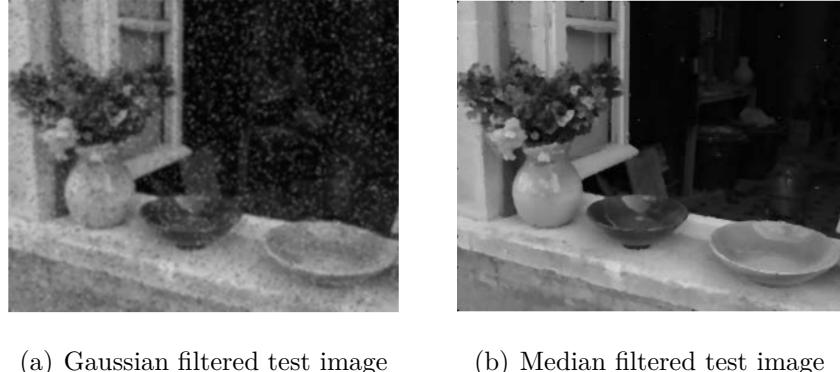


Figure 2.4: Test images example taken from *Computer Vision: Algorithms and Applications* [10]

value of each transformed pixel is a weighted combination of its neighboring pixel values, as defined by the following equality:

$$g(i, j) = \frac{\sum_{k,l} f(k, l)w(i, j, k, l)}{\sum_{k,l} w(i, j, k, l)} \quad (2.14)$$

where the coefficient  $w(i, j, k, l)$  is correlated to the multiplication between a +textit{domain kernel}, defined in equation 2.15, and a data-based *range kernel*, illustrate by equation 2.16.

$$d(i, j, k, l) = \exp\left(-\frac{(i - k)^2 + (j - l)^2}{2\sigma_d^2}\right) \quad (2.15)$$

$$r(i, j, k, l) = \exp\left(-\frac{\|f(i, j) - f(k, l)\|^2}{2\sigma_r^2}\right) \quad (2.16)$$

Therefore, the *bilateral weight function* can be finally expressed as:

$$w(i, j, k, l) = \exp\left(-\frac{(i - k)^2 + (j - l)^2}{2\sigma_d^2} - \frac{\|f(i, j) - f(k, l)\|^2}{2\sigma_r^2}\right) \quad (2.17)$$

An unavoidable drawback of the bilateral filter is closely correlated to its computational time, if compared to regular separable filter. For this reason, as outlined in [10], multiple acceleration methods have been developed during the last decade. Nonetheless, those algorithms are prone to a higher memory used w.r.t. the regular filtering, thus they should not be applied to full-color image filtering.

A last family of non-linear operators that should be introduce in this chapter

consists of morphological filters. Regarding the designed algorithm, morphological operators have been actually implemented in the post-processing phase in order to remove small estimation error in the 3D point cloud, enhancing the final dense depth 3D reconstruction of the scene.

From a theoretical point of view, *morphological operations* are typically implemented over binary or thresholded images. This kind of filtering techniques are executed by first convolving the input image with a binary structuring element, which could have different shapes, and after that the result value is defined basing on the outcome of the convolution.

If a binary image  $f$  is considered and  $s$  is the morphological kernel, the convolution operation is defined as:

$$c = f \otimes s \quad (2.18)$$

where  $c$  is the number of 1s inside each structuring element when shifting through the image. Therefore, designating  $S$ , the kernel size, the principal morphological operations, which has been tested in the post-processing phase of the algorithm pipeline, comprise:

- **dilation:**  $\text{dilate}(f, s) = \theta(c, 0)$
- **erosion:**  $\text{erode}(f, s) = \theta(c, S)$
- **opening:**  $\text{open}(f, s) = \text{dilate}(\text{erode}(f, s), s)$
- **closing:**  $\text{close}(f, s) = \text{erode}(\text{dilate}(f, s), s)$

## 2.4 Edge detection and segmentation algorithms

Strictly correlated to the latter family of non-linear operation analysed and to the overall performance of the designed algorithm there is the concept of edge detection.

Techniques for finding object boundaries in an image are extremely relevant in computer vision and especially when dealing with stereo matching and disparity estimation. As a matter of fact, object borders coincide with occlusion points in 3D, where the majority of wrong matches takes place (in depth estimation). Moreover, segmentation methods are strongly related to this topic and remarkably useful in the preparatory phase of an accurate stereo matching algorithm.

Specifically to this work, segmentation techniques have not been completely developed. This was due to the fact that the initial data provided by the

laser points grid gives already a good amount of information for obtaining an accurate 3D dense disparity map. Beside that, after the evaluation of the first reasonable results, further improvements of the algorithm has been considered to be an enhancement for smoothness and continuity among the estimations. Thus, edge detection and segmentation methods appear to be effective initial operations for multiple reasons, such as, reducing the computational time of the entire pipeline and increasing accuracy and density of the final estimations.

Therefore, it is worth to present a brief but sufficient theoretical outline of the most common edge detection algorithms designed in the computer vision area, considering that they will be part of the future improvements, which will be applied to work developed so far.

From a qualitative point of view edges are usually located in areas where there are changing in color or texture. However, segmenting an image basing on this information is usually complicated and specific methods has been developed for that.

Therefore, for pure edge detection, regions of fast intensity variation are taken into account. Specifically, edges are usually located in an image where there are steep slopes, in terms of intensity. Mathematically, these surface's characteristics can be defined through its gradient, i.e.:

$$\mathbf{J}(\mathbf{x}) = \nabla I(\mathbf{x}) = \left( \frac{\partial I}{\partial x}, \frac{\partial I}{\partial y} \right) (\mathbf{x}) \quad (2.19)$$

where the local gradient vector  $\mathbf{J}$  symbolizes the direction of the steepest ascent in terms of intensity. Because the percentage of noise is higher at high frequencies, and considering that derivatives enhance those frequencies, a low-pass filter, e.g. Gaussian, is usually applied before the gradient.

Nevertheless, to make edge detection really efficient, it would be desired to refine that continuous gradient to the single pixel locations on the edge contours. This can be achieved by finding the *local maxima* in the gradient magnitude along its direction. Therefore, the maxima are calculated by taking the derivative of the gradient, that means apply a dot between a gradient operator and the previous result. This will lead to the so called *Laplacian*:

$$S_\sigma(\mathbf{x}) = \nabla \cdot \mathbf{J}_\sigma(\mathbf{x}) = [\nabla^2 G_\sigma](\mathbf{x}) * I(\mathbf{x}) \quad (2.20)$$

where  $\sigma$  symbolizes the Gaussian filter used for the initial image smoothing. This final result is usually called *Laplacian of Gaussian*(LoG) kernel, which is a separable filter. However, commonly a slightly different kernel is employed, the *Difference of Gaussian* (DoG), which gives a similar result.

Hence, once the function  $S(\mathbf{x})$  has been defined, its *zero crossing* is computed and so the edge elements can be estimated.

Obviously, if high accuracy is needed, higher-order steerable filters can be exploited.

Furthermore, taking into account the standard DoG kernel, the fundamental parameter is  $\sigma$ , i.e. the filter spatial scale parameters. In fact, it influences the amount of noise on the image and thus the edges scale.

It has already been presented that defining confined edges is convenient for multiple applications, such as in the pre-processing image phase of a stereo matching algorithm. Considering that, combining together these edges, designing a sort of continuous contour, would be even more useful.

For example, if the procedure explained above for defining the edges is taken into account, meaning that the edges were defined through zero crossing, link them together becomes quite simple. As a matter of fact, adjacent edges have the same endpoints. Then, multiple methods exist to encode the edges together forming the contours.

Strongly related to edge detections, there is the topic of segmentation, i.e. group together pixels that share the same characteristic. In computer vision segmentation algorithms are among the first studied and designed topics ([33], [? ], [34]) and they are still nowadays a widely analysed problem ([35], [36]). Some basic segmentation techniques related to the morphological filters has already been presented in the subsection 2.3.2. In this section, the analysis of some commonly used algorithm is carried out. Since in this field the researches have been numerous simple explanations of some of them, such as *active contours*, *region splitting and merging*, *mean shift*, *normalized cuts* and *binary Markov random fields* are presented.

**Active Contours** Active contours [37] method groups together different techniques, such as *snakes* [38], *intelligent scissors* [39] and *level set* algorithms. These methods reach the final solution iteratively thanks to the combination of image and arbitrary user-guidance.

Since the merely theoretical purpose of this analysis and considering that these specific methods have not been widely tested in this project, it is not necessary to describe in detail all of these algorithms. Therefore, if there would be interest in the mathematical background of the aforementioned techniques, we remind to the manual by Szeliski [10].

**Region splitting and merging** Differently from algorithms entirely based on threshold selection and connected components computation, a useful segmentation technique is related to a recursive splitting of the entire image into subregions, which are then merged together hierarchically. Therefore, region splitting and merging is fundamentally focused on multiple stages of split and merge of image areas, at different level of density, mainly basing on region statistics.

An example of this approach is the *watershed* algorithm [40]. The core of this algorithm stands on dividing the whole image into different *catchment basins*, which are identified taking into account all the local minima present and labelling them. In order to make the algorithm suitable also for color images, it is usually applied to the gradient magnitude of the figure.

Moreover, an interesting algorithm, which appeared to be useful as a pre-processing phase of more complex algorithms such as stereo matching, optic flow and object recognition, is centred in region merging. Actually, areas of the image close among each other are combined together considering their average color difference. Therefore, if it is below a certain threshold the adjacent portions of the image are merged into a so-called *superpixel* [41].

**Mean shift** Mean-shift algorithms, as well as mode finding, k-means and mixture of Gaussian, consider position, color or other features of image points and cluster them as they would be taken from a probability density function, finding the peaks of that distribution.

**Normalized Cuts** The idea of the Shi and Maki algorithm [42] is to divide the image into groups distinguished by weak affinities, i.e. similarities. Therefore, in this methods pixels that show strong similarities will belong to the same group. The different groups are, then, separated taking into account the sum of the weights of all the cuts among pixels of different regions. Because, this could lead to unreasonable clusters, the normalized cut measure is used.

**Markov Random Field** Markov Random Field (MRF) optimization is an energy-based algorithm. These energy minimization problems based on MRF are usually solved employing graph cuts techniques.

## **2.5 Data and estimations error types**

# Chapter 3

## Environment

The developed thesis project is based on a major environment that comprises multiple structures. First of all, the Middlebury 2014 [30] dataset, which contains detailed stereo pairs, was exploited to perform multiple tests. Moreover, real indoor rectified stereo images taken with the **company cameras** were employed.

