UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE INGENIERÍA ELÉCTRICA

DISPARITY ESTIMATION FOR THE STEREO MATCHING PROBLEM WITH OUTDOOR MOVING TRUCKLOAD IMAGES

MEMORIA PARA OPTAR AL TÍTULO DE INGENIERA CIVIL ELÉTRICA

CAMILA BEATRIZ GÓMEZ NAZAL

PROFESOR GUÍA:
RODRIGO PALMA AMESTOY

PROFESORA CO-GUÍA:
MARIE GONZÁLEZ INOSTROZA

MIEMBROS DE LA COMISIÓN:
MARTIN ADAMS

SANTIAGO DE CHILE
2022

RESUMEN DE LA MEMORIA PARA OPTAR
AL TÍTULO DE INGENIERA CIVIL ELÉCTRICA
POR: CAMILA BEATRIZ GÓMEZ NAZAL
FECHA: 2022
PROF. GUÍA: RODRIGO PALMA AMESTOY

ESTIMACIÓN DE DISPARIDAD PARA EL PROBLEMA DE *STEREO MATCHING* CON IMÁGENES DE CAMIONES DE CARGA EN MOVIMIENTO Y AL AIRE LIBRE

El *Stereo Matching* (o visión estéreo) es un campo de la visión computacional que ha estado recibiendo bastante atención durante las ultimas décadas, debido a su gran rango de aplicaciones y versatilidad. Aborda el problema de la reconstrucción 3D y la estimación de profundidad (disparidad).

El problema de estimación de disparidad se puede resolver de variadas formas, una de ellas es una solución de tipo bloques, en la cual el par de imágenes de entrada pasa por distintas etapas. La primera de ellas, llamada etapa de cálculo del *Matching Cost*, resulta ser una de las más importantes, debido a que determina los pares de píxeles que son más similares entre las imágenes izquierda y derecha. La función de *Matching Cost* puede ser una función tradicional que trabaja píxel a píxel o bien una basada en *Deep Learning*, que es el enfoque usado en el estado del arte actualmente.

En este contexto, la empresa *Woodtech* presenta la necesidad de desarrollar un algoritmo de visión estéreo que sea capaz de resolver el problema de estimación de disparidad, usando pares de imágenes estéreo, específicamente de camiones de carga en movimiento y en condiciones exteriores. Esta memoria contribuye en la implementación del algoritmo requerido, incluyendo en la implementación dos tipos de funciones de *Matching Cost*, con la intención de comparar un enfoque tradicional versus uno con *Deep Learning*.

Los algoritmos implementados son numéricamente evaluados con *Datasets* de interiores (*indoor*) y exteriores (*outdoor*), lo cual provee un buen punto de partida para saber las fortalezas y debilidades de cada algoritmo.

Los resultados prueban la superioridad del enfoque con *Deep Learning* (versus el tradicional escogido) cuando las imágenes están en condiciones *outdoor*, pero también propone un desafío para que el tiempo de ejecución sea manejable en el caso de una aplicación en tiempo real. Sin embargo, el enfoque tradicional usado mostró ser mejorado al aplicar un paso de pre procesamiento a las imágenes de entrada, logrando ser casi tan bueno como el enfoque *Deep Learning*, y con mucho menos tiempo de ejecución.

Los resultados obtenidos son un primer paso para la empresa en la visión estéreo, permitiéndoles tener un algoritmo flexible con dos posibilidades de cálculo de *Matching Cost*, además de un registro numérico de la performance de cada algoritmo, lo que les servirá para tomar buenas decisiones a futuro a la hora de estimar la disparidad de sus imágenes.

RESUMEN DE LA MEMORIA PARA OPTAR
AL TÍTULO DE INGENIERA CIVIL ELÉCTRICA
POR: CAMILA BEATRIZ GÓMEZ NAZAL
FECHA: 2022
PROF. GUÍA: RODRIGO PALMA AMESTOY

DISPARITY ESTIMATION FOR THE STEREO MATCHING PROBLEM WITH OUTDOOR MOVING TRUCKLOAD IMAGES

Stereo matching (or stereo vision) is a field of computer vision that has been getting attention over the last decades, because of its wide range of applications and versatility. It addresses the problem of 3D reconstruction and depth (disparity) estimation.

The disparity estimation problem can be solved in numerous ways, one them is a block type solution, in which the input pair passes through stages. The first stage, called the matching cost computation stage, happens to be one of the most important, because it determines the pairs of pixels that are the most similar between left and right images. The matching cost function may be a traditional pixel-wise technique, or a deep learning based function, which is currently the state of the art approach for computing the matching cost.

On this context, the company *Woodtech* is in need of a stereo vision algorithm that is capable to solve the disparity estimation problem, using a pair of stereo images, specifically, of moving truckload images, in an outdoor environment. This memoir contributes in the implementation of such algorithm, and goes even further by implementing two different matching cost functions, with the intention of comparing a traditional v/s a deep learning approach.

The implemented algorithms are numerically evaluated with both an indoor and outdoor dataset, providing a good starting point for knowing each of the algorithms strength and downfalls.

The results prove the superiority of a deep learning approach (v/s the traditional pixel wise technique chosen) when the images are in outdoor conditions, but also sets a challenge to make the execution time manageable for a real time application. Nevertheless, the traditional approach used, showed to be improved when the input images were pre processed, being almost as good as the deep learning technique, and much less time consuming.

The obtained results are a first step for the company in the field of stereo vision, allowing them to have a flexible algorithm with two possible matching functions, and a record of the measured accuracy of each algorithm, which allows them to make a good decision in the future when it comes to estimating the disparity of their images.

Dedicado a mis padres, Beatriz y Andrés.

Agradecimientos

Quisiera partir agradeciendo a mi familia, en especial a mi mamá, quien siempre ha estado ahí para mostrarme su apoyo y aliento en todos mis proyectos. A mi papá también, por ayudarme en el camino del aprendizaje desde muy temprana edad.

En segundo lugar agradecer a mis amigos, Santiago, Trinidad, Ignacia por distraerme en los momentos necesarios. Y a Nicolás, por darme su apoyo incondicional.

Por último, agradecer a Rodrigo y Marie, quienes me motivaron y ayudaron desde el inicio, asi como también me dieron la confianza que necesitaba. Por supuesto, muchas gracias también a los profesores integrantes de la comisión, por brindar disposición en la corrección de este trabajo.

Table of Contents

Table Index

Illustration Index

Chapter 1

Introduction

1.1. Brief description of *Woodtech*

Woodtech is a company of the forestry and mining industry that develops automatic measurement systems. One of its main products is a system named *Logmeter*, that is capable of computing high precision volume measurements and estimate biometrics characteristics of timber loaded trucks. Their system uses cutting-edge laser technology and delivers a measurement without direct operator intervention [1].

In the need of a less expensive measurement system for timber loaded trucks, the company decides to develop a new product called *Instalog*, which is still in a developing stage. One of its features is that it contains two stereo cameras installed. At the moment, they are mainly used to obtain images of the timber loaded trucks, and then use them as a dataset to train a log detector neural network, with the means of counting the number of logs present in a truckload. A secondary function of those stereo cameras is to provide images to make depth estimations, with the intention of measuring the logs characteristics, such as their diameter. The stereo cameras used can be seen in Figure 1.1, and their placement in Figure 1.2.

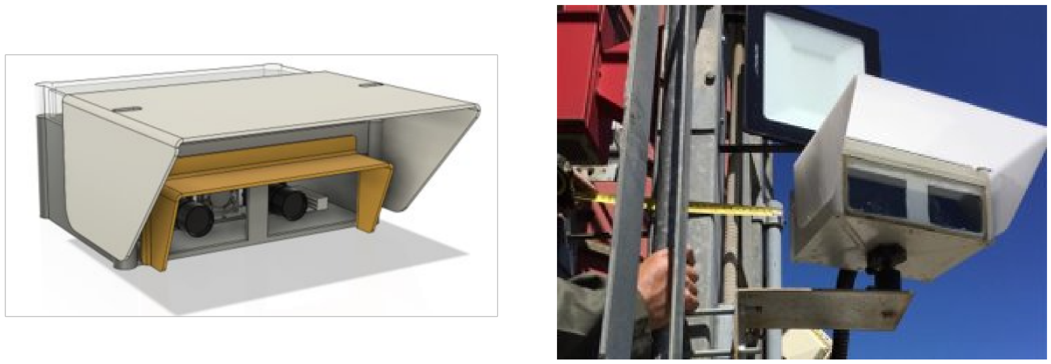


Figure 1.1: Stereo camera used for *Instalog*.

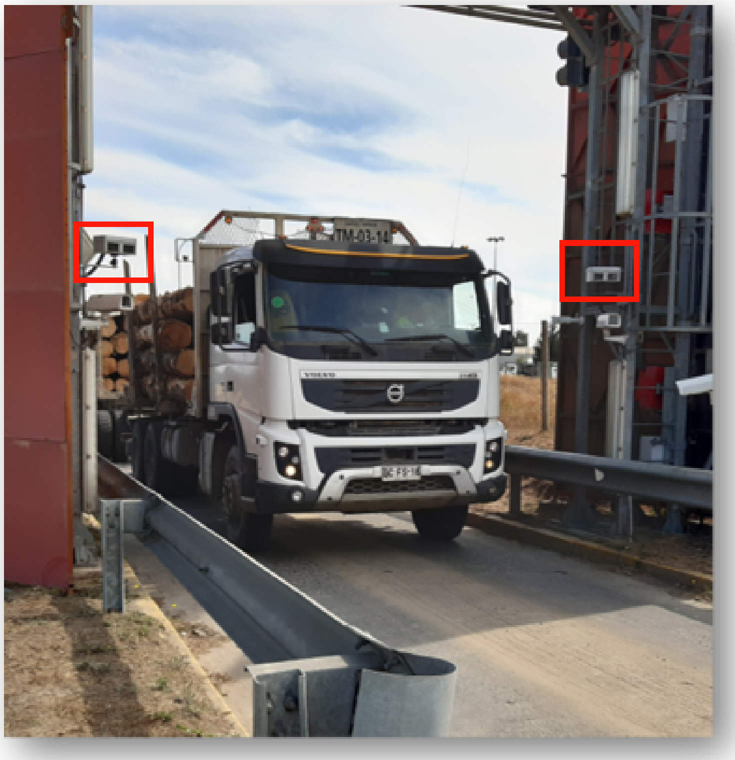


Figure 1.2: Stereo camera placement on *Instalog*.

1.2. Motivation

When it comes to the disparity estimation problem, the company has made attempts at solving it with the use of an external library, in this case, the OpenCV library [2]. The results led to a point where no more changes could be made to improve them, because the specific function used only allowed to change certain parameters. This situation ended up in the idea of implementing a stereo algorithm from scratch, giving the freedom of trying different algorithms for each step of the stereo algorithm, for example, using a neural network to find matches between images, because of their growing success at solving the stereo depth estimation problem [3].

This memoir will contribute in implementing the stereo algorithm required for the truck images, considering the outdoor environment and illumination issues that they might suffer from, with the freedom of being able to change every detail needed for better performance. It will also allow the company to have a base for solving bigger problems like 3D reconstruction of the truckloads, as well as diameter measuring of the trunks.

1.3. Problem definition

Given a pair of grayscale stereo rectified images (i.e objects are row aligned, explained in chapter 2), the objective of this memoir is to solve the disparity map estimation problem, in other words, to find how much the pixels have shifted from one image to another, and therefore be able to obtain depth estimations of the objects in the scene. The main idea is to get an ouput disparity map like the one shown in Figure 1.3.

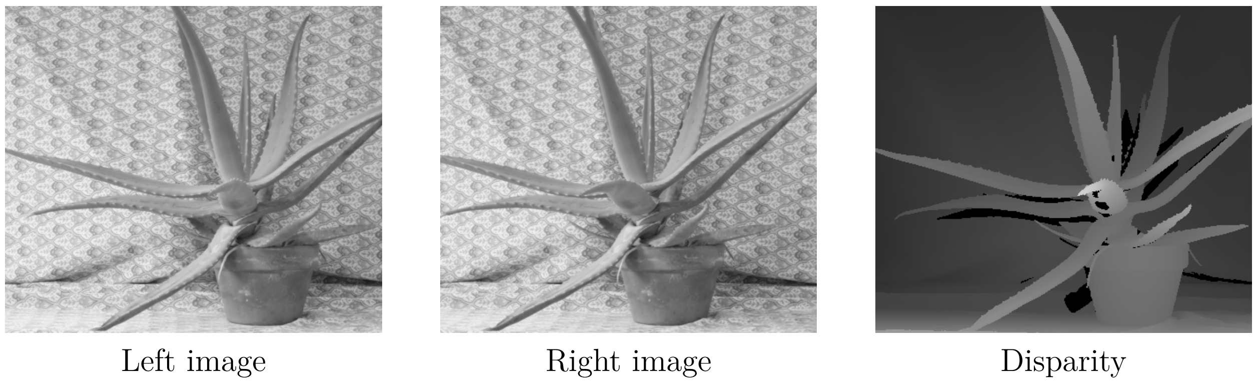


Figure 1.3: Sample of the Middlebury 2006 dataset [4].

Moreover, two attempts will be made to solve the problem by using different matching cost functions, considering that the input images are hard by nature. They will be captured by two stereo cameras (the ones explained in the previous section), and they can be taken at any time of day. This makes the problem challenging since the images might contain illumination differences between them, exposure problems, and also some of them might be in poor lighting conditions. In Figure 1.4 it is possible to see an example of the image stereo pairs that will be used as input for the stereo algorithm. Note that this example is a raw image pair, and is not row aligned yet.

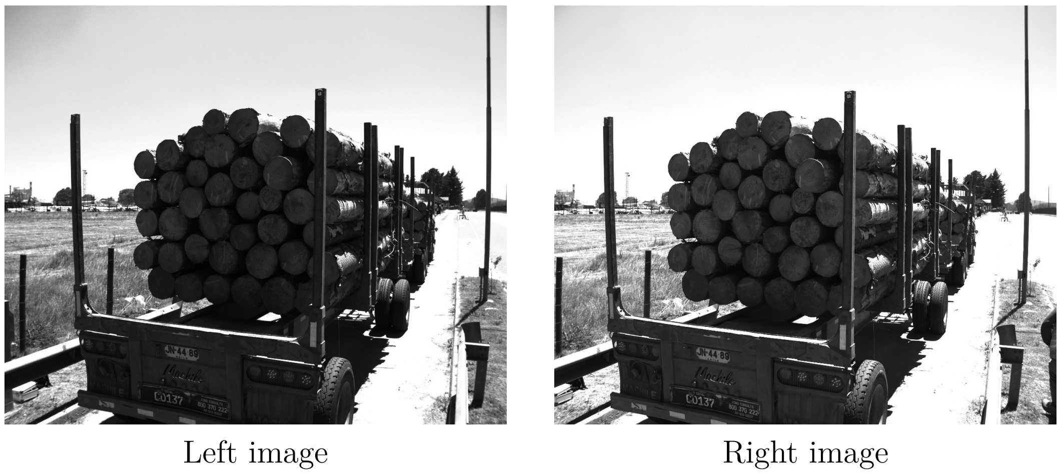


Figure 1.4: Truck image stereo pair.

The performance of the algorithm will be evaluated using stereo datasets, in which a ground truth disparity is known. The KITTI (Karlsruhe Institute of Technology and Toyota

Technological Institute at Chicago) dataset [5] for autonomous driving will be used for this, as well as the Middlebury dataset [6]. Lastly, it is important to mention that there is no ground truth disparity for the truck images.