In this chapter a brief explanation of the LaDimo stereo device follows. Furthermore, it is worth to describe the main characteristics of the dataset adopted. As a matter of fact, it results extremely useful during the overall designing of the implementation, giving visual outcome of the improvement provided to the code. The ground truth images available in the dataset were especially effective for simulating the behaviour of the company device during the first part of the code implementation.

### 3.1 Middlebury Dataset

The so called Middlebury 2014 dataset is a high-resolution stereo dataset comprising static indoor scenes and including highly detailed ground-truth disparities. The images were acquired exploiting a novel structured lighting system. This also consists of updated methods for accurate 2D subpixel correspondence search and self-calibration of camera and projectors with modelling of lens distortion.

Generally speaking, Scharstein et al. [30] provide 33 new indoor 6-megapixel datasets using their system. Thus, achieving an accuracy in the disparity estimation of 0.2 pixels on most analysed surfaces, they produce demanding challenges for the stereo algorithms developed since that.

That was, in fact, one of their main objective, being the datasets available until their work insufficient in terms of ground truth accuracy, resolution,

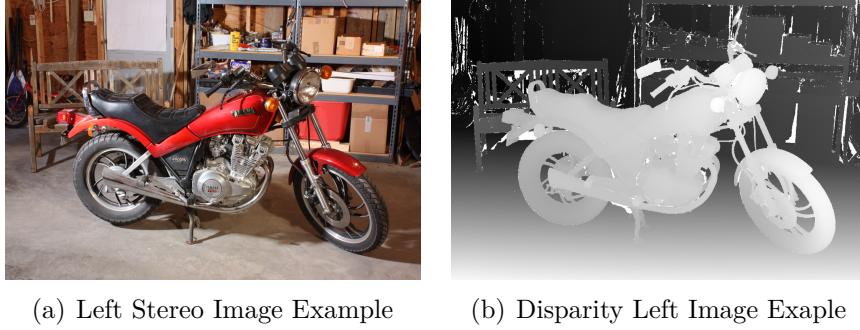


Figure 3.1: Middlebury 2014 dataset example

complexity and realism. Therefore, aiming at updating and improving the work of Scharstein and Szeliski [43], they obtained Middlebury 2014 dataset, which brought major improvements in the level of calibration accuracy for stereo images.

Specifically, the system described in [30] comprises the following novel features:

- a stereo equipment made of two digital single-lens reflex (DSLR) cameras and two point-and-shoot cameras;
- a method for robust interpolation of lighting codes and 2D subpixel correspondence search for obtaining accurate floating-point disparities;
- bundle adjustment for calibration and rectification procedures;
- a robust model selection for self-calibration of structured light projectors and lens distortion;
- a so called *imperfect* version of the dataset showing realistic rectification errors.

Figure 3.1 shows an examples of the datasets provided, which comprise the input images, with different levels of exposure, and *perfect* and *imperfect* rectified images with accurate 1D and 2D floating-point disparities, respectively.

In order to obtain accurate high-resolution stereo datasets, the authors of the described work based their idea on establishing ground-truth disparities from the input views. In this manner, calibration problem are prevented and the process can be performed via structured light. However, the drawback of this is that correct disparities can be achieved only for scene points that are non-occluded in both images. Therefore, extending the idea in [43], that is based on a self-calibration of structured light sources from initial

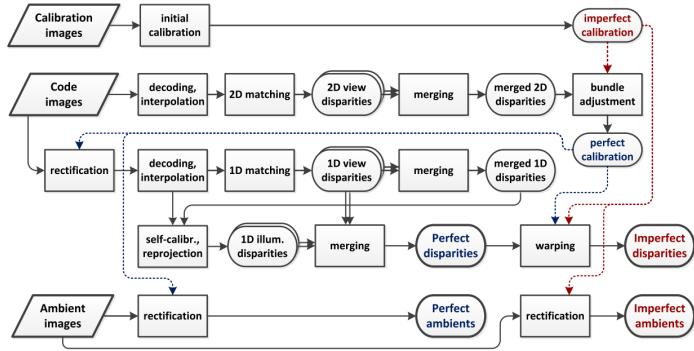


Figure 3.2: Pipeline of the overall system for creating the Middlebury 2014 dataset [30]

non-occluded disparities, they managed to register illumination disparities in half-occluded regions and model projector lens distortion as well. Moreover, imposing initial correspondences they enhance significantly the rectification accuracy. Then, subpixel precision was obtained exploiting a high number of binary patterns under multiple exposures and applying a robust interpolation.

A brief overview of the pipeline of the process, deeply illustrated in [30], is now presented.

Figure 3.2 depicts the overall pipeline of the system defined by Scharstein et al. [30] for reconstruct the stereo datasets.

As simply described, the system's inputs are: calibration images of a standard checker-board calibration target; code images obtained with structured lighting projectors from different positions; *ambient* images collected with multiple lighting conditions. First of all, re-factoring operations are applied to the unrectified code images, i.e. thresholding, decoding and interpolation. This leads to floating-point coordinates of the projector pixel illuminating the scene. Thus, the achieved values are employed as unique identifiers to impose the correspondences between the two input images. So that, the resulting *2D view disparities* are used to refine the *imperfect calibration* in the bundle-adjustment phase. The next step of the process is use the rectified input images to generate the *1D view disparity*. The self-calibration of the projector is, therefore, accessed using the merged disparities and thereby produce the *1D illumination disparities*. View and illumination disparities are, then, combined together into the *perfect disparities*. Last row in Figure 3.2 shows that rectifying with both calibration allows to achieve the corre-

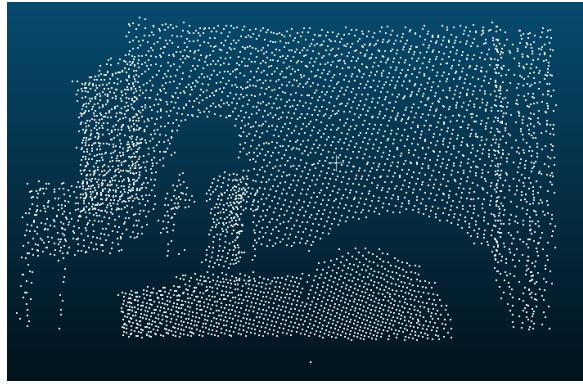


Figure 3.3: Point cloud generated from a test scene

sponding sets of ambient images.

Further details of the single steps of the process are reminded to the original paper [30] so as not to make the discussion much convoluted.

Nevertheless, it is worth to disclose that considering the evaluation of the datasets showed in [30], the choice of the Middlebury 2014 images for the initial tests can be assumed as a correct decision. As a matter of fact, experimental results reported in the original paper demonstrate how that datasets can be considered highly challenging in terms of accuracy and scene complexity. Therefore, they can be assumed as a useful starting point for recovering information from the ground truth images available and for the stereo algorithm evaluation.

## 3.2 Stereo Photogrammetry Ladimo device

One of the fundamental mainstay of this Master’s Thesis project is the device developed by Ladimo, the company at which I am building up my project. Therefore, it is rather useful for a solid understanding of the work done for this thesis to describe, even briefly, the hardware designed by the company and its implementation in the developed work.

Ladimo device is structured light based 3D camera. Its core functionality is to output a 3D point cloud data of the analysed scene for the listening client. Technically speaking, the laser projector mounted on the device splits its beams into the affected area. That set of laser points is then accurately measured by a camera (**or cameras**) sensitive to laser’s wavelength. Therefore, this system allows the user to generate a sparse grid of regularly separated

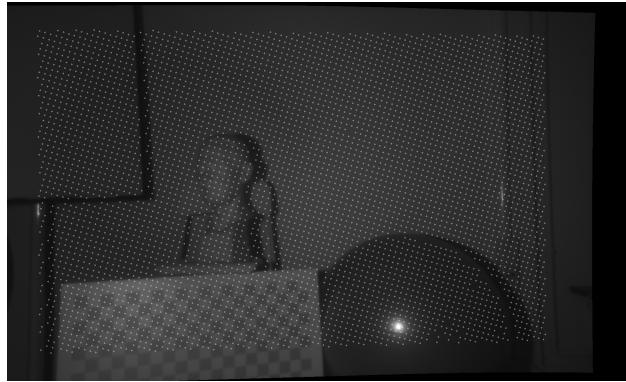


Figure 3.4: Primary stereo image with Ladimo grid point overlapped

points, which contain information on their 3D location, as visualized in Figure 3.3.

So that, the described device can be exploited in order to improve in both accuracy and efficiency a stereo matching problem. This is, indeed, the base of the study presented in this Master’s thesis. In fact, as already deeply described in 2, one of the main drawbacks of creating a 3D dense point cloud using stereo matching techniques stands in the aggregation cost part. And that is closely related to the overall range of disparities associated to each stereo pair. Moreover, other key factors are the definition of the image, the amount of texture-less region contained in the analysed scene and the number of edges, which make the algorithms used in those processes even more complicated and computationally expensive. However, as visible in Figure 3.3 and 3.4, the laser grid generated with the Ladimo system cannot be considered enough dense to constitute an accurate 3D representation of the space. As a matter of fact, the points hit only some areas of the object and their sparsification does not allow to determine a continuous and smooth change between the surfaces of different objects, or even between different regions in the same object. Furthermore, as extremely visible in the lower right-hand side of Figure 3.4, there are sections of the objects that almost completely lacks of points, usually in close proximity of occluded or shaded regions. Therefore, an efficient strategy that can be applied to tackle the problem of creating a dense depth image from a pair of rectified stereo images is to exploit the initial piece of information contained in the grid points data. As a matter of fact, those data can be employed to make the overall pipeline of the algorithm more efficient. First of all, the error rate regarding wrong matches can be highly reduced, then, the disparity range over which the corresponding points are matched can be recursively selected basing on the

disparity values of the neighbourhood of the analysed point. So that, these are only some of the improvement that can be applied to the process utilizing the information that comes from the Ladimo device. Thus, as can be therefore understood, the Ladimo system was fundamental for the efficiency of the overall pipeline of the algorithm. The different part of the algorithm and the achieved results will be deeply and precisely presented in the following chapters. As regards the current analysis of the work, it would be sufficient to introduce the general strategy applied and the reasons that lead to that decision.

Taking into account the output of the stereo device, it comprises the two stereo images of the scene, which have to be rectified in order to have the corresponding point along the same epipolar line, and the grid of points generated from the laser projector. Regarding the laser grid, as can be understood from the brief system description, it can be correctly recovered for one of the images only. As a matter of fact, the grid points are thus overlapped with the primary image, usually the left one. This configuration is, thus, exploited to run the entire pipeline. Moreover, it would be also possible to define the corresponding grid for the right image, employing the camera matrix parameters. So that, a second reciprocal pipeline could be carried out in parallel. In this manner, two different dense depth map are obtained and then handled to apply specific post-processing procedures such as left-right consistency check, which allow to determine possible wrong matches and especially occluded pixel depth values. As aforementioned, the corresponding coordinates of the points can be recovered exploiting the camera matrix parameters, which allow to build up a second grid over the support image. However, this phase is not completely necessary for the main purpose of the designed stereo matching algorithm. It will allow to perform specific and accurate post-processing procedures. Nevertheless, this has to be evaluated in terms of desired accuracy and computational time, considering that, even simpler post-processing techniques, such as morphological or median filters has shown to give an accurate enough 3D dense point cloud.