1.4. Goals

The most important goal is to implement an algorithm that is capable to derive a disparity map, with the timber loaded truck images, without using an external library. In function of the objective just mentioned, some specific goals will be set in order to achieve the desired result:

1. To create a benchmark using two types of matching cost functions, specifically, using traditional match cost functions and also with deep learning. Therefore, depending on the image characteristics, it can be known which matching function is the best to use.
2. To evaluate the output disparity maps of the algorithms with two datasets, indoor and outdoor, and realize which one is best for each environment.
3. To train a convolutional neural network (CNN) and use it as matching cost function, providing a first architecture that can easily be modified to improve the results with the truck images.
4. To implement the algorithm with an object oriented architecture, making it easy to apply changes to the algorithm and its parameters. This way, the provided code can be maintainable and easy to understand for other developers.

1.5. Document structure

First, on chapter 2, a review of the basic concepts of stereo vision will be written, giving the reader a background to understand the next chapters.

Chapter 3 will deepen the understanding of the problem, by detailing some of the literature's proposed solutions on disparity estimation.

On chapter 4 it will be shown what tools have been taken from the previous chapter, to build the algorithm that will be used to solve the problem just explained.

Next, chapter 5 will describe the datasets used to test the accuracy of the algorithm and how it will be measured. Also, details of the truckload images will be shown.

Chapter 6 will display the resulting disparity maps on all the datasets used, as well as giving the error measures for the algorithm, with proper analysis of each section.

Finally, conclusions of the entire work will be drawn, stating what was achieved and what needs further refinement.

Chapter 2

Basic Concepts

2.1. Stereo vision

Stereo Vision (also known as Stereo Matching) is an area of Computer Vision that addresses the problem of depth estimation and 3D reconstruction. A stereo vision system consists of a stereo camera, i.e, two cameras placed horizontally. The images captured simultaneously by these cameras are then processed for the recovery of visual depth information [7]. The challenge is to determine the difference in position in the two views of the images, in other words, to derive a map that shows how much each pixel has shifted between the left and right image (also known as disparity map).

Its worth mentioning that in the disparity estimation problem, the number of calculations required increases with an increasing number of pixels per image, which causes the problem to be computationally complex [8]. Nevertheless, improvements have been made in this field with recent advances in hardware technology, allowing to achieve real time processing [9].

Stereo Matching is still considered to be an open problem due to the difficulties of matching in outdoor conditions (illumination differences) and textured environments. Furthermore, real time computations are often required, which makes the problem even more challenging. Stereo vision applications include autonomous driving, robotic navigation, 3D modeling, gaming and animation, etc.

2.2. Stereo camera and disparity

A stereo camera is composed of two traditional cameras (in the simple case) placed horizontally, in other words, one on the left and one on the right. The two images captured simultaneously by these cameras are then processed to obtain the depth information of the objects within the scene.

Figure 2.1 is a diagram representing a stereo camera. The optical axes of the left and right camera are parallel and represented by *lens 1* and *lens 2*, respectively. (x_l, y_l) are the coordinates of point (x, y, z) represented in the left image plane. Same goes for (x_r, y_r) but with the right image plane.

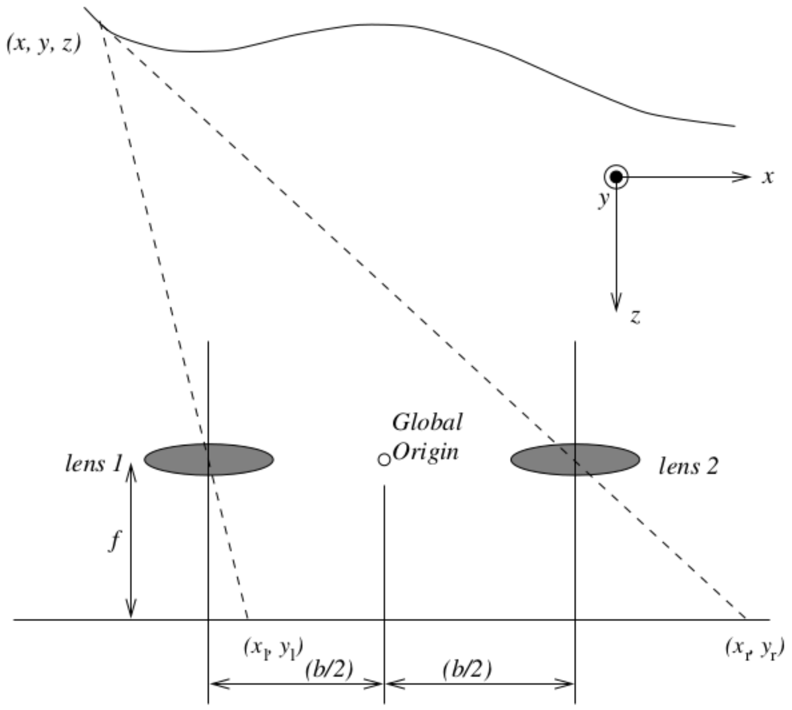


Figure 2.1: Stereo vision system [10].

Calculating x_l and x_r with respect to centers of left and right lenses respectively, it is possible to obtain:

$$x_l = \frac{f(x + \frac{b}{2})}{z} \tag{2.1}$$

$$x_r = \frac{f(x - \frac{b}{2})}{z} \tag{2.2}$$

Then, defining disparity (d) as the difference between image coordinates we obtain:

$$d \coloneqq x_l - x_r = \frac{b \cdot f}{z} \tag{2.3}$$

Therefore, knowing the disparity of an object, the baseline b and focal length f of the stereo system, it is possible to compute its depth (z) with respect to the Global Origin showed in Figure 2.1. It can also be derived from equation 2.3 that a smaller disparity value means the object is farther away from the stereo camera, and for bigger values the object is closer. If the disparity value were to be zero, then the depth would be infinite, so a disparity of zero is not considered valid.

2.3. Epipolar constraint

Given two cameras with optical centers O_L and O_R as shown in Figure 2.2, the epipolar constraint states that for a given point x_L in the left image, its corresponding point in the right image is constrained to lie on a line called the epipolar line of x_L, which is the red line showed in the diagram below (represented by $\overline{e_R x_R}$). The epipolar line of x_L is obtained by intersecting the right image plane with the epipolar plane, defined by x_L, O_L and O_R (showed in green).

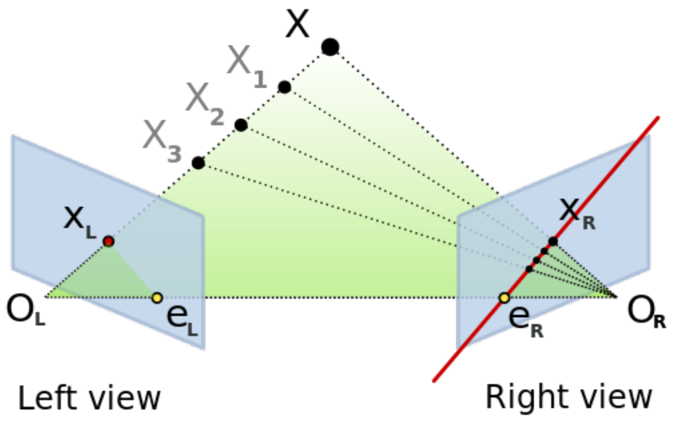


Figure 2.2: Epipolar geometry [11].

2.4. Rectification and Calibration

In order to simplify the searching problem of matching points in the epipolar lines to only one dimension, the input images can be warped in a way so that both left an right image planes become the same. Consequently, all epipolar lines become horizontally aligned and therefore parallel to the axis defined by O_L and O_R (see Figure 2.3). This way, two matching points will always lie on the same horizontal line (or row). The process just explained is called Rectification, and it can be implemented by applying 2D projective transforms, or homographies, to each input image [12] (see Figure 2.4).

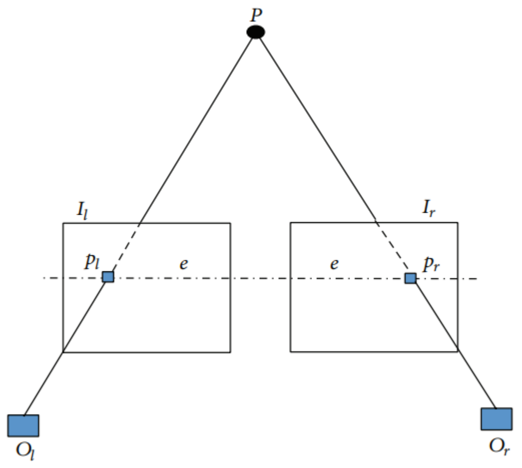
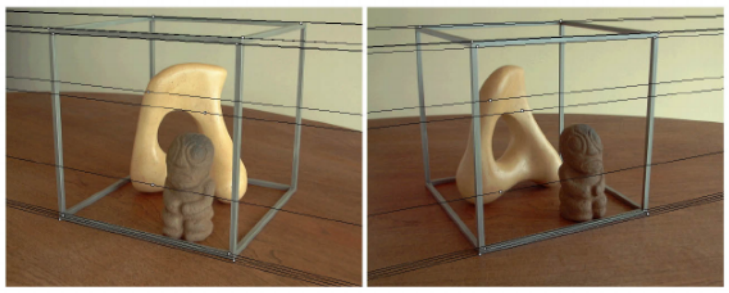


Figure 2.3: Epipolar geometry after rectification [7]. Left and right camera image planes are now the same.



Original image pair overlayed with some epipolar lines

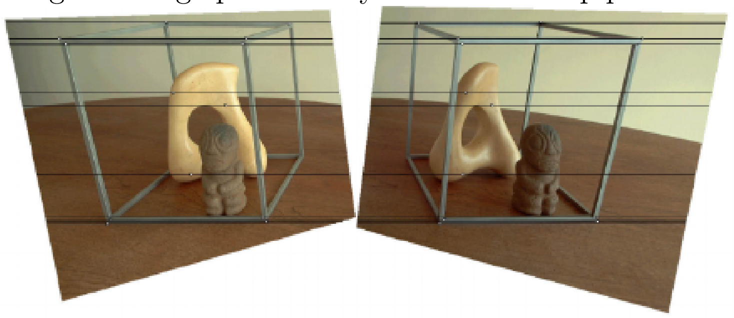
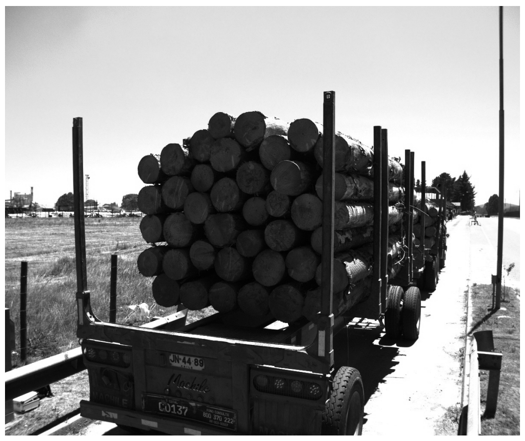


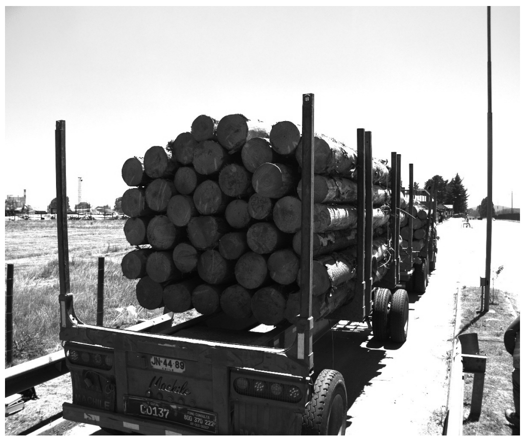
Image pair after rectification. Epipolar lines become horizontally aligned

Figure 2.4: Example of rectification [12].

Calibration will also be applied to the images to eliminate the distortion introduced by the cameras, which depends on the intrinsic parameters of them (focal length and optical center). In this memoir, only calibrated and rectified images will be used to test the algorithms. In Figure 2.6 is an example of a rectified and calibrated stereo image pair that will be used.



Left image



Right image

Figure 2.5: Raw stereo pair.

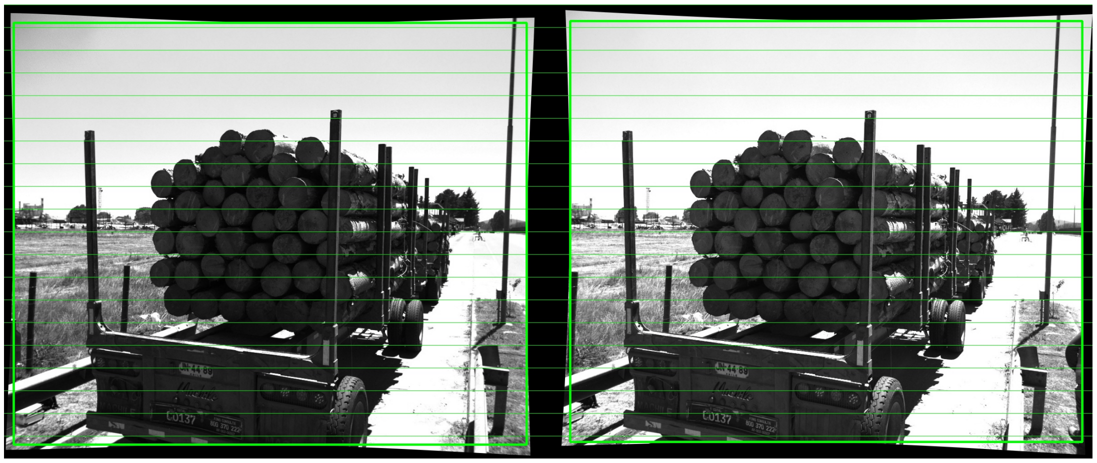


Figure 2.6: Figure 2.5 after rectification and calibration. Green lines correspond to the epipolar lines.

Chapter 3

State of the art

3.1. Standard stereo algorithm

According to Scharstein and Szeliski [6], the basic structure of a stereo algorithm follows four steps, as shown in Figure 3.1. The first step, known as the matching cost computation, determines a similarity value between a pair of pixels and a specific disparity, ending up with a 3D cost matrix. Then, cost aggregation is implemented to add more information from neighboring cost, to the cost matrix. The third step is when the best cost is chosen (of the cost matrix), i.e. the best disparity (among the disparity range) is chosen. Finally, once a disparity image has been obtained, disparity refinement is done for removing peaks, checking the consistency and interpolation of gaps.

The input images, obtained from a stereo camera, are assumed to be rectified and calibrated. The image pair will pass through all of the blocks sequentially to obtain a disparity map. In the incoming sections each step will be further detailed.

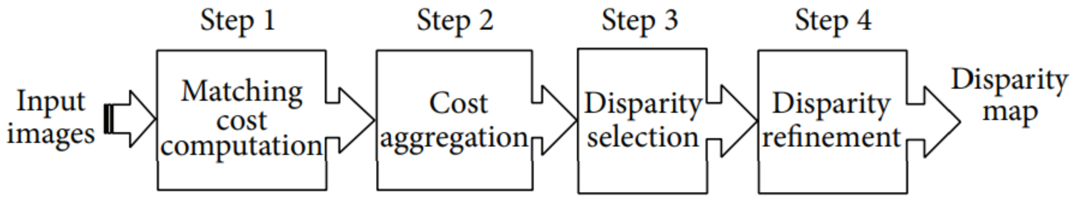


Figure 3.1: Taxonomy of Stereo Algorithms [6].

3.2. Matching cost computation

In this stage it is determined whether the values of two pixels correspond to the same point in a scene. It is done by computing a dissimilarity cost that is based on the intensity of the pixels, between one pixel (or a set of pixels) in the left and in the right image.

In order to compute the dissimilarities, first it is decided which of the two images will be the reference image, i.e the image which will be taken as reference for looking for the match pixels on the other image (match image). Considering the input image pair to be rectified and calibrated, x to be the image width (columns), y the image height (rows), d the disparity chosen, and $p = (x, y)$ the reference pixel, then there are two possible cases:

1. The reference image is the left image: In this case, the match pixel of p will be:

$$pd = (x - d, y) \tag{3.1}$$

   It is important to note that only the pixels starting at column d on the reference image will have a possible match pixel, since in the other cases $x - d$ becomes negative. An illustrative diagram of the reference and match pixel can be found on Figure 3.2. Notice that this is the case when the disparity is equal to 1 pixel.