### 3.3 Software Environment

Considering the programming related part of the built environment, it comprises multiple software and platforms. Although, some computing environments and frameworks were not entirely employed during the overall designing of the project, this was due to the decisions correlated to the way in which the work was conceived and the choices regarding the applicable strategy.

As a matter of fact, in the initial phase different methods and approaches have been tested using MATLAB<sup>1</sup>. Despite the known fact that further working implementation of the algorithm would not be run using an academic version of the aforementioned software, a first researching phase was initially planned. Multiple reasons support that decision, such as a personal considerable knowledge of the software and of its built-in libraries and functions, the possibility to check the results of the performed tests in an user-friendly manner and the side own need to get confidence with the company's hardware and software environment.

Taking those choices into account, the initial part of the work was dedicated to research and tests necessary to evaluate the proper strategy for the algorithm development and to conduct comparisons between some initial ideas and a bunch of available methods.

### 3.3.1 MATLAB computing environment

As introduced above, primary tests have been handles using MATLAB® and especially some built-in functions correlated to stereo matching and image manipulation.

MATLAB can be defined as a platform composed of a multi-paradigm<sup>2</sup> numerical computing environment and a proprietary programming language created by MathWorks. It is significantly used for matrix manipulation, algorithm implementation and plotting of data and functions. Moreover, one of its most utilized packages is Simulink, which allows the user to develop even extremely complex algorithms, especially for dynamic and embedded systems, with a block-based interface.

Although its high versatility and possibility to design programs that can be run of different systems, it was decided not to make it the main environment of the project, in order to not be limited from both a software and patent perspectives.

Consequently, after the initial preparatory phase of the project, the primary development of the algorithm has been realized through a C++ code, which has been built up using Microsoft Visual Studio and the OpenCV library.

---

<sup>1</sup>The software was used for academic purposes being available through the personal university account

<sup>2</sup>functional, imperative, procedural, object-oriented, array

### 3.3.2 Microsoft Visual Studio integrated development environment

Microsoft Visual Studio is an integrated development environment (IDE) created by Microsoft. It is used for several purposes, such as software development, websites, web services, mobile as well as web applications. Obviously, it is based on Microsoft software development platforms, e.g. Windows API, Windows Form and Windows Presentation Foundation. It is also advisable to mention some of its assets, which make it the most preferred IDE, in terms of popularity<sup>3</sup>.

First of all, it allows to create both native, or machine language, code and managed code. Moreover, among its most powerful features there are IntelliSense, the intelligent code completion included in the editor, and the integrated debugger, which operates both as source-level debugger and a machine-level debugger. Furthermore, it contains lot of other built-in tools, designed for multiple applications and objectives, plug-ins that enhance coding capabilities and it supports almost any programming language.

Considering specifically its architecture, it actually does not support inherently any programming language, solution or tool. Rather, it permits the integration of functionalities coded as VSPackage, i.e. Visual Studio Package, which are software modules that broaden the Visual Studio IDE by supplying UI elements, services, projects, editors and designers. Therefore, at its native behaviour, it works as a *Service* solution. As an example, the support for programming language is added by using a specific VSPackage known as *Language Service*. Exploiting that service various functionalities can be included in the IDE.

### 3.3.3 C++ programming language

Subsequently to the initial preparatory phase, in which the main employed environment was MATLAB, the actual design of the final algorithm has been developed basing on the C++ syntax.

The decision of exploiting MATLAB only for the initial planning of the work and for the first tests correlated to the feasibility of the various ideas was mainly due to multiple reasons, which cover different aspects of the entire design of the project. First of all, since the beginning there was the intention of designing an algorithm that would be adaptable to different types of hardware and which would be further adjusted to other programming languages,

---

<sup>3</sup>According to data from Google Trends and available at <https://pypl.github.io/IDE.html>

e.g. CUDA, in order to obtain better final performances. Moreover, considering plausible forthcoming commercial issues, there was the need to ensure a developing as free from copyright as possible. Flexibility and efficiency were other fundamental drivers of that choice. As widely known in the technological field, C++ programming language can ensure velocity and efficiency of the designed software as well as flexibility with different types of hardware. In fact, it can be optimal for coding both at high levels of abstraction, and at low levels, e.g. at machine language level. Then, in relation with this very project implementation, using C++ was possible to widely exploit the functions of the OpenCV library.

Therefore, adopting C++ as the fundamental programming language of this Master's Thesis project, it was possible to reach a sufficient level of computational speed, trying to minimize the memory requirements and simultaneously keep a high amount of abstraction, necessary for working effectively in the computer vision field.

### 3.3.4 OpenCV cross-platform library

OpenCV, Open Source Computer Vision Library, is a library of programming algorithms primarily intended at real-time computer vision.

It was employed as a fundamental support structure of this project mainly because of the functions accessible in that library, the fact that is open source and the remarkable documentation available. Moreover, thanks to those algorithms, the implementation of methods and functions inside the project pipeline became faster, more reliable and multiple pre-processing and post-processing techniques has been tested, following structural changing in the main algorithm.

Therefore, OpenCV library can be considered as an important resource of this work. For that, and for more theoretical purposes, it appropriate to introduce the main feature of that computer vision based library.

First of all, OpenCV was initially developed by Intel, then later maintained by Willow Garage, a robotic research lab and technology incubator and subsequently by Itseez, currently acquired by Intel. Precisely, the library is cross-platform and open-source, under BSD, Berkeley Software Distribution, license. The library was initially created to advance CPU-intensive applications, related to real-time tracing and 3D display walls project. Considering that, it is therefore clear why it is widely exploited when dealing with computer vision correlated works.

As defined in [44], the main targets of the initial project were:

- Accelerate vision research providing open and optimized code for es-

sential vision framework.

- Aim for a more readable and interchangeable code by providing a universal infrastructure to use.
- Enhance vision-based commercial projects by producing a portable and optimized code available with a free license.

Currently, OpenCV library is one of the most used platform by mainly computer software companies. As a matter of fact, it has been calculated that it counts more than 47 thousand people of user community and an evaluated number of downloads surpassing 18 million <sup>4</sup>.

Thus, it has been chosen as one of the main assets of the developed work, because of the aforementioned reasons, and in relation to the available applications, which can be used in multiple computer vision related areas, such as, 2D and 3D feature toolkits, facial recognition systems, mobile robotics, object detection, segmentation and recognition and diverse machine learning fields.

### 3.3.5 CloudCompare 3D editing and processing software

CloudCompare is a software that was exploited during this project almost exclusively for the accurate 3D point cloud visualization, which it can easily provide. Therefore, it can be considered as a side software used in this work, which help to get coherent visualizations of the result, things that results difficult or overcomplicated employing OpenCV functions and C++. On the contrary, with this software was easy to visualize the computed results, and deeply understand if the modifications made in the code works correctly, improving the final results.

Therefore, to provide a brief presentation of this software, it is basically a 3D point cloud processing software. Originally, it was developed to operate comparison between two dense 3D points clouds, such as the ones collected with laser scanner, or between a point cloud and a triangular mesh. On top of that, it has been further expand to a more multi-purposes point cloud processing software, providing multiple advanced algorithms for data handling and processing.

---

<sup>4</sup>Data from <https://opencv.org/about/>

# **Chapter 4**

## **Methods**

A properly planned project requires the analysis of multiple methods and strategies that can be applied during its design. Therefore, this chapter displays, focusing on the multiple aspects of the entire project, the different techniques, encountered revising the broad literature accessible in this field, and the decisions that brought to the actual development of the algorithm, which will be thoroughly described in the chapter 5.

In this chapter, an extensive outline of the work made is not produced, though. Instead, the following sections focus on a concise, but anyway exhaustive, tracing of the methods available for a correct dense disparity estimation and for image data processing. Concurrently, the choices made for the specific aspect of the algorithm are defined and explained.

### **4.1 Standard disparity estimation methods**

An extensive and broad analysis of the standard and the deep learning methods for dense disparity estimation has already been presented in chapter 2, where the review of the literature related to this Master's thesis project has been carried out. Nevertheless, these sections focus on a different aspect of that analysis, which is correlated to the algorithms that turned out to be the most relevant for this work and to the features of some of the current machine learning based techniques, which appear to be applicable to this project.

Considering the standard algorithms for recovering a dense depth image from a stereo pair, the initial approach on which it was decided to focus on for the designing of the project refers to the Semi-Global Matching (SGM) stereo method developed by Hirschmüller [4]. The decision of starting to concentrate on that evaluation of the stereo matching problem was mainly driven

by the recent methods designed for dense disparity estimation. As a matter of fact, most of the algorithms created in the last decade are based on the Hirschmüller idea or, at least, they make reference to several aspects of the functions outlined in [4]. This fact is clearly comprehensible considering that the SGM method was classified as the best algorithm at the time of its publication and it is currently one of the top-ranked algorithm in terms of sub-pixel accuracy and computational time. Hence, taking into account the benchmark data based on two of the most employed dataset, i.e. Middlebury [30] and KITTI [31] datasets, and after an initial general review of the literature publish in this field, it results that the top-performing algorithms use key features of SGM in their main pipeline. Moreover, it is worth to point out that the majority of the most accurate real time deep learning based algorithm tends to adopt the main functions of the SGM model, exploiting neural networks to ensure real time evaluations of the disparities. Therefore, it was initially though that a reasonable option for the developing of an algorithm, which would work in real time providing accurate outcomes, would focus the attention of the Hirschmüller method.

As already introduced in section 2.2, the SGM method combines the idea of the standard local based algorithms, where the main cost calculations are made over limited size windows, and global algorithm, for which the best pixel disparity estimation is based on a global energy function. Thus, mainly because its hierarchical cost calculation, SGM method outperformed the top-ranked algorithm over the dataset [43], proving real-time performance of that dataset images. However, with the current dataset, such as Middlebury 2014 [30], KITTI 2012 [31] and KITTI 2015 [32], the standard Semi-Global Matching algorithm provide optimal results in terms of accuracy but it tends to slightly suffer if computational time is considered. For this reason, the most recent real time methods for dense disparity estimation establish the cost computation part of their pipeline over the SGM idea, although, they exploit hardware efficiency or even neural network structures to reach extremely fast computations.

Thus, considering this project, it was decided to initially build up the main pipeline on the SGM method structure and employ the stereo device designed by the company to reach real time performances, without loosing in accuracy. Furthermore, as previously anticipated, an upstream choice was made. As a matter of fact, the current top-performing algorithms strongly depend on convolutional and deep neural network to achieve fast computations. Contrarily to that, we decided to based the skeleton of our method on the data available through the stereo system to reach a real time implementation. In this manner, all the problem related to the shifting between synthetic and real environment, faced by deep learning algorithms, would be avoided.

The first outline of the algorithm was, hence, partially based on the Semi-Global Matching method [4]. In order to analyse the suitability of the implementation, absolute computational time performances were not initially taken into account. At first it was chosen to focus on the relative efficiency that would be estimated among the different methods used in the principal pipeline.

As aforementioned in chapter 3, these preliminary evaluations has been developed in MATLAB. In that phase an implementation sufficiently similar to the standard SGM method was designed. Thus, the focus was centred on the different types of algorithms used for matching the corresponding pixels and, then, on the part of the pipeline related to the matching and the aggregation costs, as precisely defined by Scharstein and Szeliski in [5]. The adoption of this working strategy was due to the following motivations. First of all, the predominant research aim of this project, whose purpose would most likely further shifts in more market based one, then the need of test the effective performance in accuracy of pure SGM based method and the requirement of analyse the relative efficiency between that algorithm and our method, whose pipeline would be based on the information acquired exploiting the company device.

Considering that a fundamental part of this initial testing phase was focused on the matching cost evaluation, multiple matching algorithm has been tested in order to find the most efficient one mainly in terms of accuracy. Therefore, basing on the work done in [45], [46] and [6], and making reference to the algorithm used for computing visual correspondence described in [12] and [47], the performance of different matching cost algorithms has been analysed.

#### 4.1.1 Census transform

A crucial phase of the SGM algorithm stands in the matching cost computation. As briefly explained in Chapter 2, this cost is evaluated by computing the difference in the value of intensity among corresponding pixels. Therefore, especially when dealing with unexpected luminance variation inside the analysed scene, it comes out that non-parametric local transforms are optimal as the support for that correlation [12]. As a matter of fact, these kind of measurements do not rely on the actual intensity values of the pixels, but on the relative local intensities instead. Therefore, using this kind of models, a considerable number of outliers can be tolerated and improved results are visible especially along object boundaries.

Among the commonly used and best performing non-parametric local transform, census and rank transform are described. Moreover, couple of different

implementations of those two classes of transform are introduced.

It is widely known in computer vision field that correspondence problem is crucial in depth estimation from stereo pairs. Correspondence can be, thus, defined by transforming the images with a specific method and then establishing correlations. In that, the key factor is the transformation, which must tolerate unexpected variation in intensity, for example due to unwanted light source, and in general changes in image bias and gain.

Therefore, non-parametric area-based transform rely on local ordering of intensities and not on the actual intensity values. If an image is considered, and defining as  $p$  a random pixel, its intensity becomes  $I(p)$ . Then, let  $N_d(p)$  be the set of pixel in a certain square neighbourhood of size  $d$ , where  $p$  is the central pixel. Hence, the binary function  $\xi(p, p')$  takes value 1 if  $I(p') < I(p)$ , 0 otherwise. A non-parametric local transform only rely on the set of pixels in the specific neighbourhood. Therefore, the *Census transform*,  $R_\tau(p)$  maps the image subregion surrounding  $p$  to an array of bits representing the set of local pixels whose intensity is less or equal than that of  $p$ . Mathematically, the census transform can be formulated as:

$$R_\tau(p) = \bigotimes_{[i,j] \in D} \xi(p, p + [i, j]) \quad (4.1)$$

where,  $N(p) = p \oplus D$ , with  $\oplus$  is the Minkowski sum and  $D$  the set of displacements, and denoting with  $\otimes$  the concatenation. After the census transform has been applied, the actual comparison among corresponding pixels in done by computing their *Hamming* distance.

Higher accuracy in corresponding pixel matching can be achieved using an extension of the Census transform, defined as center-symmetric Census transform [48]. Therefore, if a square image subregion is taken into account, which has dimensions  $n \times m$ , we already know from equation 4.1 that  $s(u, v) = 0, if u \leq v, 1$  otherwise, where now  $s()$  is the sign function. Precisely, in this notation the census transform can be thus defined as:

$$CT_{m,n}(x, y) = \bigotimes_{i=-n'}^{n'} \bigotimes_{j=-m'}^{m'} s(I(x, y), I(x + i, y + j)) \quad (4.2)$$

where  $n' = [n/2]$ , and  $m' = [m/2]$ . Hence, as shown in Figure, the center-symmetric census transform can be mathematically defined as:

$$CS - CT_{m,n}(x, y) = \bigotimes_{(i,j) \in L} s(I(x - i, y - j), I(x + i, y + j)) \quad (4.3)$$

where  $L = L_1 \cup L_2$ ,  $L_1 = R_{-n',0} \times R_{-m',0} \setminus (0, 0)$ ,  $L_2 = R_{1,n'} \times R_{-m',1}$  and  $R_{a,b} = x \in \mathbb{Z} | a \leq x \leq b$ . Because, only center-symmetric pairs of pixels are

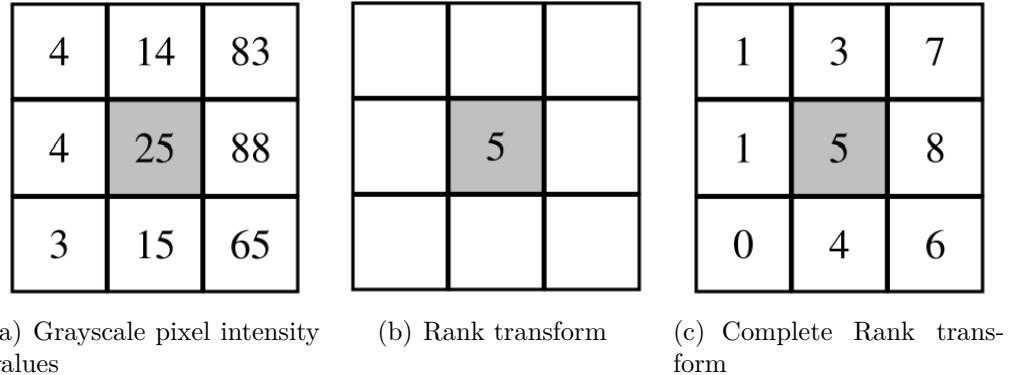


Figure 4.1: Example patch with grayscale pixel intensity values

compared, the formulation in equation 4.3 needs less bit than the one in 4.2 to describe the same patch. As aforementioned, the actual matching cost is then computed using the Hamming distance between corresponding pixels.

### 4.1.2 Rank Transform

Another patch-based pixel descriptor is the Rank transform. As the Census, it is morphologically invariant. Therefore, it is not sensitive to any monotonically increasing of the grayscale intensity values.

The definition of this measure is, actually, quite simple. The pixels in the specific patch considered are ordered basing on the ranking of all grayscale values in that area. Then, the reference pixel *rank encoding* is determined by counting the number of neighboring pixels with a gray value smaller than the considered pixel, that is, basically, its position in the intensity ranking of the patch. Figure 4.1 shows a  $3 \times 3$  example patch where the central pixel, i.e. the reference one, is map to its scalar rank encode. This patch-based descriptor, as the Census transform, was introduced by Zabih and Woodfill [12] and, defining an image pixel with  $P$  and its neighborhood with  $N(P)$ , it can be mathematically expressed by the following equality:

$$R(P) = \|P' \in N(P) | I(P') < I(P)\| \quad (4.4)$$

Looking at the mathematics of the Census and the Rank transforms, it is evident that the former encodes more information.

In addition to the two presented non-parametric local transforms, there is another area based descriptor that need to be mentioned.

The Complete Rank transform can be thought as an improvement of the Rank transform. It was more recently presented by Demetz et al. [47], with

the goal of keep a higher amount of information with respect to its standard version. Actually, it works in the following manner. At first the normal Rank transform is applied to all the elements of the patch. Then, these ranks are concatenated, as it is done for the Census, so that the complete rank descriptor is generated. As visible in Figure 4.1, pixels with the same intensity take the same rank value. Therefore, the encoding for the patch displayed in the image is:

$$\mathbf{enc}_{CRT} = (1, 3, 7, 1, 5, 8, 0, 4, 6)^\top \quad (4.5)$$

#### 4.1.3 Conventional intensity based methods

## 4.2 Deep-learning based methods

As introduced in section 2.2.2 the deep learning method approach for stereo matching was not chosen as the base method for the developing of this project for different reasons, which have already been explained in Chapter 2.

However, the structure of a deep learning based project was extremely useful in the definition of the strategy used for the designing of our work. Fundamental is the novel stereo matching method developed by Poggi et al. [2], who exploited a small quantity of sparse initial depth data to enhance the overall stereo matching problem. In their paper, they presented *Guided Stereo Matching*, an innovative stereo algorithm, which uses a sparse set of accurate depth measurement obtained from reliable hardware system, e.g. a LiDAR, with the aim of improving the general performance of a standard depth estimation algorithm.

Therefore, considering the structure of our system, the strategy used in [2] appears to have valuable insights, which would be exploited in the definition of our own work. Actually, they proposed to use sparsify yet reliable depth measurements for the following tasks: improve the general accuracy of the neural network designed for achieving the depth estimations, reduce the domain shift, i.e. the drops in accuracy that deep learning based methods face when dealing with data from a new environment, and the potential improvement of result when training the network starting from that sparse set of inputs.

Similarly to that approach, in this project, we decide to base our overall estimations on the knowledge of the sparse group of 3D measurements coming from the company device. However, differently from them, the algorithm described in this thesis uses as the standard methods in the stereo matching pipeline, which appear to work better for multiple types of environments.

Thus, taking into account the results provided in [2], where the authors

demonstrate how their strategy outperforms the current state-of-the-art stereo matching methods, we assumed that a reasonable approach to apply would exploit the initial cues coming from the laser grid of points in order to improve the performance of the main algorithm. As a matter of fact, in their paper, Poggi et al. [2] assert that the initial set of estimations can be obtained with whatever device that can give sparse data.

Consequently, during the initial stages of the development some tests have been performed designing a simulated point grid, in order to have a clear outcome of the reliability of this strategy. Hence, employing the Middlebury 2014 dataset, the initial set of clues was designed taking the sparse depth information from the ground truth images available. Obviously, the depth measurements coming from the dataset can be considered highly accurate, differently from the data from the LadiMo device, which would most likely be affected by some sort of noise, mainly Gaussian. Nevertheless, it is actually extremely useful to perform these tests in order to figure out since the first phases of the project, the qualitative performance of the adopted strategy. The details of the conducted tests are widely described in Chapter 5, where some of the temporary results obtained are proposed too.