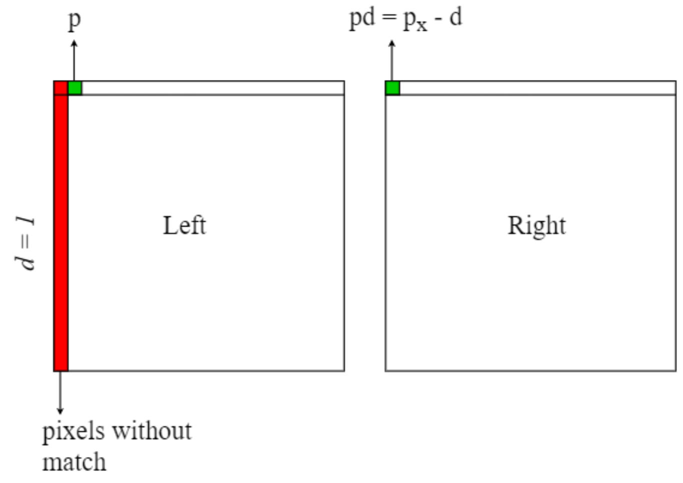


Figure 3.2: Illustration of reference and match pixel when left image is reference and disparity is 1 pixel.

2. The reference image is the right image: In this case, the match pixel of p will be :

$$pd = (x + d, y) \tag{3.2}$$

   Here, pixels on the reference image can be at most on column $x - d$, since if they are at a bigger column $x + d$ surpasses the image width. An illustrative diagram of the reference and match pixel can be found on Figure 3.3. Notice that this is the case when the disparity is equal to 1 pixel.

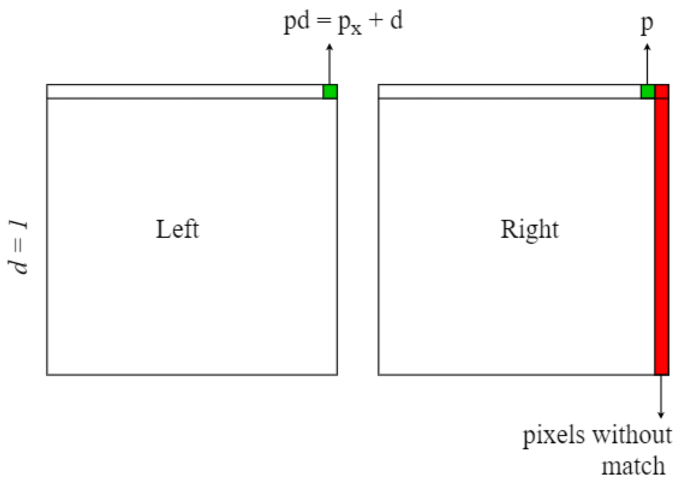


Figure 3.3: Illustration of reference and match pixel when right image is reference and disparity is 1 pixel.

Note that since the input image pair is rectified, the reference and match pixels will always lie on the same row (y). Once the reference image has been chosen, the dissimilarity cost will be computed for each pixel pair (p and pd) and disparity under consideration, ending up with a matching cost matrix: $C(p, d)$.

The disparities under consideration are defined by the disparity range, which is the range of disparity values that will be considered in the search for the match pixel. It corresponds to the difference between the maximum and minimum disparity value defined. A disparity of zero is not considered valid, therefore the minimum disparity must be at least 1 pixel.

Considering equations 3.1 and 3.2, when calculating the matching cost matrix, some pixels do not have a match pixel depending on the disparity value. In the case of the left image as reference, this happens at the leftmost columns of the reference, whereas for the right image as reference, it happens at the rightmost columns. The dissimilarity value assigned to the matching cost matrix in those pixels is infinite.

There are different approaches for computing the matching cost, some of them will be explained in the incoming sections, as they are of interest in this memoir. It is important to remember that from now on, a pair of images is assumed to be rectified, so that the epipolar lines match the rows of the images.

3.2.1. Absolute differences (AD)

This algorithm uses the absolute difference of the intensities between reference and match image, as a pixel dissimilarity measure [7]. It is defined by the following equation:.

$$AD(p, d) = |I_{ref}(p) - I_{match}(pd)| \tag{3.3}$$

The $AD(p, d)$ equation must be calculated for every pixel in the reference image and every disparity in the disparity range defined, thus, the matching cost matrix is obtained. The pixels that do not have a match, depending on the disparity, are assigned an infinite value, as mentioned.

3.2.2. Sum of absolute differences (SAD)

The SAD algorithm [7] sums the absolute difference between the intensities of each pixel in the reference and match support window (w), as stated in equation 3.4. The window size must be odd and is set by the programmer.

$$SAD(p,d) = \sum_{p\,\in\,w} |I_{ref}(p) - I_{match}(pd)| \tag{3.4}$$

The $SAD(p,d)$ equation must be calculated for every pixel in the reference image and every disparity in the disparity range defined, thus, the matching cost matrix is obtained. The pixels that do not have a match, depending on the disparity, are assigned an infinite value, as mentioned.

3.2.3. BirchField Tomasi (BT)

In order to establish the dissimilarity between two pixels, the Birchfield Tomasi approach [13] proposes to use the linearly interpolated intensity functions surrounding the two pixels in question. Defining a scanline as the epipolar lines to be compared, and considering $p = (x_l, y)$ to be the reference pixel and $pd = (x_r, y)$ the match pixel, the following quantity can be defined:

$$\bar{d}(x_l, x_r, I_l, I_r) = \min_{x_r - \frac{1}{2} \leq x \leq x_r + \frac{1}{2}} \left| I_l(x_l) - \hat{I}_r(x) \right| \tag{3.5}$$

where I_l corresponds to the intensity function of scanline (row) y of the reference image, I_r to the intensity function of scanline y of the match image, and $\hat{I}_r$ as the linearly interpolated intensity function between the sample points of the match scanline. Considering this, a symmetric quantity can be obtained:

$$\bar{d}(x_r, x_l, I_r, I_l) = \min_{x_l - \frac{1}{2} \leq x \leq x_l + \frac{1}{2}} \left| \hat{I}_l(x) - I_r(x_r) \right| \tag{3.6}$$

Now, the BT dissimilarity between a pair of pixels is defined symmetrically as the minimum of the two quantities:

$$BT(x_l, x_r) = \min\{\bar{d}(x_l, x_r, I_l, I_r), \bar{d}(x_r, x_l, I_r, I_l)\} \tag{3.7}$$

To compute $\bar{d}(x_l, x_r, I_l, I_r)$, it is necessary to define the two following terms, I_r^- and I_r^+, which are the linearly interpolated intensities (see Figure 3.4):

$$I_r^- \equiv \hat{I}_r(x_r - \frac{1}{2}) = \frac{I_r(x_r) + I_r(x_r - 1)}{2} \tag{3.8}$$

$$I_r^+ \equiv \hat{I}_r(x_r + \frac{1}{2}) = \frac{I_r(x_r) + I_r(x_r + 1)}{2} \tag{3.9}$$

Lastly, let $I_{min} = min\{I_r^-, I_r^+, I_r(x_r)\}$ and $I_{max} = max\{I_r^-, I_r^+, I_r(x_r)\}$ then:

$$\bar{d}(x_l, x_r, I_l, I_r) = \max\{0, I_l(x_l) - I_{max}, I_{min} - I_l(x_l)\} \tag{3.10}$$

This computation, along with its symmetric counterpart $\bar{d}(x_r, x_l, I_r, I_l)$ allows to calculate the final Birchfield Tomasi dissimilarity between a pixel pair, which was defined in equation 3.7. This equation must also be computed for every pixel in the reference image and every disparity in the disparity range. Pixels that do not have a match are assigned an infinite value, as well as in the previous dissimilarity measures explained.

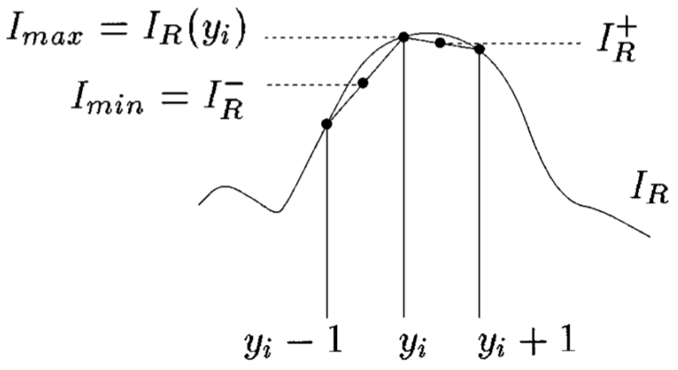


Figure 3.4: Computation of I_r^- and I_r^+ [13].

3.2.4. Pixel dissimilarity with Deep Learning

Deep learning in the stereo matching problems has gained attraction and interest of the computer vision community, due to their remarkable performance, far exceeding traditional approaches [14], like the ones explained before. In fact, currently the top 10 performing algorithms on the KITTI 2012 dataset [5] are deep learning based [15]. The architecture chosen to implement in this memoir is a patch comparison neural network, that will be explained below. Nevertheless, it is important to note that this is not the only approach at solving the stereo matching problem, as other types of CNN architectures can be used.

When implementing a pixel dissimilarity with deep learning, a Convolutional Neural Network (CNN) architecture is to assign a cost to matching pixels. In the case of patch comparison, the CNN compares two image patches and returns a probability distribution indicating the similarities and dissimilarities between them. An illustrative diagram is shown in Figure 3.5.

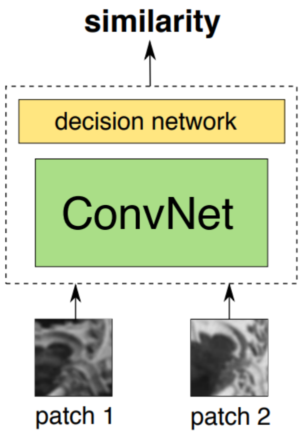


Figure 3.5: Illustration of a CNN patch comparison architecture [16].

In order to assign a cost between matching pixels, the input patches must be centered in pixel $p = (x, y)$ on the reference image and pd on the match image (coordinates of pd depend on which of the two images is taken to be the reference).

A CNN architecture that is of interest of this memoir is the one presented in Žbontar et al [17], and is shown in Figure 3.6. It was chosen because of its simplicity and good performance. The network consists of eight layers, L1 through L8. The first layer is convolutional, while all others are fully- connected. The inputs to the network are two 9×9 gray image patches. The final layer (L8), projects the output to two real numbers that are fed through a softmax function, producing a distribution over the two classes: good match (patch similarity) and bad match (patch dissimilarity).

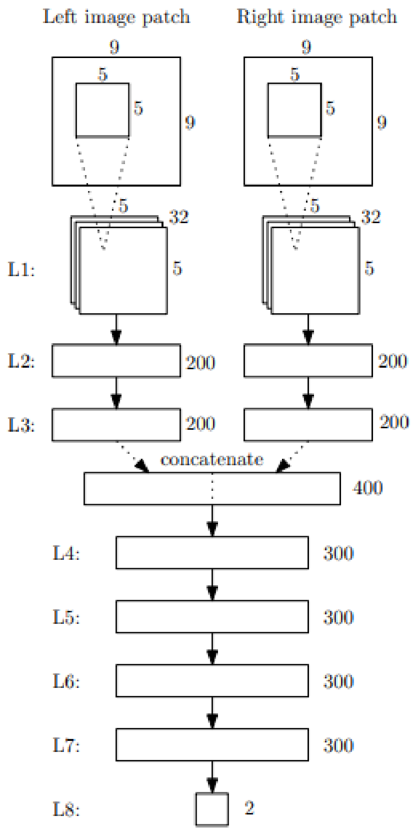


Figure 3.6: CNN architecture [17].

For training, the authors extract patch pairs (negative and positive) from the KITTI stereo dataset [5], which consist of stereo image pairs with its corresponding ground truth disparity. Details can be found in a chapter 5.

The matching cost $C(x, y, d)$ is computed directly from the output of the network:

$$C(x, y, d) = C(p, d) = f_{neg}\{P_L(p), P_R(pd)\} \tag{3.11}$$

Where $f_{neg}\{P_L(p), P_R(pd)\}$ corresponds to the output of the network for the negative class (i.e patch dissimilarity) with input patches $P_L(p)$ centered at $p = (x, y)$ on the reference image and $P_R(pd)$ centered at pd on the match image.

To compute the match cost, a forward pass is needed for each pixel and disparity under consideration, which would be a total of $width \times height \times disp\ range$ forward passes. Of course, this is not possible in terms of runtime. To solve this problem, Žbontar et al proposes the following:

1. Layers L1, L2, and L3 can be computed only once per pixel location, and don't need to be recomputed for every disparity d.
2. The output of L3 can be computed in a single forward pass for all locations.
3. Replacing layers L4 to L8 with convolutional filters of sixe 1×1.

Implementing the steps just detailed will make the runtime manageable, but still not suited for a real time application.

3.3. Cost aggregation

This stage minimizes matching uncertainties from stage 1, because generally it is not sufficient for precise matching and robust disparity estimations.

3.3.1. Semi Global Matching (SGM)

Following Hirschmuller's Semi Global Matching [18], additional constraints are added to the matching cost matrix in the form of an energy function that depends on the disparity image D:

$$\begin{aligned} E(D) = \sum_{p} \Big(& C(p, D(p)) \\ & + \sum_{q \in \mathcal{N}_p} P_1 \times 1\{|D(p) - D(q)| = 1\} \\ & + \sum_{q \in \mathcal{N}_p} P_2 \times 1\{|D(p) - D(q)| > 1\} \Big) \end{aligned} \tag{3.12}$$

where $1\{\cdot\}$ denotes the indicator function. The first term is the sum of all pixel matching costs for the disparities of D. The second term adds a constant penalty P_1 when the disparity of neighboring pixels differ by one, and the third term adds a constant penalty P_2 (larger than P_1) when the neighboring disparities differ by more than one. Using a lower penalty for P_1 permits an adaptation for inclined or curved surfaces in the image. P_2 allows to preserve discontinuities, because the penalty is constant independent of the size of the disparity change.

The stereo matching problem can now be solved by finding the disparity image D that minimizes the energy $E(D)$. However, this type of global minimization (i.e in 2D) is a NP-complete problem, that cannot be solved in a reasonable amount of time. The proposed solution of [18] is to aggregate matching costs in 1D from all directions equally using a dynamic programming approach. The aggregated (smoothed) cost $S(p, d)$ for a pixel p and disparity d is calculated by summing the costs of all 1D minimum cost paths that end in pixel p at disparity d, as shown in Figure 3.7

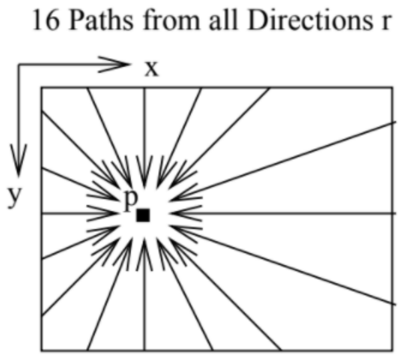


Figure 3.7: Aggregation of costs [18].

The cost $L_r(p,d)$ along a path traversed in the direction r of pixel p at disparity d is defined recursively:

$$\begin{aligned} L_r(p,d) = C(p,d) + min\Big\{ & L_r(p-r,d), \\ & L_r(p-r,d-1) + P_1, \\ & L_r(p-r,d+1) + P_1, \\ & \min_i L_r(p-r,i) + P_2 \Big\} - \min_k L_r(p-r,k) \end{aligned} \tag{3.13}$$

where $C(p,d)$ corresponds to the cost associated to pixel p and disparity d, obtained in the matching cost computation stage. Parameters i and k of the last two terms belong in the disparity range defined in the previous stage. The last term is necessary to avoid $L_r(p,d)$ of becoming to big. It is also worth noting that parameters P_1 and P_2 are set by the programmer, however it needs to be ensured that $P_1 \leq P_2$.