### 4.3 Pure derivative based data estimation algorithm

As previously introduced, two different algorithm structures have been implemented and compared in this project, in addition to the standard Semi-Global Block Matching (SGBM) algorithm, whose result was used as a baseline for carrying out an appropriate comparison with the novel implementation designed. Therefore, the first one of these methods is closely related to the standard SGM implementation, it exploits the initial knowledge of the sparse depth estimations, however, the core of the algorithm follows the Hirshmüller algorithm [4]. Differently, the other implementation designed is more strictly related to the initial sparse depth clues, and it is focused on those for the overall pipeline of the algorithm. As it will be reported in Chapter 6, this latter approach will provide a faster yet, in general, equally accurate implementation. Therefore, these two designed strategies will be deeply explained in Chapter 5. Beside that, it is worth to propose here a concise outline of both of them, in order to introduce their main features.

Regarding the last introduced method, it highly exploits the information coming from the input sparse clues. Therefore, starting from that initial 3D

point cloud and using a pair of stereo images of the considered scene, a dense point cloud is built by estimating the space positions of an high amount of scene points. The key of this algorithm actually stands on the way in which those estimations are carried out. Thus, considering the point grid given by the stereo device, the local derivatives among neighboring points are evaluated. These will be then used for two main scopes: identify if inside a specific subpatch of the point cloud there could be an edge, that is that points belonging to the same small cluster have indeed highly different values of depth, and for computing the denser point cloud by interpolating the 3D value of neighboring points and exploiting the values of the pre-calculated derivatives for making the interpolation faster.

It can be, hence, easily understood how core of this strategy stands on the local derivatives estimation. As a matter of fact, let us consider a square cluster of 4 neighboring points from the initial point cloud, it is possible to evaluate the *internal* and the *external* derivatives among these points. Specifically, the former are referred to be identified by the vectors that points *inside* the square patch. Instead, the external derivatives are said to be the ones evaluated using the other points, which are neighbors to the patch corners. Figure **PUT FIGURE HERE** shows the aforementioned point cloud patch. Summarizing, the denser 3D point cloud is estimated by filling up each one of the subpatches that can be identified from the initial grid. In order to perform this, the estimations are carried out exploiting the derivative vectors, which are weighted by the distance from the new estimated point and each one of the window corners. This detail, in fact, shows that four temporary estimations are done for each new point. In this way, the best estimation is actually chosen by projecting the evaluated 3D point to the image plane and calculating the matching cost between the corresponding pixels for each temporary estimation. For the matching cost, as it will be more widely explained in Chapter 5, different methods have been investigated. However, the plain absolute difference appeared to be the best in terms of performance for that algorithm. In fact, at the end of the whole computation, similar results have been found with different matching cost algorithms. Hence, in that case, it is worth nothing to use most expensive methods, which will make the computations heavier giving only slightly better results. Besides that, specific penalties have been added on top of the pure matching cost to enhance the distance from the respective window corner in terms of x and y coordinates. Summing up the general outcome gained from this approach, it results to be a lightweight yet effective method. As it will be presented in the Chapter 6, this algorithm provides 3D dense point clouds that can be considered sufficiently accurate for a general use employing a relatively low amount of resources. Despite that, internal parameters of that method can be adjusted,

depending on the user needs, in order to obtain outcomes with a different grade of accuracy. Evidently, this will affect the performance in terms of computational time and used resources, however, this is always considered for every testing environment.

## 4.4 SGM based data estimation algorithm

As aforementioned, the second method that has been designed is based on a more standard approach, closely related to the SGBM algorithm. Basically, it exploits the information coming from the input point grid to make the standard pipeline of the SGBM method faster, which could be extremely expensive when dealing with high density images. Actually, the initial points information is employed to select the disparity levels on which the main algorithm is run. Simply speaking, the final implementation can be considered as a *disparity-wise* Semi-Global Block Matching, where there are not usually continuous changing in the disparity levels, especially if edges are detected inside the specific image patch.

Taking into account the general performance of this approach, it runs slower than the previous strategy. Moreover, the final 3D point cloud shows an higher density, however, looking at the achieved results, the increase in accuracy obtained in this case seems not to be worth it the observed rise in computations.

Therefore, at the reached stage of the project, it appears that a slightly decrease in accuracy can be considered reasonable for a quite high increase in computation speed. However, every use case has to be widely analysed in order to choose the most profitable solution. Additionally, it is necessary to point out that this project should be regarded as still in a researching phase. Multiple tests have been performed with both dataset, i.e. almost perfect, and real images. However, for a good analysis of the specific use case, it is necessary to run various tests on the specific working environment, and eventually, adjusting parameters or side sections of the main pipeline.

Beside that, a further discussion over those topics will be presented in Chapter 7. Anyhow, it was considered helpful for a deeper understanding of the results that will be presented in the Chapter 6, to introduce also here some brief comments over the outlined algorithms.

Hereafter, a short overview on the pre and post-processing techniques employed in the whole project is presented. Further details will be described in the following chapter. Anyway, it is useful to introduce here the main techniques used and especially the differences in performance, which have

been analysed, and also their scope in relation to the final accuracy of the algorithms.

## 4.5 Pre-processing techniques

In the first section of the algorithm pre-processing techniques have turned out to be, not indispensable, but useful for the accuracy of the final results. The focus was especially centred on the noise removal. As a matter of fact, if a stereo pair is given, it is generally affected by Gaussian, i.e. random, noise, which can be due to multiple reasons, such as instability of environmental conditions, lightning, local temperature gradients that can affect the stereo system or errors in the setting of the stereo device.

Therefore, for those reasons and for the purpose of launching the algorithm having the input image as clearer as possible, especially in terms of pixels' intensity, blurring filters have been tested and analysed during the development of this first phase of the pipeline. Specifically, median and bilateral filters occur to be ideal for smoothing away noisy components in the frequency spectrum, keeping the performance of the whole algorithm reasonable.

The details of that analysis are left to Chapter 5. However, it is worth to indicate that, even if heavier from a computational side, the median filter came out as the preferred choice for image blurring when dealing with real stereo pairs. As a matter of fact, when the images from the datasets have been tested, there was not any needs to apply some sort of pre-processing, being them already cleaned from noisy components.

Talking about the real dataset and especially about the initial grid of points obtained from the company stereo device, some pre-processing operations have been performed over those data too. The goal of this process was the requirement of starting the main computations basing on accurate and possibly complete data. Focusing on that, it actually happened that the input point cloud was, in general, not fully complete. Instead, some of the points had missing data. This was probably due to occlusion, in fact most of the *bad* points areas were located on edges or on regions affected by shadows.

Therefore, this problem was tackled in the following way. First of all, all those input data have been analysed and, specifically, the percentage of completely missing values with respect to the total amount of points was calculated. This was performed to obtain an initial estimation of the percentage of errors present in the input data, thus to evaluate the degree of effort that it should be required to have a final accurate result. Precisely, let us think about the case of having an extremely low amount of errors, e.g. the 1.5% of the entire point cloud. Moreover, consider the condition in which these errors

are sparsified and, additionally, that would be sufficiently complicated to fill in those missing data. Thus, in this situation, it would be worth nothing to spend resources to improve the completeness of the input data, reaching an outcome that would be, most likely, only slightly more accurate and for which the whole algorithm would be way heavier and more complicated. On the contrary, a feasible option can be the one of running multiple time a very lightweight algorithm, exploiting the most accurate new estimated points to improve the density of the successive 3D clouds.

For the actual purposes of this Chapter it is unnecessary to present further details of this data processing, which would be more widely analysed in the following sections.

On the other hand, it is convenient to introduce here some features about the post-processing operations, which have been carried out in the last section of the algorithm.

## 4.6 Post-processing techniques

Considering the results achieved with both the Middlebury 2014 dataset and the real environment images an additional blurring process has been run over the final data. Differently from the the smoothing operations employed in the first part of the depth estimation method, in this phase the morphological filters emerge to be the most appropriate trade-off between final result and computational resources.

Also during the designing of this area of the project several tests have been carried out in order to obtain an estimation of the most suitable post-processing technique to run over the final result.

Therefore, considering the disparity images achieved using the stereo pair from the *perfect* dataset, the application of an additional median filter seemed more appropriate. However, for the outcome accessed using images taken with the LadiMo device, the use of morphological filters appear more convenient. Additionally, several of those have been tested. Opening, closing, erosion and dilation have been run over both the raw disparity result and over the raw denser 3D point cloud. Moreover, in order to have a clear idea of the performance of the different kernels, even mixture of those morphology operations have been generated.

At the end, the application of an opening filter gave the most accurate result on the 3D point cloud.

# Chapter 5

## Implementation

The actual implementation of the designed algorithms introduced in Chapter 4 is now extensively described. As anticipated in the previous Chapter, here all the details of the developed phases of the project are deeply analysed and explained. A path similar to the one followed during the actual designing of the algorithms is used for outlining the whole project.

First of all, the preparatory tests performed for evaluating the feasibility of the conceived idea are reported and explained. Subsequently, the focus moves on the two main methods designed, the derivative based one and the strategy based on the standard SGM pipeline. After that, the side phases of the work are presented. The pre and post-processing operations are accurately defined and the comparison among the different filtering operations are carried out.

### 5.1 Preparatory and testing phase on MATLAB

As anticipated in Chapter 4, the first phase of the project can be mainly considered as a researching stage. As a matter of fact, exploiting the stereo images from the Middlebury 2014 dataset, some tests have been performed to obtain an estimation of the feasibility of the conceived idea.