Calculation of equation 3.13 starts with the image borders, initializing all L_r borders with the values of the cost matrix. Once the borders are set, it is possible to compute all the other values.

In Hirschmuller's paper is recommended to compute the cost $L_r(p,d)$ with 16 directions, as this ensures that the majority of the image is covered. Finally, the aggregated cost matrix is obtained by summing the cost paths of all directions:

$$C_{SGM}(p,d) = \sum_r L_r(p,d) \tag{3.14}$$

3.3.2. Cross Based Cost Aggregation (CBCA)

The basic idea of cross based cost agreggation [19] is to aggregate the costs from a neighborhood of the pixel, but the neighborhood is adaptively selected so that pixels within it have similar disparities.

First, an upright cross is constructed for each pixel. The left arm p_l at position p extends left as long as the following two conditions hold:

1. The absolute difference in intensities of the image at positions p and p_l is smaller than τ (see equation below):
$$|I(p) - I(p_l)| < \tau \tag{3.15}$$

2. The horizontal distance (or vertical distance, in case of top and bottom arms) between p and p_l is less than η (see equation below):
$$\|p - p_l\| < \eta \tag{3.16}$$

The right, bottom, and top arms are constructed analogously. Once the four arms are known, the support region $U(p)$ can be defined as the union of the horizontal arms of all positions q laying on $p's$ vertical arm (see equation 3.17):

$$U(p) = \bigcup_{q \in V(p)} H(q) \tag{3.17}$$

Where $V(p)$ is $p's$ vertical arm and $H(q)$ are $q's$ horizontal arms. An illustrative example of a support region can be seen in the Figure below.

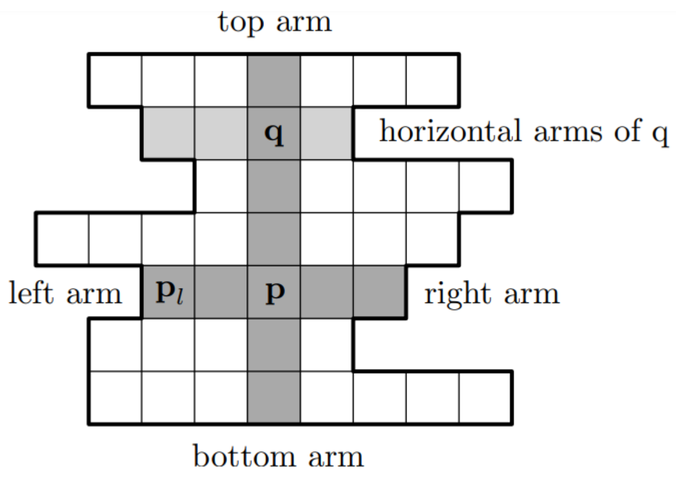


Figure 3.8: Example of support region for pixel p [17].

The authors suggest that aggregation should consider the support regions of both images in a stereo pair. Let $U_d(p)$ be the combined support region (of left and right images), then the matching cost is averaged over $U_d(p)$, therefore obtaining the aggregated cost:

$$C_{CBCA}(p, d) = \frac{1}{|U_d(p)|} \sum_{q \in U_d(p)} C(q, d) \tag{3.18}$$

where $C(q, d)$ corresponds to the cost associated to pixel q and disparity d, obtained in the matching cost computation stage. This process of aggregating the costs in the support region $U_d(p)$ can be done iteratively in some cases, to increase the accuracy of the stereo method [17].

3.4. Disparity selection

This is the stage where the best disparity (or the smallest cost) for a pixel is chosen from the aggregated cost matrix. For this, a Winner Takes All (WTA) approach is used [18]. In it, the disparity d associated to pixel p in the final disparity image is defined by:

$$d_p = \underset{d \in D}{\arg\min}\ C_{CA}(p, d) \tag{3.19}$$

where D corresponds to the disparity range and $C_{CA}(p, d)$ to the aggregated cost.

3.5. Disparity refinement

After choosing the best disparity with the WTA function, the disparity image can still contain certain kinds of errors, that can be recovered with post processing. Some of this post processing algorithms will be explained in the incoming sections.

3.5.1. Left-Right consistency check

Considering D_L to be the disparity using the left image as the reference image, and D_R to be the disparity using the right image as the reference image, mismatches can be detected using a consistency check between left and right disparities [18]. This way, the final disparity for each pixel can be defined by the following equation:

$$D(p) = \begin{cases} D_L(p), & \text{if } |D_L(p) - D_R(q)| \leq 1 \\ D_{\text{mismatch}}, & \text{otherwise} \end{cases} \tag{3.20}$$

where D_{mismatch} corresponds to an invalid disparity value, i.e. 0, and $q = p - D_L(p)$.

3.5.2. Sub-pixel enhancement

To increase the resolution of a stereo algorithm, a quadratic curve is fitted through the neighboring costs to obtain a new disparity image [17]:

$$D_{SE}(p) = d - \frac{C_+ - C_-}{2(C_+ - 2C + C_-)} \tag{3.21}$$

where $d = D(p)$ is the disparity obtained after the Left Right consistency check, and $C_+ = C_{CA}(p+1, d)$, $C_- = C_{CA}(p-1, d)$, $C = C_{CA}(p, d)$ all correspond to the aggregated cost.

3.5.3. Median filter

The median filter is used to remove noise from an image and tha basic idea behind it is to replace every pixel of the image with the median value of its neighbors. The neighborhood of a pixel is defined by a kernel size, which must be odd. This filter is useful for removing small and isolated mismatches in the final disparity while preserving the edges.

3.6. Pre filtering

For prefiltering the input images, the Sobel X filter can be applied. It computes an approximation of the gradient of the intensity function in the x or horizontal direction. It is usually used for edge detection. In the context of stereo algorithms, it is useful when applied to the input images, as it makes them less prone to illumination and exposure changes [20]. This way the algorithm is able to find more disparities and more accurately. Below is a an example of the Sobel X filter.

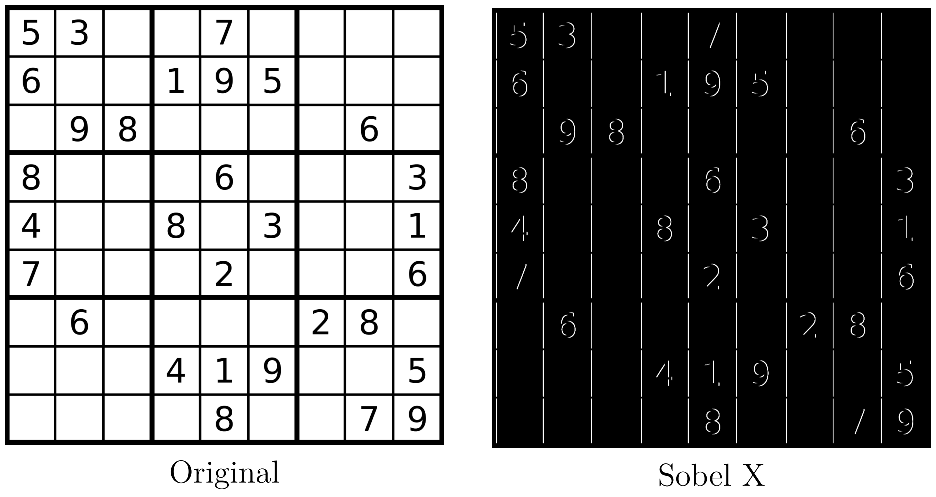


Figure 3.9: Result of applying the Sobel X filter.

Chapter 4

Implementation Methodology

The stereo algorithm implemented follows the taxonomy of Scharstein and Szeliski [6] explained in chapter 3. Two configurations have been implemented, and they differ only in the matching cost computation stage. The first configuration will be referred as the Birchfield Tomasi (from now on BT) configuration, and the second as the CNN configuration. The building blocks of the two algorithms will be explained in the incoming sections.

4.1. Matching cost computation

In this memoir, the matching cost matrix will be arranged in a three dimension array of coordinates (x, y, d), with x being the width of the input image pair and y the height. An illustration of the cost matrix can be seen on Figure 4.1. In it, it is possible to see that each depth holds the cost values of a different disparity.

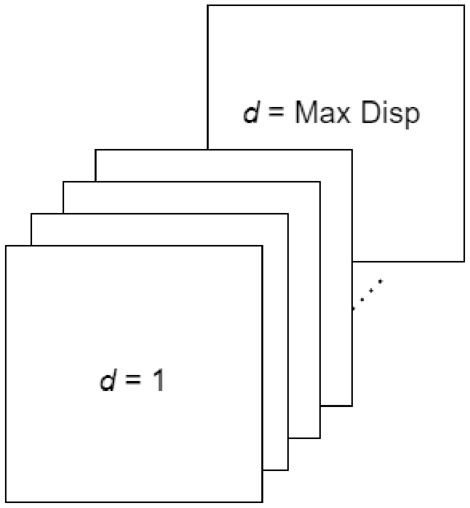


Figure 4.1: Matching cost matrix.

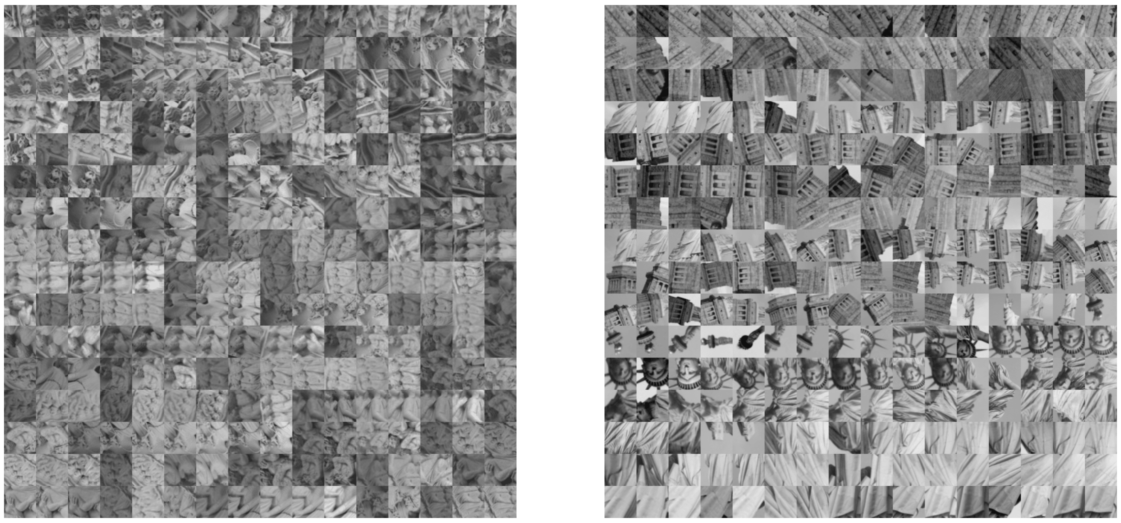
4.1.1. BT configuration

This configuration will have as a matching function the Birchfield Tomasi pixel dissimilarity. It was chosen considering that it gives better results than the absolute differences dissimilarity, according to [13].

4.1.2. CNN configuration

This configuration will have as a matching cost function the CNN architecture explained in the previous chapter (see Figure 3.6), presented in the paper of Žbontar et al [17]. However, only one of the three suggested optimizations were implemented for computing the match cost (also explained in chapter 3). This decision will play a significant role in the execution time, as it will be shown later.

Also, a change has been made from the dataset used in [17], due to the fact that training with the Multi-view Stereo Correspondence Dataset [21] was more straightforward than with KITTI. This is because the Multi-view dataset comes with pre made patches, unlike the KITTI. The selected dataset consists of corresponding patches sampled from 3D reconstructions of the Statue of Liberty (New York), Notre Dame (Paris) and Half Dome (Yosemite). The Notredame patches (Figure 4.2 left) were used for training and evaluation of the CNN, while the Liberty patches (Figure 4.2 right) for testing the CNN.



Notredame dataset sample Liberty dataset sample

Figure 4.2: Examples of the Multi-view Stereo Correspondence Dataset[21].

The CNN was trained during 20 epochs, using a total of 400.000 patch pairs of the Notredame dataset. 70% of them were used for training the CNN, and 30% for evaluating performance in each epoch. Batch size was 128 and learning rate 1^{-4}. The network training was done with tensorflow and a RTX2080 nvidia GPU.

Once the training and evaluation is done, the Liberty dataset was used to test how well the CNN is able to predict good and bad matches between patches. The neural network was tested with 100.000 patch pairs (with class balance). The performance metrics chosen are the accuracy and precision, defined in equations 4.1 and 4.2, respectively.

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{4.1}$$

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{4.2}$$

Where:

- TP(True Positive) is when the CNN assigns correctly a positive label to a pair of patches.
- TN(True Negative) is when the CNN assigns correctly a negative label to a pair of patches.
- FP(False Positive) is when the CNN assigns incorrectly a positive label to a pair of patches.
- FP(False Negative) is when the CNN assigns incorrectly a negative label to a pair of patches.

So, in summary, the accuracy is the percentage of correct predictions in the test set, and the precision is the percentage of correct positive predictions in the test set. Considering this, the obtained results of the CNN in the test set are shown below.

Accuracy	87,24%
Precision	90,00%

Table 4.1: CNN testing results.

4.2. Cost aggregation

For both configurations, Semi Global Matching will be implemented for this stage. As mentioned earlier, it is recommended to use 16 directions of optimization , but in this memoir only 8 will be used, for computational complexity reasons. They are shown in the below Figure.

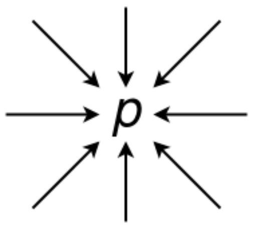


Figure 4.3: Directions used for aggregating the matching cost.

When it comes to the parameter setting (P_1 and P_2), they will be set depending on the input images and match cost function used.

4.3. Disparity selection

Once the cost agreggation step is finished, a WTA aproach will be used to choose the best disparity. However, some restrictions will be imposed when finding the minimum cost, to further refine the disparity image:

1. If the minimum found for a pixel is not unique, then the disparity value assigned to the pixel will be zero.

2. If the minimum found for a pixel is infinite, then the disparity value assigned to the pixel will be zero (an infinite value on the matching cost matrix can happen because the reference pixel does not have a matching pixel with a certain disparity).

3. The minimum found must win to the second best value by a certain percentage threshold (defined by the programmer). If this threshold is not surpassed, then the disparity value assigned to the pixel will be zero. The value chosen for the threshold was 15%

4.4. Disparity refinement

After the minimum disparity is selected, a Left-Right consistency check is performed. This implies that the preceding steps must be performed twice: once considering the left image as reference and a second time considering the right image as reference. When computing the disparity with the left image as reference it will be addressed as the left to right disparity (D_{L-R}). The other case will be addressed as the right to left disparity (D_{R-L}). Before feeding both disparities to the Left Right check step, they will be filtered with the median filter.

After the Left Right check step, an initial disparity map is obtained (D^1, see Figure 4.4), this will be filtered with sub pixel enhancement and the median filter (D^2, see Figure 4.4). Lastly, every pixel of the disparity is multiplied by a scale factor, with the purpose of a better visualization. It must be so that the maximum disparity allowed multiplied by the scale factor is at most 255, considering that the disparity images will be stored in a 2D array of 8 bits.

4.5. Summary

In the Figure below, a summary of the blocks of both algorithms is presented. The second block (matching cost computation) can be either the Birchfield Tomasi dissimilarity (BT), or the patch comparison convolutional neural network (CNN). All other blocks are the same for both configurations, they include the cost aggregation step (SGM), disparity selection (WTA) and post processing (median filter, L/R check and SE enhancement).