Therefore, as Figure 5.1 displays, ground truth images have been initially used. Thus, a point grid, similar to the one generated with the LadiMo device, has been simulated and overlapped to the input images. In particular, as the difference between Figure 5.1(a) and Figure 5.1(b) illustrates, to have a reliable simulation of the 3D point cloud generated by the device, the only grid that was actually defined was the one over the principal image, i.e. the left one. As a matter of fact, as defined in section 3.2, the hardware exploited in the project is provided with only one laser projector. Therefore, this allows

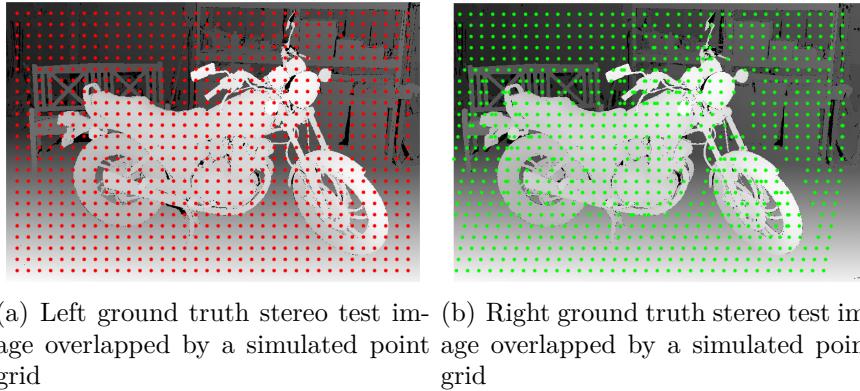


Figure 5.1: Ground truth images from the Middlebury dataset used for the initial testing phase

to utilize the produce point cloud only over one image<sup>1</sup>. After that, those input data, together with the grayscale stereo pair, have been employed in a stereo matching pipeline to recover a disparity map of the analysed scene. In order to acquire useful baseline data regarding the practicability of the thought strategy, tests and analysis over the different areas of the main loop of the algorithm have been investigated. Actually, this research has been performed with a precise method.

First of all the stages of a standard stereo matching method, as defined by Scharstein and Szeliski in [5], have been separately examined. Thus, in order to do this in a reasonable way, the literature regarding both standard and novel stereo matching algorithm has been studied and the main features of the designed algorithm checked out. As an example, if the matching cost phase is taken into account, different approaches are presented in the literature for evaluating the similarity among the image pixel values for the respective disparity levels. As a matter of fact, as introduced in Chapter 2, the Hamming distance between corresponding pixels can be exploited to determine that measure. In relation to that, different algorithms exist to apply a transformation to the input images, making them available for an appropriate implementation of the aforementioned matching cost calculation. For the actual tests carried out, local methods have been mainly analysed, being more appropriate for the type of data available and for the purposes of the designed strategy.

---

<sup>1</sup>Technically speaking the initial cloud of point created through the device can be independently related to both of the image, depending on which is used as the reference one, just applying the correct transformation between the laser system of reference and the reference one

Therefore, as matching cost measures, simple operations have been initially taken into account, such as the sum of absolute differences and the sum of squared differences. Moreover, more accurate methods have been later evaluated, i.e. the Rank transform, and its improved version, the Complete Rank transform, broadly discussed in [47]. Then, the Census transform and the Center-Symmetric Census transform proposed by Spangenberg et al. in [48] have been investigated. Considering all of those algorithm, it resulted that the most suitable methods are equivalently the rank and the census transform. In fact, being these non-parametric transformations, they are not sensitive to changing in lightning inside the environment, thing that usually happen when managing real industrial areas. Considering that, at the end the Center-Symmetric Census transform has been chosen because of its reliability, which is comparable to the aforementioned methods, and for the fact that it uses a lower amount of bits to describe the single pixel value compared to the standard Census transform.

During this phase of testing, the decision was to run a single pipeline of an *SGM-based* algorithm, whose aggregation cost phase would be based on the four orthogonal directions. Moreover, another operative choice was to use the whole image for the complete execution of the designed method. Therefore, it was thought that, at least for this researching phase, it would be meaningless to define a hierarchical implementation of the code, where the image is initially downsampled and subsequently scaled up again in order to make the execution faster, as made by Hermann and Klette in their application of the Semi-Global Matching algorithm [49].

At the end of this *research* section of the project, the results did not show an exact path to follow. Specifically, the disparity image obtained was actually lower in accuracy compared to the method implemented in MATLAB by Eric Psota and available at the website of the Middlebury stereo dataset <sup>2</sup> and to the MATLAB built-in method *disparitySGM*. Moreover, the developed implementation was quite slow, if correlated to the benchmark of most of the standard SGM-based methods.

However, some considerations should be proposed regarding that outcome. First of all, the algorithm designed runs iteratively, that is no parallel loop is coded to make the execution faster. This is, in fact, explained by the decision of being interested in the relative performance among the own designed methods. Furthermore, any sort of GPU enforcing was used. Hence, it was quite expected to not obtain good performance. On the contrary, the thought strategy was evaluated as feasible and, additionally, the exploiting of proper initial data provided by the stereo camera would likely be an enhancement

---

<sup>2</sup>links Middlebury College Stereo

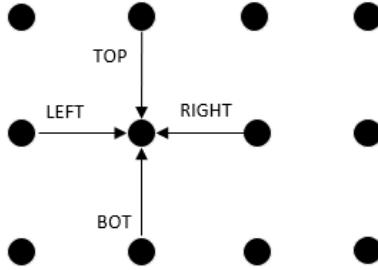


Figure 5.2: Detail of a point grid patch where a set of neighbouring points is highlighted

to the overall performance of the algorithm.

## 5.2 Derivative based method implementation

Verified the feasibility of the considered strategy, the project design shifted to the actual algorithm development. Therefore, exploiting the OpenCV libraries and using the images from the Middlebury 2014 dataset, the derivative based method started to be implemented in a C++ code.

In this initial phase of its designing, the simulated grid based on the ground truth images was still used as the method to obtain the input point cloud data. As a matter of fact, the only practical advantage of employing such simulated data stands on the amount of noise that they contain. As a matter of fact, an appropriate comment to that strategy would be that, once the real data from the device would be used, the result would be probably be worse in terms of overall accuracy. Anyway, at that initial stage of the algorithm development, the interest was focused on comparing the performance of the pure SGM-based and the derivative-based implementations.

### 5.2.1 General features of the method and simulated grid specifications

The so called derivative-based method, which arises to be both efficient and generally accurate, based its strength on the sparse point cloud gained

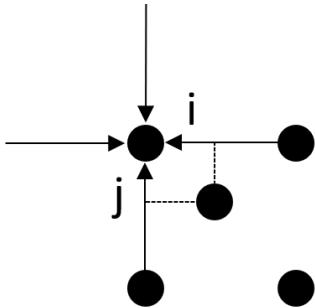


Figure 5.3: Detail of a point grid patch where the estimation of a new point at relative coordinates  $i$  and  $j$  is performed

through the structured light stereo camera.

The main features of that algorithm are now precisely described. Moreover, attention is focused on the conventions utilized to define the main parts of the method and on the functions created for obtaining the 3D point estimations over the initial laser grid. Subsequently, some fundamental sections of the designed implementation are more broadly described, thus to provide to the reader a complete understanding of the work done.

For this reason, the analysis starts by considering a simulated grid of points, which was defined in the C++ code in order to initially mimic the real laser grid generated by the LadiMo device. This approach, as aforementioned, does not invalidate the outcome obtained through all the designing process of the method. Instead, it is a reasonable strategy for achieving some initial temporary results, which can then provide a guidance to the following phases of the whole process.

Therefore, let us consider the entire point cloud as a regular grid where each point is defined by its coordinate in the space. Additionally, it can be also thought that the corresponding 2D coordinates in the reference image plane can be easily retrieved using the camera matrices. Hence, if a neighboring set of four points is taken into account, as highlighted by Figure 5.2, the local derivative vectors for all the considered coordinates can be easily calculated. This estimation of the derivative is extremely useful for the following reasons. First of all, they are, actually, exploited to generate the estimations inside each grid sub-region. Moreover, they can be employed to evaluate if, inside the specific patch of points, there could be an edge of some shape. In this latter case, the magnitude of the Z component of the derivative vector will, in fact, be higher than a set threshold, which can be defined looking at the

average values of the Z coordinate of the grid points.

Therefore, explaining the project design procedure implemented, it can be summarized as follows. Initially, for each one of the points in the grid the derivatives have been calculated with respect to its four neighbors, as the rows in Figure 5.2 underline. This means that each point will have, then, four distinct set of local derivatives, conventionally defined as follows:

- **from top:** derivative between the specific point and its top neighbor;
- **from boottom:** derivative between the specific point and its bottom neighbor;
- **from left:** derivative between the specific point and its left neighbor;
- **from right:** derivative between the specific point and its right neighbor.

In these definition a neighbor to a point is defined with the subsequent rule. Let us consider a point, which is classified through the grid coordinates  $r$ , along the row grid direction, and  $c$ , along the column direction. Its neighboring points are, hence, identified by:

- **top** neighbor: with coordinates  $r - 1$  and  $c$ ;
- **bottom** neighbor: with coordinates  $r + 1$  and  $c$ ;
- **left** neighbor: with coordinates  $r$  and  $c - 1$ ;
- **right** neighbor: with coordinates  $r$  and  $c + 1$ ;

Using that scheme the four derivatives estimated for each point will contain the difference between the actual point ***full location components*** and the ones of the relative neighbor regarding. In the implemented code, the introduced ***full location components*** are in the order: the  $x$  and  $y$  components of the point projected in the principal image plane, therefore they are measured in pixels, the  $X$ ,  $Y$  and  $Z$  space coordinates of the point, in meters, and finally,  $d$  is the disparity value, in pixel, related to the depth of the point, which can be evaluated through equation 1.1.

Therefore, let us now consider to estimate an unknown point inside a sub-region of the whole grid, that was decided to be identified by 4 points, as Figure 5.3 displays. The final 3D coordinates of the new point will be, at first, defined by four different estimations, each one related to the four corners of the grid subwindow, as follows.

$$est_{TL} = TL + i \cdot der_{from-left} + j \cdot der_{from-top} \quad (5.1a)$$

$$est_{TR} = TR + (i - 1) \cdot der_{from-right} + j \cdot der_{from-top} \quad (5.1b)$$

$$est_{BL} = BL + i \cdot der_{from-left} + (j - 1) \cdot der_{from-bottom} \quad (5.1c)$$

$$est_{BR} = BR + (i - 1) \cdot der_{from-right} + (j - 1) \cdot der_{from-bottom} \quad (5.1d)$$

where  $TL$ ,  $TR$ ,  $BL$  and  $BR$  respectively point to the estimation done with respect to the top-left corner of the grid sub-window, the top-right corner, bottom-left and bottom-right corner. Then,  $i$  and  $j$  are the relative  $x$  and  $y$  coordinates of the estimated point, whose values range between 0 and 1. Finally, in the notation employed,  $der_{dir}$  stands for the different derivative vectors related to each corner point and visualized by the arrows in Figure 5.3.

Once, these estimations are achieved exploiting the derivatives, as equations in 5.1 define, the best evaluation is obtained by calculating each matching cost between the corresponding pixels between the stereo images. The matching cost is determined following the subsequent routine.

In relation do this, it should be reported that, due to the researching aspect of this project, lots of changing have been made to the code. Therefore, to provide a clear explanation of the decisions made and the procedure followed, which, otherwise, could appear sometimes winding, all the relevant changing to the functions used are disclosed. Beside this brief comment about the created code, the best, among four, estimation of the new grid point is such evaluated.

The estimation provides the values of the 3D coordinates of the calculated point and its disparity. The point location is, then, projected through the camera matrix to the reference image (usually the left image) pixel coordinates and the corresponding position in the support image is determined using the estimated disparity value. At this point the matching cost is computed as the absolute difference between the intensity values of the analysed pixels in the left and in the right image planes. However, this kind of measure appears to give weak results. As a matter of fact, the main problem was that, in some situations, the estimation labelled as the best one actually contained a value of disparity doubtful, if compared to the one of the point's closest neighbours. Moreover, this generally happened in textureless regions, where it is obviously more difficult to make accurate estimations.

To overcome this problem, a penalty value was added to each partial estimation. Moreover, that value was weighted linearly basing on the distance between the estimated point and the corresponding corner. In this manner, the estimation, related to the disparity and to the coordinate values of the

closest corner, is most likely to be addressed as the best one. Once the best estimation for each new point is calculated, the evaluated values of the 3D point coordinate are processed, applying a morphological filter, in order to remove undesired noise and smooth out the overall data.

The estimation procedure just described is not actually able to handle all kind of situations, concerning the provided input objects. In fact, incorrect values used to be estimated in sub-regions affected by edge features, and thus occlusions.

Therefore, to correctly manage these situations, multiple cases have been defined, regarding the edge cases.

### 5.2.2 Edge cases and penalty values discussion

At first, four main conditions have been distinguished. These comprise the possibility to do not have any type of edge inside a grid subregions, the existence of a vertical edge in the patch, an horizontal edge case and finally a so called *undefined* case has been defined.

Therefore, in each one of these situations, the point estimation procedure differs slightly, so that the aforementioned linear penalty can be more wisely set. Regarding this two distinct values of the penalty have been employed, a lower and a higher value. In fact, because it is not efficiently possible to correctly estimate the position of the potential edge, the choice of the correct estimation has to be somehow driven by exploiting the linear penalties.

Moreover, to identify the correct type of edge, the derivatives have been exploited and, specifically, the magnitude of the  $Z$  component of the vector. Basically, a general edge case, which is equivalent to the *undefined* condition, is initially determined by computing the absolute difference between the  $Z$  components of all the possible couples of corners, i.e. the  $Z$  value of the local derivatives, and comparing that with a pre-determined threshold, which is related to the average depth value of the initial points of the grid.

Therefore, during the algorithm pipeline, the possibility of having an edge feature inside the specific patch is initially checked. Then, if the depth values of the four corner are almost equivalent, this means that the area is actually planar. Thus, in this situation, a fast interpolation method is run, so that the overall computation can be even faster.

On the contrary, if a vertical or an horizontal edge is found, the small and big penalties are used accordingly. For example, if the edge is vertical, the small penalty will be multiplied to the relative  $y$ -coordinate of the estimated point, instead the big penalty will weight the  $x$ -coordinate.

Finally, for the undefined case, the penalty weight is equivalent in both directions,  $x$  and  $y$ , and its magnitude ranges between the two previously defined

penalties.

Those three different edge cases are the one initially tested. However, later appeared that a more meaningful strategy should be based on the definition of two grater categories, which would comprise the cases described above. These two bigger classes describe, thus, **soft** and **strong** edges. In fact, results of tests done with dataset images show that employing a single threshold for the depth difference and focusing only over the edge direction was not entirely correct. As a matter of fact, in a generic scene different types of objects exists. Therefore, adapting the edge analysis to the object distances, for example between the whole foreground and the background and between multiple objects of the foreground, that would give a smoother and more accurate dense 3D point cloud.

### 5.2.3 Derivative computation and sanity check

The core of the estimations are, actually, the local derivatives computed for each point of the grid. Basically, each grid point contains four vectors, which are the local derivatives between the reference point and its four orthogonal neighbors.

Initially the mere derivatives have been calculated as pointing towards the reference point, like Figure 5.2 shows.

However, it came out that this procedure was not enough accurate, to generate a proper dense 3D point cloud. Therefore, a *sanity check* has been later defined inside the derivative calculation pipeline to achieve preciser computation of the local derivatives, especially for the grid sub-patches where edges are likely to be found.

Moreover, a higher accuracy of the final result has been achieved when evaluating, for each grid point, two slightly different types of derivative vectors. Specifically, *internal* and *external* derivatives. This distinction allowed to achieve faster and more accurate results, especially when dealing with edges in the grid sub-region. Precisely, if the presence of an edge is verified for the considered patch, the external (with respect to the sub-window) derivatives were used. Otherwise, exploiting the internal derivatives the computation could be made faster, being them equal in couple among the corners of the patch.

Referring to what introduced at the end of Section 5.2.2, in the latest implementation designed, the estimation process is driven by the edges analysis. As aforementioned, the test results demonstrate that a distinction based only upon the edge shape is not enough to reach a good degree of accuracy. Therefore, the analysis cases have later been defined basing on the strength of the edges between the objects in the scene.

Hence two different thresholds have been so determined. The first is used to identify occlusions that basically occur between foreground objects and the background. These have been designated as *strong* edges. Differently, the other case is specified as the *soft* edges case. This is related to occlusions happen between parts of the same object at different values of depth, as could be between the chin and the neck of a person.

Moreover, on top of this first level of edge selection, the following cases have been then defined, which are correlated to the specific value of the penalty. Precisely, they are *pure vertical* and *pure horizontal* edges, for which it should happen that the depth values of the corners are equals in couple. Then, there are some particular cases, which have the scope of leading the disparity estimation of the new point in a more accurate way. They are: *diagonal top left*, *diagonal top right*, *diagonal bottom left*, *diagonal bottom right*. Furthermore, there is the undefined case, which is triggered when the potential edge found in the subregion has a strange shape.

Taking now into account the actual estimation functions, these are slightly different for the aforementioned edge cases identified. Summing up the overall procedure, it can be said that, the main distinction, which guides the whole process is the first one, that is the one that distinguish among, strong, soft and no edges. Then, for all the subcases, the main difference stands on the values and order of the penalties.

Hence, in the *no edges* case, four temporary estimations are performed for each new grid point by carrying out a linear interpolation, which exploit the *internal* derivative vectors. Then, for the other two main cases, the procedure is generally the same. In fact, the fundamental function of distinguish between those types of occlusions is to make the final outcome smoother.

Thus, when dealing with edges, at first the algorithm checks for each temporary estimation, that is for each subregion corner, if there could also be edges in the neighborhood outside of the region. If this is false, a linear interpolation is computed employing the *external* derivatives. Otherwise, the interpolation is done with the *internal* derivatives and only for the  $x$  and  $y$  pixel positions. Therefore, the depth value is matched with the one of the corresponding corner, thus to enhance the smoothness of the final result.

### 5.3 Semi-Global Matching based method implementation

The other implemented method based on the initial grid of points follows more closely the standard stereo-matching structure. Moreover, it does not

distinguish between ***strong*** and ***soft*** edges, but the different cases are identified only considering the shape of the potential edge present inside the subwindow.

Therefore, this algorithm initially computes the *matching cube cost* for the specific grid patch. The difference between the standard SGM method stands on that cube, whose third direction does not generally contains continuous values of the disparity. On the contrary, those values are defined by the subwindow corners, whose disparity define the levels of the cost cube.

Therefore, if no edge is found, only the matching cost part of the stereo matching algorithm is carried out. Otherwise, the aggregation cost phase is implemented. In this latter case, the cost is actually aggregated only on the direction orthogonal to the shape of identified edge.

At the end of the pipeline simple post-processing operations are performed over the obtained data, as done for the other algorithm. Considering the overall method, it result to be give a denser point cloud compared to the previous method. However, it computational time is extremely higher. This is clearly due to the fact that this algorithm estimates each single pixel inside the specific subregions. Contrarily, in the *lightweight* strategy the number of samples that the user wants to estimate can be adjusted. In this way the trade-off between accuracy and resources utilization can be externally defined.

## 5.4 OpenCV built-in Semi-Global Box Matching algorithm

## 5.5 Pre-processing testing and analysis

The pre-processing phase can be ideally divided in multiple parts. First of all, general operations have been applied for both estimation methods in order to remove noisy components from the stereo pair and from the ground truth images, when exploiting the simulated grid for the estimations. Then, a section of the pre-processing phase can be considered the importing of all the necessary calibration data and the creation of a sort of look-up table, which was used together with the grid of points.

On the contrary, when working with the real data coming from the LadiMo stereo device, some initial operations have been necessary to clean up *bad* point values and for estimating some of the missing data, which would be determined without losing in accuracy.

Regarding the image pre-processing, different noise removal filters have been

tested over database images, in order to understand which would provide the best trade-off in most of the cases. Median and bilateral filters appear to be, almost equivalently, the best choice for smoothing away the noise while loosing the smallest amount of information from the input images.

Considering the others pre-processing operations implemented in the algorithm, they depend on the condition of working or not with the simulated grid.

### **5.5.1 Look-up table and point grid pre-processing**

Therefore, for the initial testing phase of the algorithm designing, when multiple tests have been carried out using the images from the Middlebury dataset, a look-up table has been defined in order to ensure a fast data finding. This structure has been easily built by progressively numbering the simulated grid-points, so that their values correspond to the row index of the data matrix. Considering the real data, instead, the two main initial operations that have been defined in the pipeline regard data analysis. As a matter of fact the laser grid data are not entirely perfect. Contrarily, for the of the points the 3D data are completely missing. This generally happen close to the edges and it is most likely due to occlusions and shadows. Moreover, the used device has some lack of performance when dealing with relatively small objects.

### **5.5.2 Image undistortion and rectification**

## **5.6 Post-processing testing and analysis**

## **5.7 Preparation to future improvements**

# Chapter 6

## Evaluation

You have done your work, but that's<sup>1</sup> not enough.

You also need to evaluate how well your implementation works. The nature of the evaluation depends on your problem, your method, and your implementation that are all described in the thesis before this chapter. If you have created a program for exact-text matching, then you measure how long it takes for your implementation to search for different patterns, and compare it against the implementation that was used before. If you have designed a process for managing software projects, you perhaps interview people working with a waterfall-style management process, have them adapt your management process, and interview them again after they have worked with your process for some time. See what's changed.

The important thing is that you can evaluate your success somehow. Remember that you do not have to succeed in making something spectacular; a total implementation failure may still give grounds for a very good master's thesis—if you can analyze what went wrong and what should have been done.

---

<sup>1</sup>By the way, do *not* use shorthands like this in your text! It is not professional! Always write out all the words: “that is”.