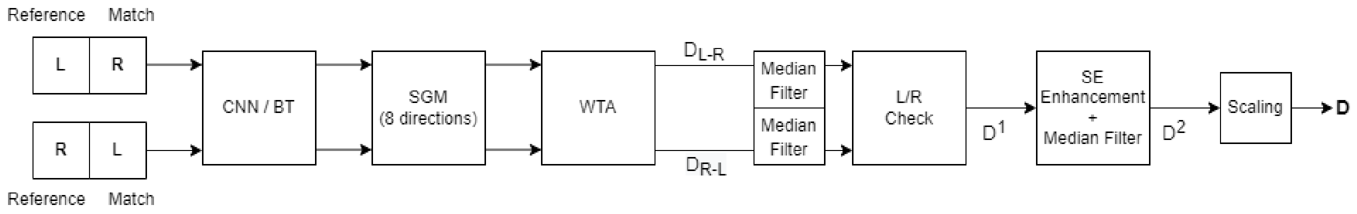


Figure 4.4: Processing steps of the implemented stereo algorithm. SGM stands for semi global matching, WTA for winner takes all, L/R check for left right check and SE enhancement for subpixel enhancement.

The algorithm will be implemented with an object oriented architecture, meaning it will be a main disparity estimator class with their own functions or methods, corresponding to the building blocks detailed in the previous sections. The disparity class is to be placed inside a bigger library owned by the company, named *CalcAgent*, which is a big system that communicates its sub-libraries together to end up with a flux of different algorithms.

Chapter 5

Evaluation Methodology

The two configurations mentioned in the previous section will be evaluated with two datasets: the Middlebury (indoor) and the KITTI (outdoor) datasets. For the Truck images, a visual or qualitative evaluation will be done.

5.1. Middlebury dataset

The Middlebury Datasets [4] are a set of 6 different datasets. They contain stereo image pairs that are rectified and without radial distortion, with their corresponding ground truth disparity.

The dataset chosen was the 2006 datasets. It consists of 21 stereo image pairs with pixel-accurate correspondence information, obtained using a structured light technique [22]. Each image set consist of 7 views (0..6) taken under three different illuminations (1..3) and with three different exposures (0..2). Ground truth disparity maps are only provided for views 1 and 5. They also provide 3 different resolutions for each stereo pair: full size, half size and third size. For this memoir, the third size resolution has been chosen (width: 413..465 pixels, height: 370 pixels) and the algorithm will only be tested with a single illumination and exposure.

The ground truth disparities for the third size resolution are encoded using a scale factor of 3, meaning that the real maximum disparity is equal to 85 ($\frac{255}{3}$). A disparity of zero (black pixels) is considered an unknown disparity. Below is an example of the 2006 Middlebury dataset.

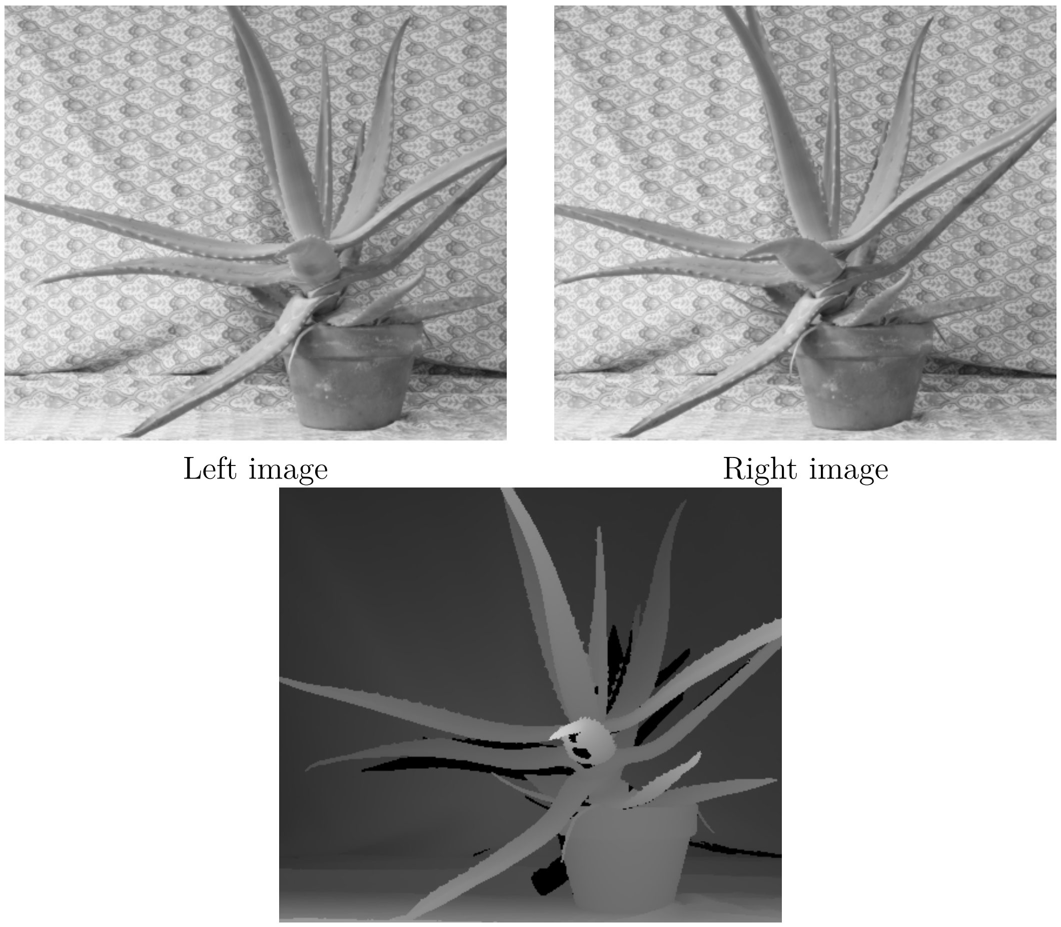


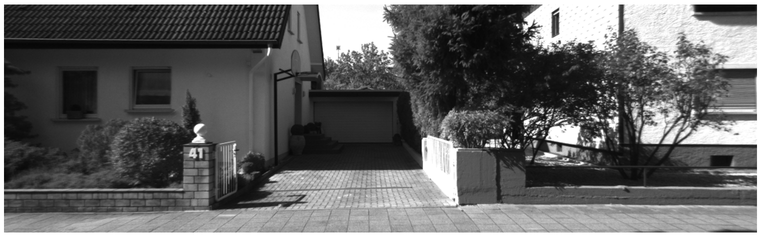
Figure 5.1: Sample of the Middlebury 2006 dataset [4].

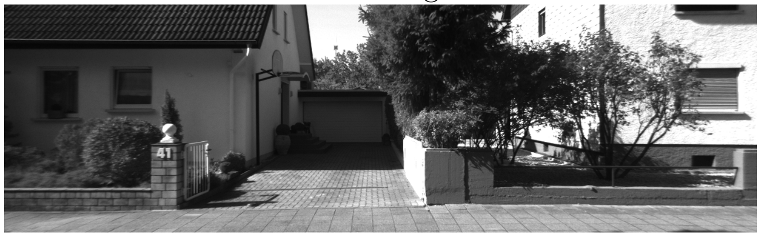
5.2. KITTI dataset

The KITTI stereo dataset [5] is a collection of gray image pairs taken from two cameras mounted on the roof of a car, 54 centimeters apart. The images are recorded while driving around the city of Karlsruhe, in sunny and cloudy weather, at daytime. The dataset comprises 194 training and 195 test image pairs at a resolution of 1238 ... 1242 × 374 ... 376 pixels. Each image pair is rectified. A rotating laser scanner, mounted behind the left camera, provides ground truth depth. The true disparities for the test set are withheld by the authors, allowing submissions only once every three days. The goal of the KITTI stereo dataset is to predict the disparity for each pixel on the left image. In this memoir, since the CNN configuration was trained with another dataset, the training images of KITTI will be used for evaluating the algorithms. Only 20 images will be used, because predicting all 194 images would be extremely time consuming.

For each of the 20 images, a different disparity range will be used (to save computation time), depending on the ground truth disparity. So, the disparity range will be set according to the smallest and biggest disparity appearing in the ground truth. The different disparity ranges of the images will go from 51 to 100 pixels.

In contrast with the Middlebury dataset, in the KITTI there is no scale factor, so disparities are represented “as is”, i.e., intensity 60 means the disparity is 60. Below is an example of a stereo pair and its corresponding ground truth disparity. For visualization purposes the ground truth has been scaled, so that disparities appear more bright. A disparity of zero (black pixels) means the disparity is unknown in that position.


Left image


Right image

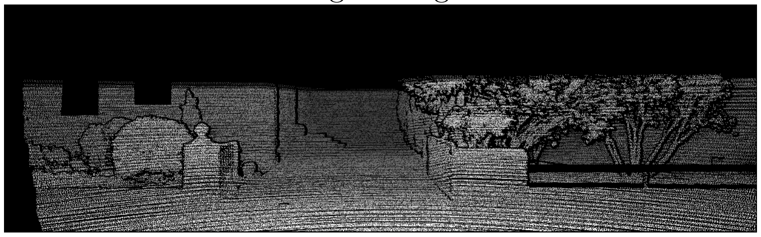

Ground truth disparity

Figure 5.2: Sample of the KITTI 2012 dataset [5].

5.3. Truckload images

The truck images that will be used are size 1200x1000 pixels. They are captured by the two stereo cameras detailed in chapter 1, and they are capturing images the entire day. When its nighttime, there are artificial lights placed for each camera. Examples of the images can be seen below.

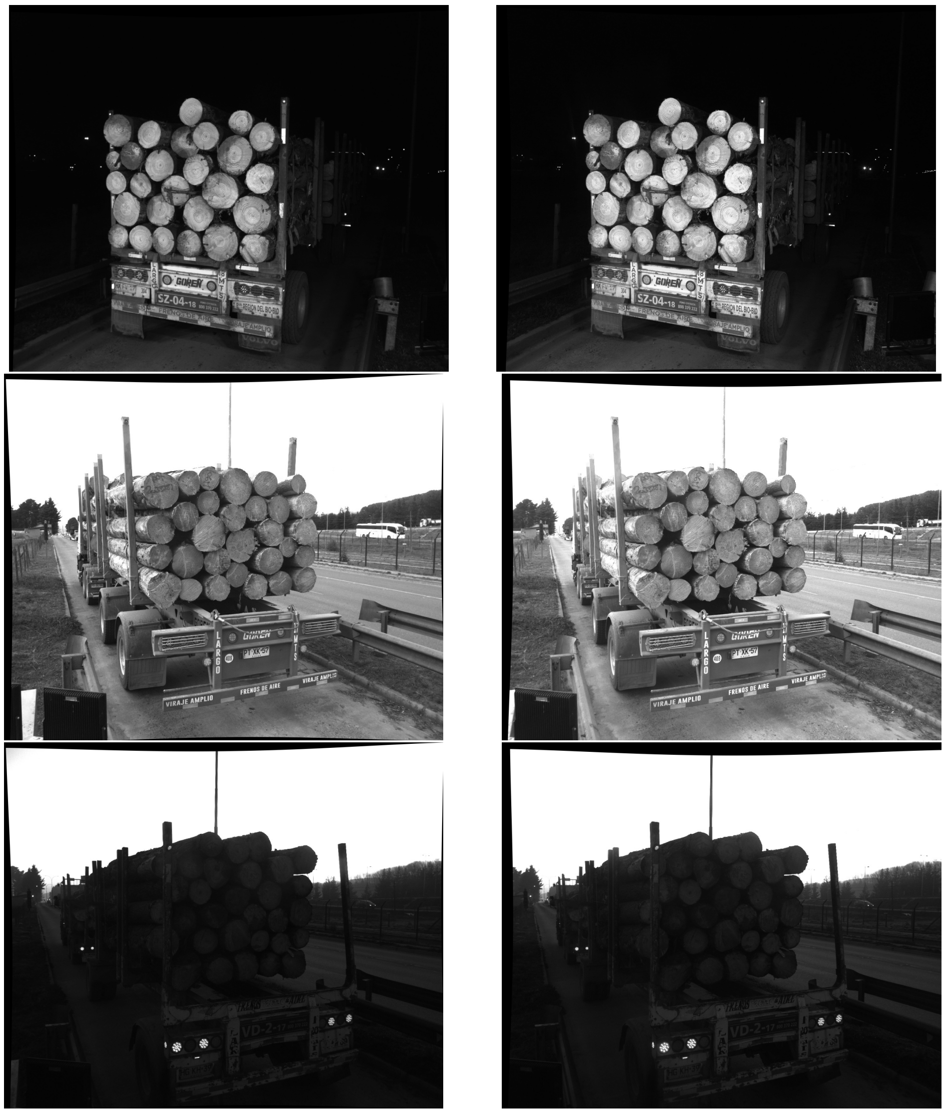


Figure 5.3: Left and right Truck images.

5.4. Error computation

In the sense of computing a measure of accuracy of the algorithm, three types of errors will be defined. These will be calculated for each disparity image predicted by the algorithm, using the input images in the two datasets mentioned earlier (KITTI and Middlebury).

Consider P_{total} to be the total amount of pixels whose disparity is greater than zero in the ground truth disparity image (the pixels whose disparity is zero (i.e black pixels), will not be added to P_{total}, because a value of zero in the ground truth represents an unknown disparity, as mentioned). Now, we can define the following quantities:

1. Occlusion error: It is calculated by summing all the occluded pixels in the predicted disparity image and dividing it by the total amount of pixels defined earlier (P_{total}), as follows:

$$\text{Occ Error} = \frac{\sum p_{occluded}}{P_{total}} \tag{5.1}$$

where $p_{occluded}$ represents an occluded pixel. A pixel p is defined as "occluded" when its predicted disparity is zero and its ground truth disparity is non zero, therefore:

$$p_{occluded} \iff d_{predicted}(p) = 0 \text{ and } d_{GT}(p) > 0 \tag{5.2}$$

2. Mismatch error: It is calculated by summing all the mismatched pixels in the predicted disparity image and dividing it by the total amount of pixels (P_{total}), as follows:

$$\text{Mism Error} = \frac{\sum p_{mismatched}}{P_{total}} \tag{5.3}$$

where $p_{mismatched}$ represents a mismatched pixel. A pixel p is defined as "mismatched" when its predicted disparity and the ground truth disparity differ by more than a specific threshold, set by the programmer. Therefore:

$$p_{mismatched} \iff |d_{predicted}(p) - d_{GT}(p)| > \tau \text{ and } d_{GT}(p) > 0, \, d_{predicted}(p) > 0 \tag{5.4}$$

In the literature, the threshold τ varies from 0.5 pixels to 3 pixels, but is usually 1 pixel[18] [17].

3. Overall error: It corresponds to the sum of the two errors previously defined in equations 5.1 and 5.3:

$$\text{Overall Error} = \text{Mism} + \text{Occ} \tag{5.5}$$

The threshold τ for correct matches will be set according to the dataset used. In this memoir, a threshold of 1 pixel will be used for the Middlbury dataset and a threshold of 3 pixels for the KITTI. The threshold for KITTI is looser because it is considered to be a harder dataset to use for evaluation, since it is in an outdoor environment.

When computing the error measures, in the case of Middlebury, since the ground truth disparity is scaled by a factor of 3, the predicted disparity map as well as the error threshold will also be scaled by 3 (error threshold of Middlebury ends up being 3 pixels after scaling). For the KITTI, the ground truth disparity is not scaled, so the error measures will be computed before scaling the final predicted disparity (therefore the error threshold of KITTI remains the same, meaning is 3 pixels).

Considering that the output disparity map of both algorithms is when the left image is the reference (because of the Left Right consistency), some of the leftmost columns of the left image do not have a match column on the right image, depending on the disparity value. This is why the disparity value of the first n columns ($n \in$ disparity range) will be set to zero. For example, in the case of a Middlebury image, that have a disparity range of 84 (1 to 85), the first 84 columns of the disparity will have a value of zero. It is worth mentioning though, that a disparity value has actually been calculated for those columns, but because no interpolation technique was applied to them, it has been decided not to consider them when computing the error, and therefore it is better to leave the value to zero. Finally, leaving the first n columns with a value of zero in the final disparity image will not represent a problem in the truck images, since their size is 1200 x 1000 pixels.

Chapter 6

Results and Analysis

In the following sections, the output disparity map for the two algorithm configurations explained before (BT and CNN) will be shown, divided by each dataset: Middlebury, KITTI and the truckload images.

The disparity map, originally a grayscale 8 bit image, is transformed to a color image, to enhance the visualization. This is done by using the Open-CV [2] color map function. The colormap chosen is the “Jet” colormap (see Figure 6.1). The darkest shade of blue represents an invalid disparity (i.e zero). Warmer colors represent larger values of disparity, therefore a smaller value of depth, meaning that the object is closer to the stereo camera than a blueish color. As explained in chapter 5, the first n columns of the disparity image ($n \in$ disparity range) are set to zero, and hence they are displayed as the darkest shade of blue in the Jet colormap.

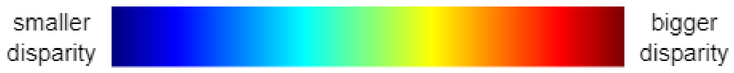


Figure 6.1: Jet Colormap [2].

To show the performance of the configurations, error tables will be shown for each dataset, indicating the three types of error described in the previous section (mismatch, occlusion and overall). These were calculated for each image in the respective dataset, hence, only the mean and standard deviation (SD) of each error will be detailed. The full tables indicating the errors for each image on both datasets can be found on annexes A.1 and A.2.

6.1. Middlebury dataset

Considering a disparity range of 84 (1 to 85), a scale factor of 3 and an error threshold for correct matches of 1 pixel, in table 6.1 there is a detail of the computed errors for both configurations. Looking at the overall error, it can be said that both configurations have similar performance on this dataset, with both the mean and standard deviation being very similar. However, the difference lies on how the mismatch and occlusion errors are distributed, with the CNN having less occlusions but more mismatches. This gets reflected in that the output image has more predicted disparities (i.e less holes), but those predictions are not correct. An example of this can be seen in Figure 6.2, where the output of the CNN configuration looks more filled compared to the BT, however both images have a similar overall error: 20.47% (BT) and 19.02% (CNN).

	BT configuration		CNN configuration	
	Mean	SD	Mean	SD
Mismatch error (%)	14.27	11.11	18.10	12.66
Oclusion error (%)	17.98	14.97	14.90	10.00
Overall error (%)	32.25	22.52	33.00	21.63

Table 6.1: Results for Middlebury dataset. Error threshold is 1 pixel.

In Figure 6.3, there is a display of the output disparity maps of both configurations, with 5 image pairs from the dataset. The first row is the best evaluated image on the entire dataset for both algorithms, the same goes for the last row, but its the worst evaluated image. Both algorithms perform worst when there is a textureless area. Its important to notice there is no difference in illumination or exposure between left and right input images.

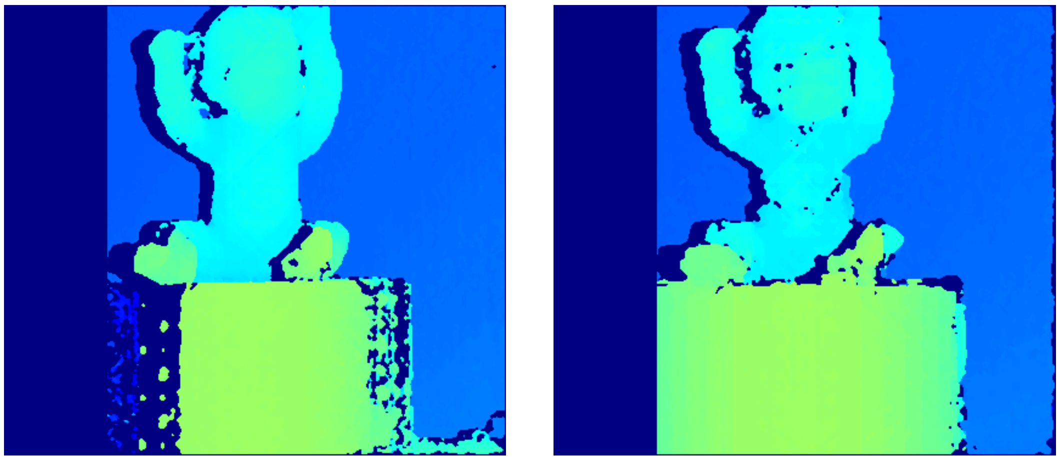


Figure 6.2: Comparison of occlusions and mismatches between BT and CNN configuration. Image name is “Baby1”.

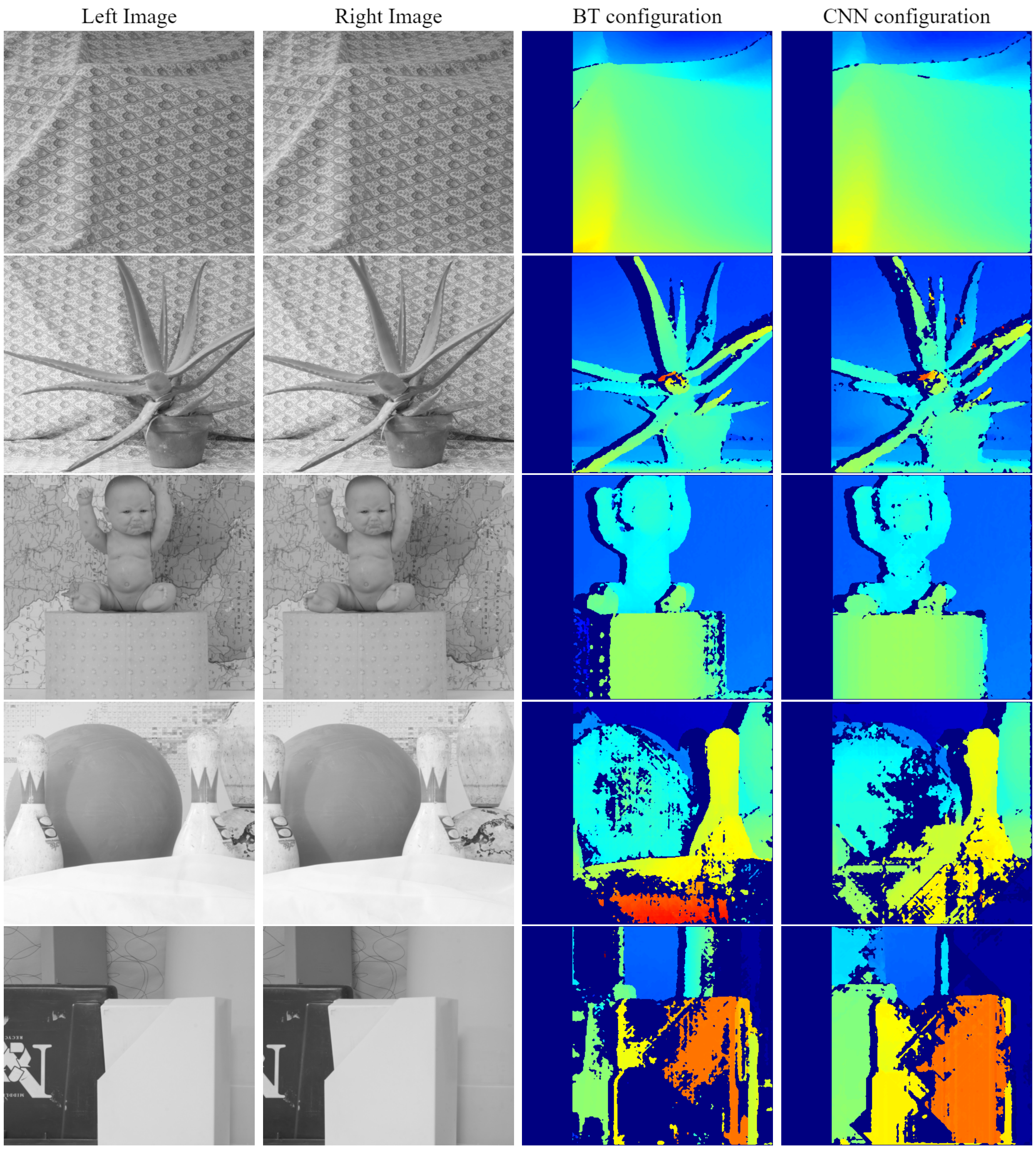


Figure 6.3: Results for Middlebury dataset with BT and CNN configuration. Left columns: input images, center right column: BT configuration, righmost column: CNN configuration. Image names are, from top to bottom: “Cloth1”, “Aloe”, “Baby1”, “Bowling1” and “Plastic”.

When it comes to execution time (see Table 6.2), both configurations only differ by the matching cost computation stage, since all other stages are the same. Considering this, the BT configuration surpasses the CNN by far, however, it is important to consider that the CNN predictions are not maximally optimized, as explained in the Implementation Methodology chapter.

	Execution time (sec)	
	BT	CNN
Matching cost	∼4	∼220
All other stages	∼20	

Table 6.2: Average execution time on Middlebury dataset. Input pairs are size (413...465 x 370) and disparity range is 84.

6.2. KITTI dataset

For this dataset, the disparity range has been set according to the disparity range presented in the ground truth image and its different for every image. This is because it was necessary to save as much computation time as possible, since the images are much bigger than in the Middlebury dataset (1200x375 v/s 460x370). The minimum disparity range used was 51 pixels while the maximum was 100 pixels. Scale factor was also set according to the disparity range of each image and the error threshold for correct matches is 3 pixels.

In table 6.3 there is a detail of the computed errors for both configurations. It is possible to note that the CNN configuration is slightly superior, since both the mean and SD of the overall error are lower than in the BT configuration. The occlusion error is also lower and does not have a higher mismatch error than BT, like with the Middlebury dataset.

	BT configuration		CNN configuration	
	Mean	SD	Mean	SD
Mismatch error (%)	7.32	2.94	7.57	3.53
Oclusion error (%)	28.62	15.29	25.10	13.91
Overall error (%)	35.94	17.61	32.67	15.82

Table 6.3: Results for KITTI dataset. Error threshold is 3 pixels.

In Figure 6.5, there is a display of the output disparity maps of both configurations using input images of Figure 6.4. The results for the first 5 rows of the image pairs are somewhat similar for both configurations, however, in the last image pair (last row, image 181), the images have illumination difference, and the BT configuration fails almost completely to find any matches, having an overall error of 81% on this particular example. On the contrary, the CNN maintains its performance having an overall error of 15% on that image pair. This proofs the theory that deep learning based matching costs are indeed more suitable for outdoor environments that have no control over the illumination or exposure.

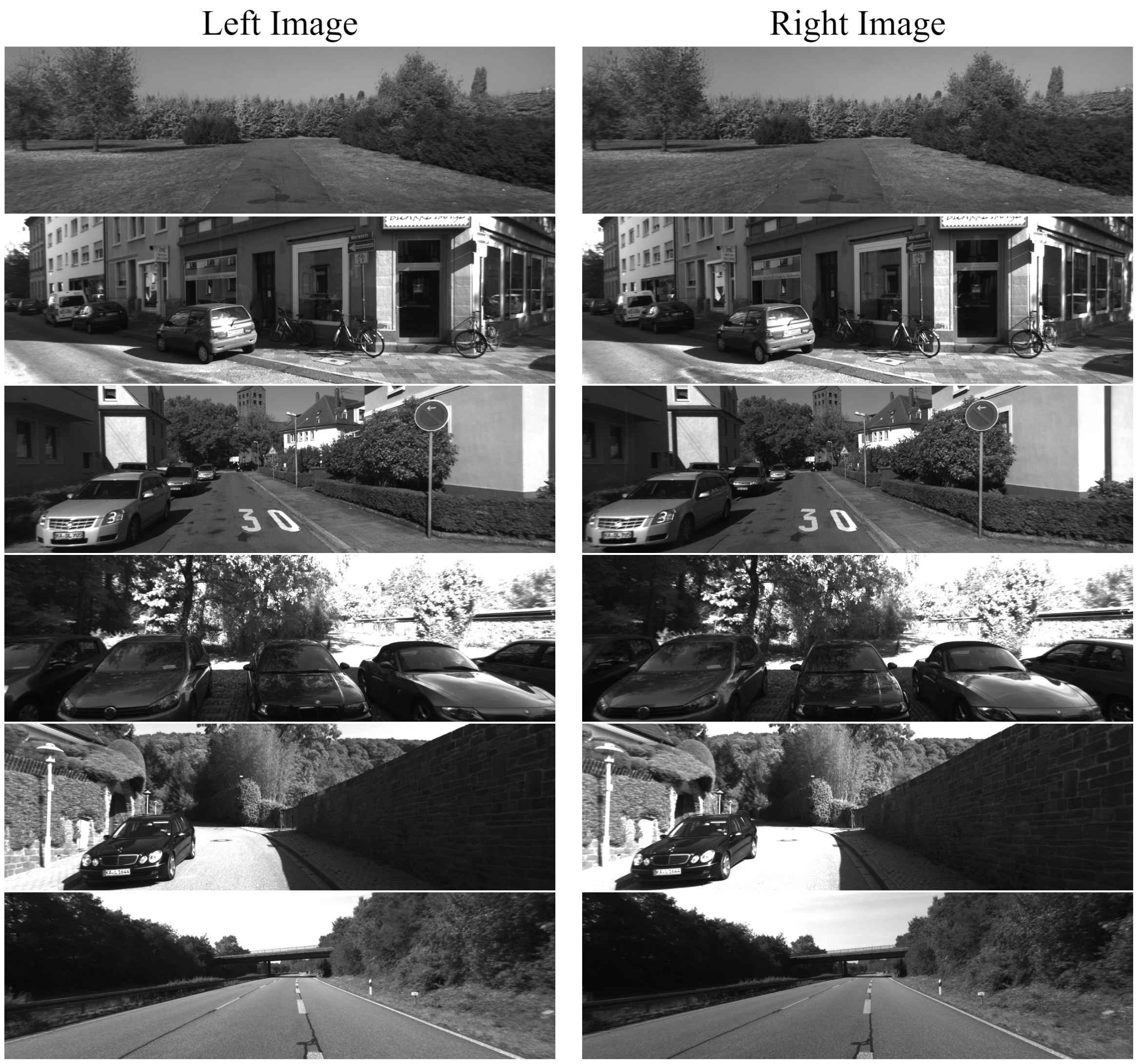


Figure 6.4: Left and right stereo images of the KITTI dataset. Image names are, from top to bottom: “146”, “68”, “110”, “180”, “126” and “181”.