# **Chapter 7**

## **Discussion**

At this point, you will have some insightful thoughts on your implementation and you may have ideas on what could be done in the future. This chapter may be combined together with the evaluation chapter. All the new insights and findings are given here! This chapter is a good place to discuss your thesis as a whole and to show your professor that you have really understood some non-trivial aspects of the methods you used...

# **Chapter 8**

## **Conclusions**

Time to wrap it up! Write down the most important findings from your work. Like the introduction, this chapter is not very long. One to two (never over three) pages might be a good limit. Still, the chapter gives the background, goals, content, and the findings. However, all that should already be in the previous chapters. This is just a summary (as are the abstract and the introduction).

For making PDF/A version requested by the Aalto Library, open the end result pdf file in Acrobat and store it as PDF/A. Then verify the result (everything should be fine, at least as PDF/A-2b version works).

Congratulations, your thesis is ready and it looks beautiful!

# Bibliography

- [1] A. Seki, M. Pollefeys, T. Corporation, E. T. Zürich, and Microsoft, “SGM-Nets: Semi-global matching with neural networks,” *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, vol. 2017-Janua, no. 1, pp. 6640–6649, 2017.
- [2] M. Poggi, D. Pallotti, F. Tosi, and S. Mattoccia, “Guided Stereo Matching,” 2019.
- [3] A. Tonioni, F. Tosi, M. Poggi, S. Mattoccia, and L. D. Stefano, “Real-Time Self-Adaptive Deep Stereo,” pp. 195–204, 2020.
- [4] H. Hirschmüller, “Stereo processing by semiglobal matching and mutual information,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, no. 2, pp. 328–341, 2008.
- [5] D. Scharstein, R. Szeliski, and R. Zabih, “A taxonomy and evaluation of dense two-frame stereo correspondence algorithms,” *Proceedings - IEEE Workshop on Stereo and Multi-Baseline Vision, SMBV 2001*, no. December, pp. 131–140, 2001.
- [6] J. Ko and Y. S. Ho, “Stereo matching using census transform of adaptive window sizes with gradient images,” *2016 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference, APSIPA 2016*, pp. 2–5, 2017.
- [7] S. Birchfield and C. Tomasi, “Depth discontinuities by pixel-to-pixel stereo,” *International Journal of Computer Vision*, vol. 35, no. 3, pp. 269–293, 1999.
- [8] A. Klaus, M. Sormann, and K. Karner, “Segment-based stereo matching using belief propagation and a self-adapting dissimilarity measure,” *Proceedings - International Conference on Pattern Recognition*, vol. 3, pp. 15–18, 2006.

- [9] V. Kolmogorov and R. Zabih, “Computing visual correspondence with occlusions using graph cuts,” *Proceedings of the IEEE International Conference on Computer Vision*, vol. 2, pp. 508–515, 2001.
- [10] R. Szeliski, “Computer vision: algorithms and applications,” *Choice Reviews Online*, vol. 48, no. 09, pp. 48–5140–48–5140, 2011.
- [11] R. Hartley and A. Zisserman, “Multiple view geometry in computer vision. cambridge university press, isbn: 0521540518,” 2004.
- [12] R. Zabih and J. Woodfill, “Non-parametric local transforms for computing visual correspondence,” *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 801 LNCS, pp. 151–158, 1994.
- [13] Z. F. Wang and Z. G. Zheng, “A region based stereo matching algorithm using cooperative optimization,” *26th IEEE Conference on Computer Vision and Pattern Recognition, CVPR*, 2008.
- [14] Q. Yang, L. Wang, R. Yang, H. Stewénius, and D. Nistér, “Stereo matching with color-weighted correlation, hierarchical belief propagation, and occlusion handling,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, no. 3, pp. 492–504, 2008.
- [15] M. Bleyer, M. Gelautz, C. Rother, and C. Rhemann, “A stereo approach that handles the matting problem via image warping,” in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 501–508, IEEE, 2009.
- [16] D. Hernandez-Juarez, A. Chacon, A. Espinosa, D. Vazquez, J. C. Moure, and A. M. Lopez, “Embedded real-time stereo estimation via Semi-Global Matching on the GPU,” *Procedia Computer Science*, vol. 80, pp. 143–153, 2016.
- [17] A. Kuzmin, D. Mikushin, and V. Lempitsky, “End-to-End learning of cost-volume aggregation for real-time dense stereo,” *IEEE International Workshop on Machine Learning for Signal Processing, MLSP*, vol. 2017-Septe, pp. 1–6, 2017.
- [18] E. S. Gastal and M. M. Oliveira, “Domain transform for edge-aware image and video processing,” *ACM Transactions on Graphics*, vol. 30, no. 4, 2011.

- [19] C. C. Pham and J. W. Jeon, "Domain transformation-based efficient cost aggregation for local stereo matching," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 23, no. 7, pp. 1119–1130, 2013.
- [20] J. Žbontar and Y. Lecun, "Stereo matching by training a convolutional neural network to compare image patches," *Journal of Machine Learning Research*, vol. 17, pp. 1–32, 2016.
- [21] J. Žbontar and Y. Le Cun, "Computing the stereo matching cost with a convolutional neural network," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 07-12-June, no. 1, pp. 1592–1599, 2015.
- [22] N. Mayer, E. Ilg, P. Hausser, P. Fischer, D. Cremers, A. Dosovitskiy, and T. Brox, "A Large Dataset to Train Convolutional Networks for Disparity, Optical Flow, and Scene Flow Estimation," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2016-Decem, pp. 4040–4048, 2016.
- [23] R. Haeusler, R. Nair, D. Kondermann, C. Mostegel, M. Rumpler, F. Fraundorfer, H. Bischof, M. G. Park, K. J. Yoon, I. H. Md Yusof, M. An, M. H. Barghi, W. Luo, A. G. Schwing, Z. Chen, X. Sun, L. Wang, Y. Yu, and C. Huang, "A Deep Visual Correspondence Embedding Model," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 07-12-June, no. i, pp. 885–894, 2015.
- [24] A. Spyropoulos, N. Komodakis, and P. Mordohai, "Learning to detect ground control points for improving the accuracy of stereo matching," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. i, pp. 1621–1628, 2014.
- [25] A. Seki and M. Pollefeys, "Patch based confidence prediction for dense disparity map," *British Machine Vision Conference 2016, BMVC 2016*, vol. 2016-Septe, no. c, pp. 23.1–23.13, 2016.
- [26] X. Hu and P. Mordohai, "A quantitative evaluation of confidence measures for stereo vision," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 11, pp. 2121–2133, 2012.
- [27] M. Poggi, F. Tosi, and S. Mattoccia, "Quantitative Evaluation of Confidence Measures in a Machine Learning World," *Proceedings of the IEEE International Conference on Computer Vision*, vol. 2017-Octob, pp. 5238–5247, 2017.

- [28] M. Poggi and S. Mattoccia, “Learning from scratch a confidence measure.,” in *BMVC*, 2016.
- [29] M. G. Park and K. J. Yoon, “Leveraging stereo matching with learning-based confidence measures,” *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 07-12-June, pp. 101–109, 2015.
- [30] D. Scharstein, H. Hirschmüller, Y. Kitajima, G. Krathwohl, N. Nešić, X. Wang, and P. Westling, “High-resolution stereo datasets with subpixel-accurate ground truth,” *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 8753, no. 1, pp. 31–42, 2014.
- [31] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, “Vision meets robotics: The kitti dataset,” *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1231–1237, 2013.
- [32] M. Menze and A. Geiger, “Object scene flow for autonomous vehicles,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3061–3070, 2015.
- [33] R. Ohlander, K. Price, and D. R. Reddy, “Picture segmentation using a recursive region splitting method,” *Computer Graphics and Image Processing*, vol. 8, no. 3, pp. 313–333, 1978.
- [34] R. M. Haralick and L. G. Shapiro, “Image segmentation techniques,” *Computer vision, graphics, and image processing*, vol. 29, no. 1, pp. 100–132, 1985.
- [35] D. Comaniciu and P. Meer, “Mean shift: A robust approach toward feature space analysis,” *IEEE Transactions on pattern analysis and machine intelligence*, vol. 24, no. 5, pp. 603–619, 2002.
- [36] D. Cremers, M. Rousson, and R. Deriche, “A review of statistical approaches to level set segmentation: integrating color, texture, motion and shape,” *International journal of computer vision*, vol. 72, no. 2, pp. 195–215, 2007.
- [37] A. Balke and M. Isard, “Active contours, the application of techniques from graphics, vision, control theory and statistics to visual tracking of shapes in motion,” 1998.

- [38] M. Kass, A. Witkin, and D. Terzopoulos, "Snakes: Active contour models," *International journal of computer vision*, vol. 1, no. 4, pp. 321–331, 1988.
- [39] E. N. Mortensen and W. A. Barrett, "Intelligent scissors for image composition," in *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pp. 191–198, 1995.
- [40] L. Vincent and P. Soille, "Watersheds in digital spaces: an efficient algorithm based on immersion simulations," *IEEE Transactions on Pattern Analysis & Machine Intelligence*, no. 6, pp. 583–598, 1991.
- [41] G. Mori, X. Ren, A. A. Efros, and J. Malik, "Recovering human body configurations: Combining segmentation and recognition," in *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, vol. 2, pp. II–II, IEEE, 2004.
- [42] J. Shi and J. Malik, "Normalized cuts and image segmentation," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 22, no. 8, pp. 888–905, 2000.
- [43] D. Scharstein and R. Szeliski, "High-accuracy stereo depth maps using structured light," in *2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings.*, vol. 1, pp. I–I, IEEE, 2003.
- [44] G. Bradski and A. Kaehler, *Learning OpenCV: Computer vision with the OpenCV library.* " O'Reilly Media, Inc.", 2008.
- [45] H. Hirschmüller and D. Scharstein, "Evaluation of cost functions for stereo matching," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2007.
- [46] S. Patil, J. S. Nadar, J. Gada, S. Motghare, and S. S. Nair, "Comparison of Various Stereo Vision Cost Aggregation Methods," *International Journal of Engineering and Innovative Technology*, vol. 2, no. 8, pp. 222–226, 2013.
- [47] O. Demetz, D. Hafner, and J. Weickert, "The complete rank transform: A tool for accurate and morphologically invariant matching of structures," *BMVC 2013 - Electronic Proceedings of the British Machine Vision Conference 2013*, 2013.

- [48] R. Spangenberg, T. Langner, and R. Rojas, “Weighted semi-global matching and center-symmetric census transform for robust driver assistance,” *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 8048 LNCS, no. PART 2, pp. 34–41, 2013.
- [49] S. Hermann and R. Klette, “Iterative semi-global matching for robust driver assistance systems,” *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 7726 LNCS, no. PART 3, pp. 465–478, 2013.

# Appendix A

## First appendix