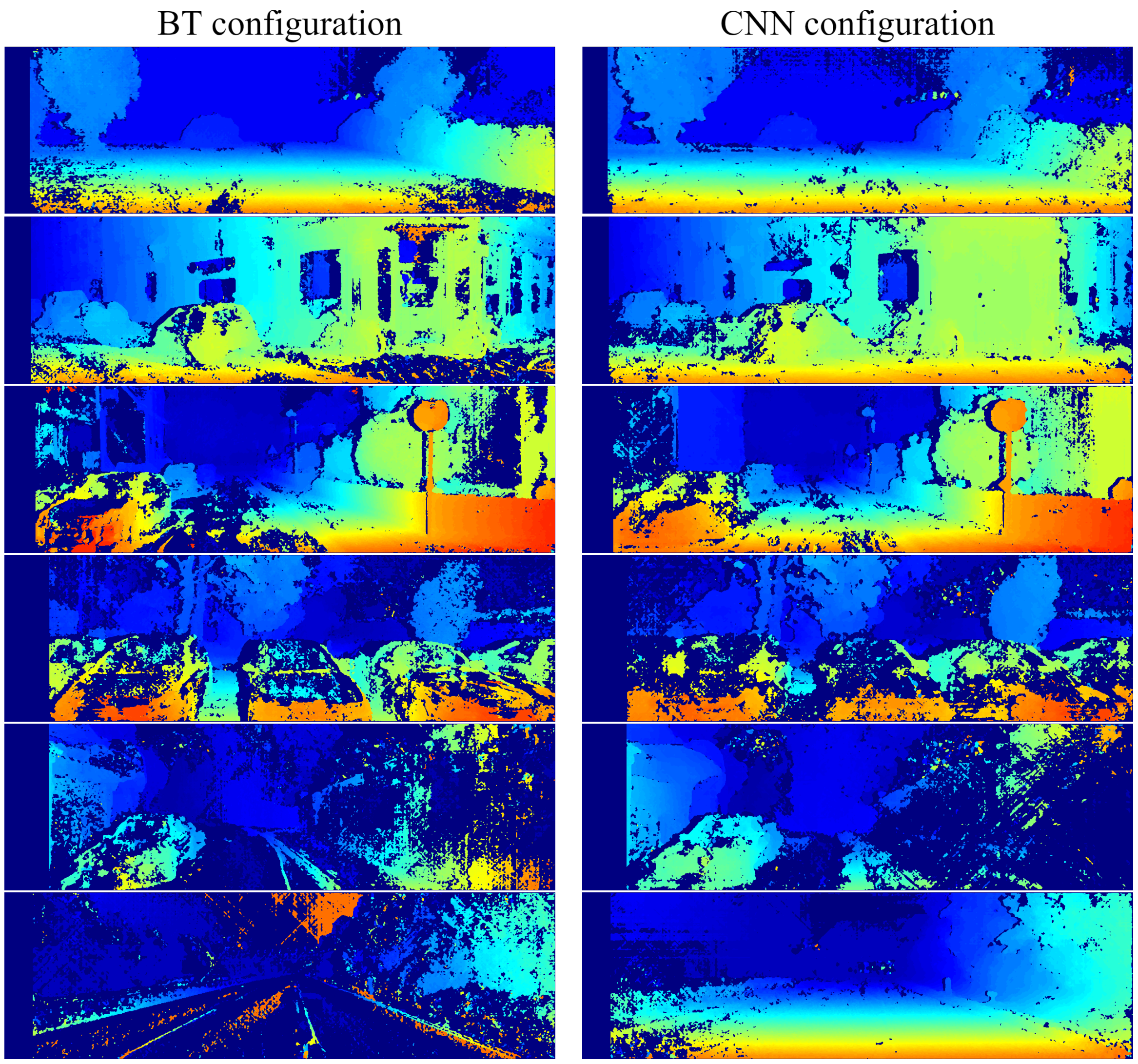


Figure 6.5: Output disparity maps using KITTI dataset (see Figure 6.4). Left column corresponds to the BT configuration output and right column to the CNN configuration.

As well as in Middlebury, the execution time (see Table 6.4) of the BT configuration is much smaller, but since the KITTI images are bigger, the time difference between BT and CNN becomes bigger as well. Still, Žbontar et al [17] reports an execution time on the KITTI images of 95 seconds (only the matching cost computation stage), so in the best case scenario, the CNN configuration would still take longer than BT.

	Execution time (sec)	
	BT	CNN
Matching cost	∼13	∼600
All other stages	∼60	

Table 6.4: Average execution time on KITTI dataset. Input pairs are size (1238...1242 x 374 ...376) and disparity range can be from 51 to 100.

6.3. Truck images

For the truck images, five input image pairs have been chosen, shown in Figure 6.6 (see left columns). They showcase the different illumination situations that might be encountered, since images are captured at any time of day. Not only that, but there can also be an illumination difference between left and right images of the stereo camera. As well as in the KITTI example shown in the previous section (see Figure 6.5, last row), the BT configuration does not perform well when the input pair has illumination difference, hence is not suitable for this type of application on its own, due to the fact that the timber trunks are not recognised in most examples, as shown in Figure 6.6.

On the other hand, the CNN is able to find matches on most parts of the trunks (see Figure 6.6), but the performance its still compromised in some cases, for example in the first image pair (first row), that suffers from poor illumination conditions. Notice that the CNN is not disturbed in the case of a night input pair (last row).

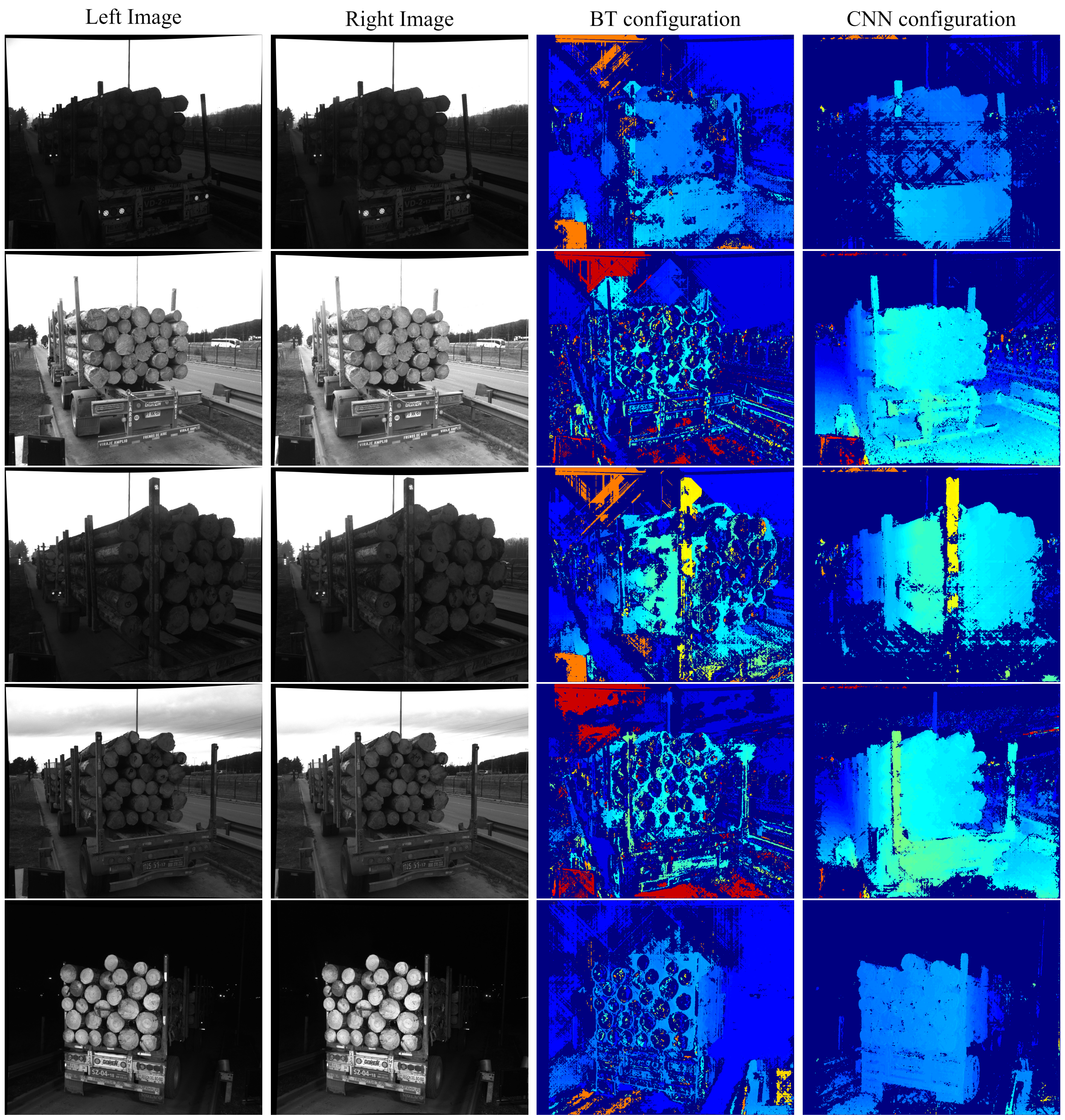


Figure 6.6: Results using the truckload images. Left columns: input images, center right column: BT configuration, rightmost column: CNN configuration.

It is important to note that the SGM (P_1 and P_2) parameters have been changed in the CNN configuration case, making a distinction between the daytime and nighttime images. This is because the night images did not perform well with the daytime parameters. The parameters used for the night images (referred as configuration 1) were $P_1 = 0, P_2 = 0.1$ and for the day images (referred as configuration 2) were $P_1 = 0.001, P_2 = 0.02$

In Figure 6.7 there is a display of the daytime images using CNN configuration 1 and 2. It is observed that with configuration 2, more disparities are recovered. On the contrary, when using configuration 2 on the night images (see Figure 6.8), the top part of the trunks looses its boundaries, but when using parameters of configuration 1 they remain intact.

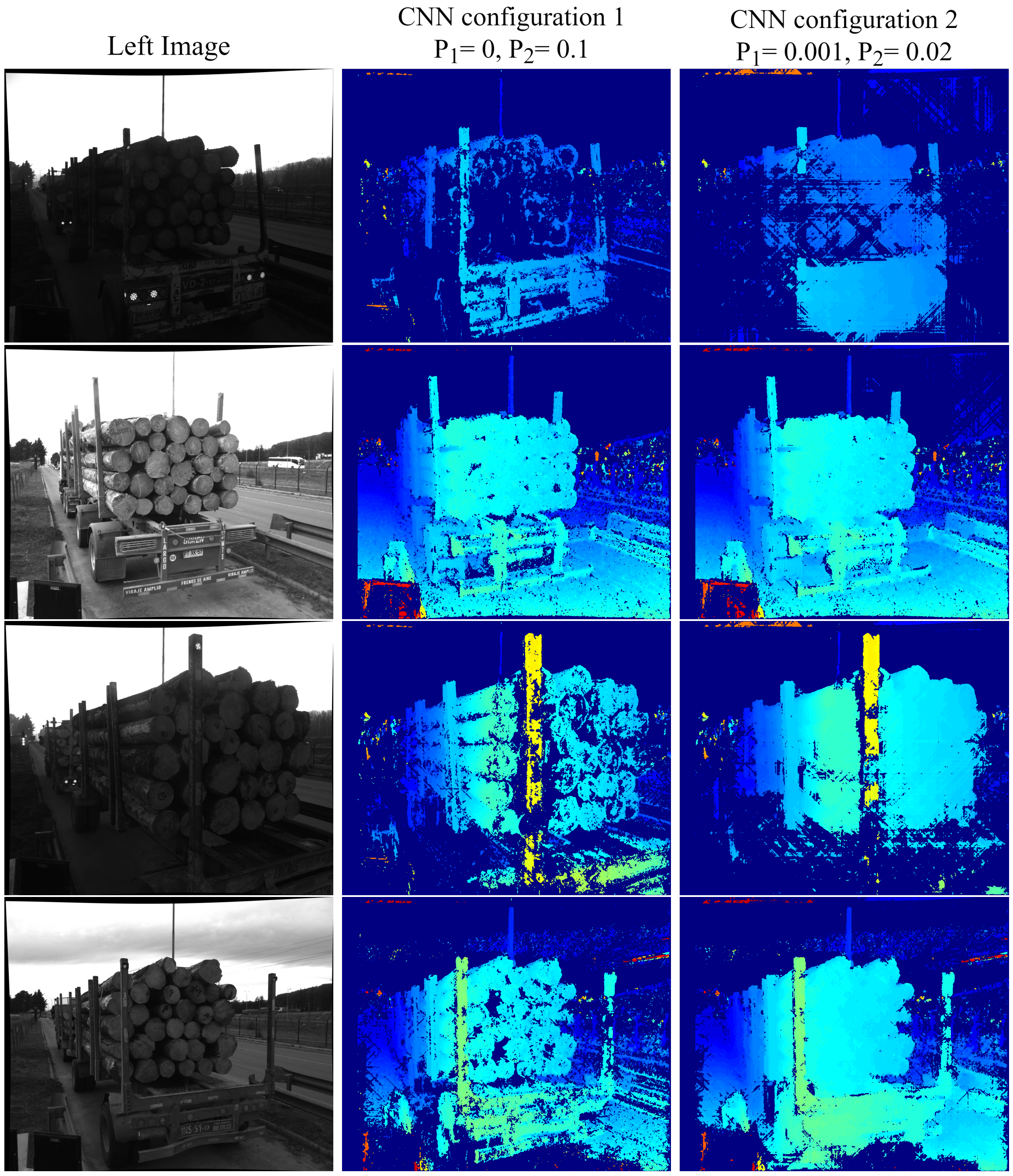


Figure 6.7: CNN configuration results with the same matching cost matrix but different P_1 and P_2 parameters (of semi global matching algorithm). Input images were taken at daytime.

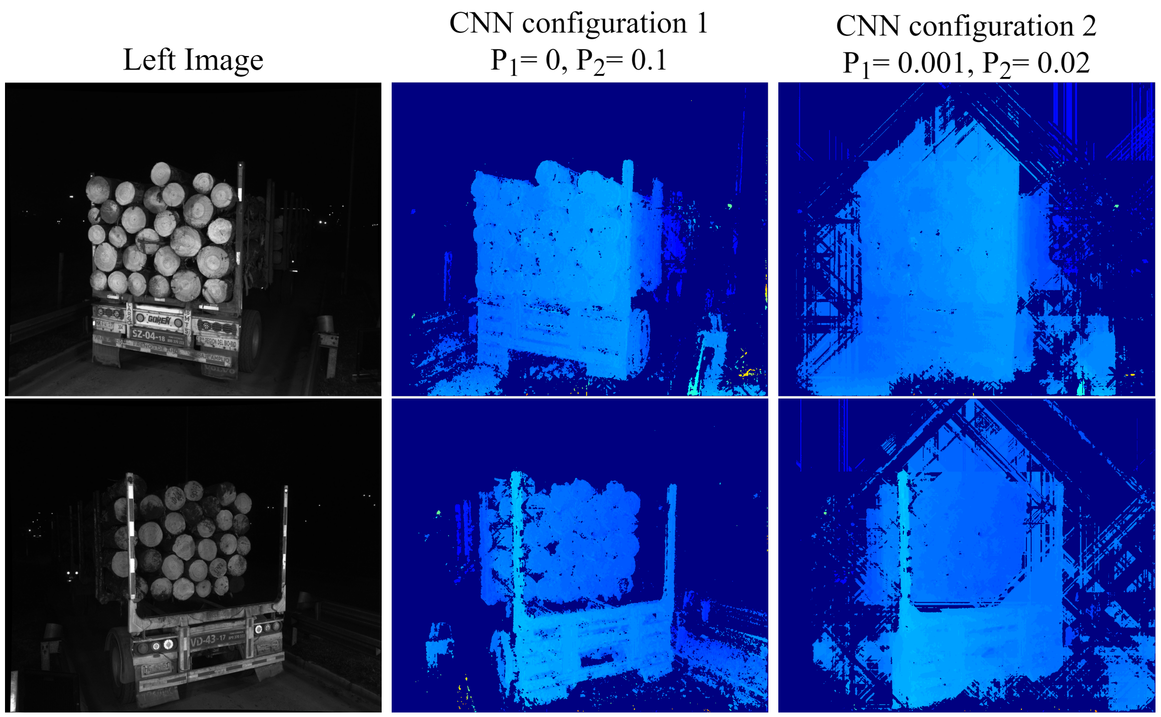


Figure 6.8: CNN configuration results with the same matching cost matrix but different P_1 and P_2 parameters (of semi global matching algorithm). Input images were taken at nighttime.

At last, in Table 6.5 it is possible to see the average execution time on the truck images. As expected, the difference in time between BT and CNN grows even more than in the KITTI dataset, because of the input image size.

	Execution time (sec)	
	BT	CNN
Matching Cost	∼20	∼1270
All other methods	∼100	

Table 6.5: Average execution time on the truck images. Input pairs are size (1000 x 1200) and disparity range is 56.

6.4. Improved results: pre-filtering of the input images

In order to make the BT matching cost function less dependent on the intensity of the pixels, a pre filter step was added to the input images: the Sobel X filter. Unfortunately, it was only added to the BT configuration, since it was not in the initial objectives to train the CNN with prefilterd images, and the time left to do it was very restrained.

This prefilter step decreased the overall error (compared to the original BT) in both datasets (see Table 6.6), reducing it by approximately 15% in the Middlebury dataset and 2% in the KITTI.

	Prefiltered BT configuration			
	Middlebury		KITTI	
	Mean	SD	Mean	SD
Mismatch error (%)	4.20	3.25	2.95	1.31
Occlusion error (%)	13.30	13.27	31.01	10.88
Overall error (%)	17.50	16.22	33.96	11.81

Table 6.6: Results for Filtered BT configuration.

In Figure 6.9 there is a display of an image of the KITTI dataset which did not perform well in the BT configuration, but with the prefilter step is able to recover more disparities making the error drop from 81%, approximately, to 27% after the prefilter step is applied. The same is observed in Figure 6.10, where the disparities recovered after the prefilter step is increased. In the case of an image pair that has no difference in illumination or exposure between left and right images, the prefilter step also showed to be useful, as it is seen in Figure 6.11.

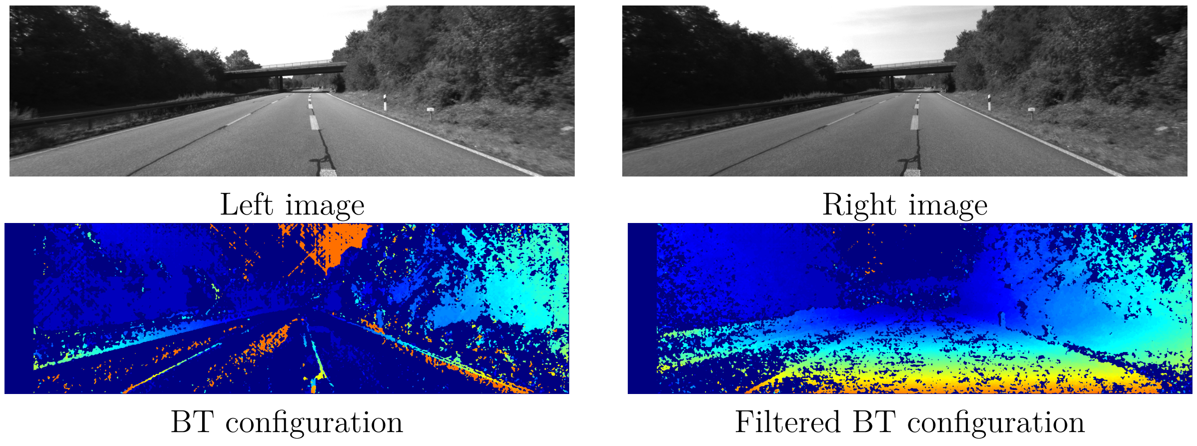


Figure 6.9: Output disparity maps of BT and Filtered BT with a KITTI image with illumination difference.

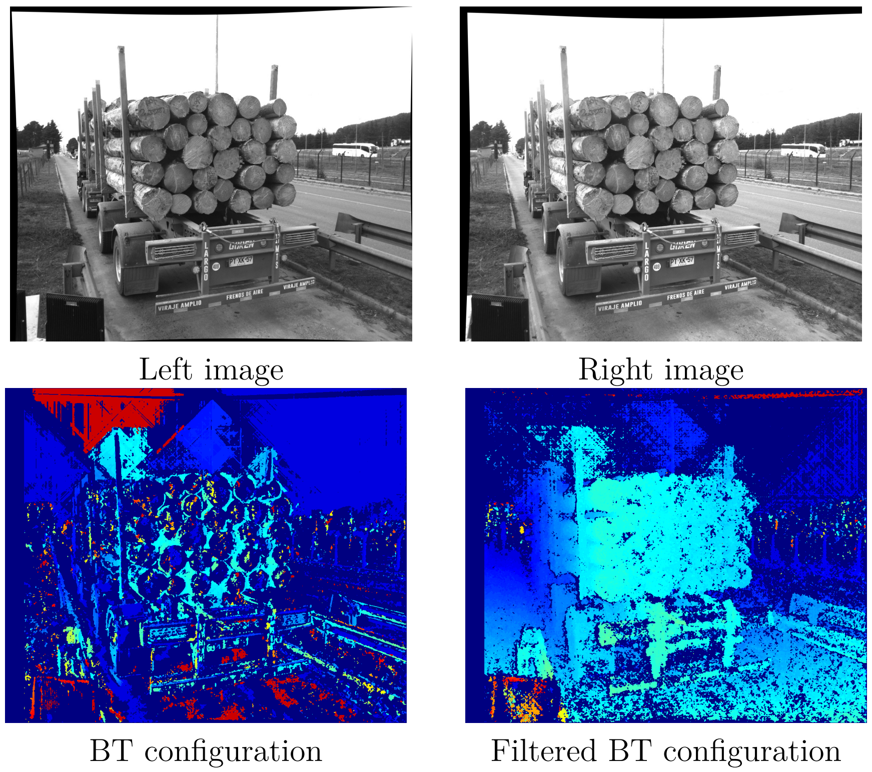


Figure 6.10: Output disparity maps of BT and Filtered BT with a Truck image with illumination difference.

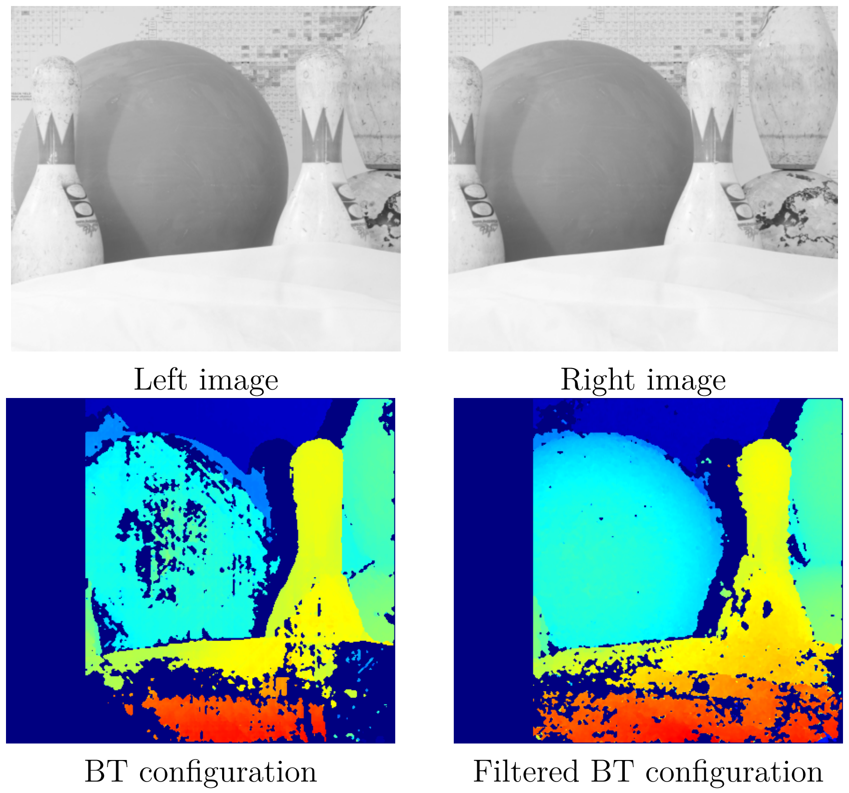


Figure 6.11: Output disparity maps of BT and Filtered BT with a MIddlebury image.

6.5. Summary

The Table below summarises the results mentioned in the previous sections. As stated before, the Filtered BT configuration surpasses the BT in performance for both datasets, making a bigger difference in the Middlebury dataset, with 15% overall error difference. When comparing the Filtered BT configuration with the CNN, it is also better in the Middlebury dataset. However, in the KITTI dataset has a slightly bigger mean overall error than CNN, but this gets compensated with the smaller variance, which makes Filtered BT more stable in terms of the error than CNN.

	Overall error			
	Middlebury		KITTI	
Configuration	Mean	SD	Mean	SD
Filtered BT	17.50	16.22	33.96	11.81
BT	32.25	22.52	35.94	17.61
CNN	33.00	21.63	32.67	15.82

Table 6.7: Comparison of the three algorithm configurations: Filtered BT, BT and CNN.

A comparison of the three implemented algorithms is shown in Figure 6.12. It is seen that Filtered BT made a big impact in the result of the truck images, showing that is able to recover more disparities than the BT configuration on its own, increasing the predicted disparities specially in the trunks area.

When comparing Filtered BT with the CNN, the CNN is able to predict more disparities in the trunks area for most images, but not in the background and floor, however, this is irrelevant to the problem in question (because what matters is the trunks).

Taking a closer look at the input images (Figure 6.12, left column), some of them have poor illumination conditions, yet the Filtered BT configuration maintains its performance in all of them, including the worst illuminated image, which is on the first row of Figure 6.12. The reason for the illumination immunity of Filtered BT is because of the prefilter step of Sobel X. On the contrary, the CNN configuration's performance suffers in the first row image, having a lot of holes on the final disparity.

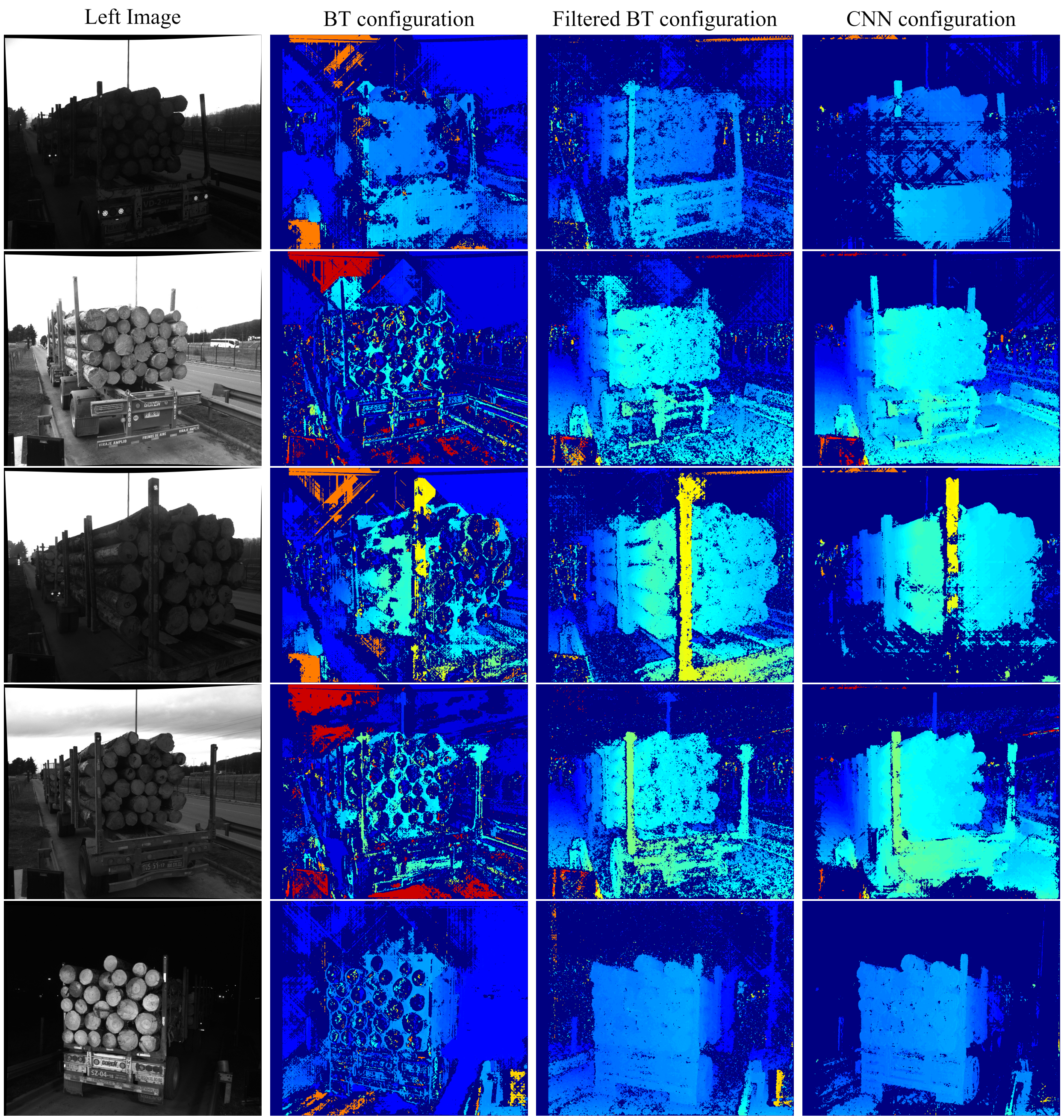


Figure 6.12: [Results of BT, Pre filtered BT and CNN configurations using Truckload images.

Chapter 7

Conclusion and Future Work

In this memoir, a stereo vision algorithm has been implemented, inside the software system of the company Woodtech, leaving an easily modifiable, ready to use implementation. Moreover, it includes two very different approaches for computing the matching cost, one traditional, and one based on deep learning. The two approaches have been tested using two datasets with ground truth disparities, and a qualitative test has been made on a third set of images, of timber loaded trucks, provided by the company.

The two configurations implemented (BT and CNN) have similar performance on the Middlebury dataset, which is indoor and with controlled illumination. This dataset is far to be similar to the specific problem presented at first, but provides a good start when evaluating both algorithms, specially since the ground truth disparities are much fuller than the KITTI (that has sparse ground truth disparities), making the evaluation more accurate.

The KITTI dataset, on the other hand, has a similarity with the truck problem in the sense that both are in outdoor conditions. This dataset has showed that the CNN configuration is able to estimate better disparities even when the input image pair suffers from illumination difference. The BT configuration however, fails to estimate a good amount of disparities in some input images with this issue. Considering this, one can safely assume that the CNN configuration will perform better on images with an outdoor environment.

After making the qualitative analysis on the truckload images with both algorithms, it is clear that the BT configuration is not suitable on its own in this kind of problem, due to its dependency on the intensity of the pixels. This motivated the intention to prefilter the images and make them immune to the luminary issues. When evaluating BT with the prefilter, it ended up performing much better in the indoor Middlebury dataset, but it did not make that much of a difference in the KITTI. Despite this, on the truck images made a big impact when it came to predicting the disparities of the timber trunks, and is able to recover more disparities than the CNN in some images.

Now when it comes to the execution time, the cost aggregation and post processing steps require to be optimized, as well as the CNN matching cost function. It is obvious that none of the implementations are suitable for real time computations and require an in depth analysis of the time complexity, which escapes the scope of this memoir.

Regarding the algorithm benchmark, it is hard to decide which one is best, since there are a lot of factors to consider, for example the execution time. In that sense, it will always be better the Filtered BT approach, that can achieve similar results to the CNN in the KITTI dataset and in the truck images as well. However, if the accuracy of the algorithm is considered more important, then the CNN approach is best, because it obtains the best overall error in the KITTI dataset, and images can easily be downsampled, so that execution time is not too large.

Considering what has been said, there is a lot of room for improvement. When it comes to the execution time, there are two possibilities of improvement, first the matching cost computation stage, and then the cost aggregation step (semi global matching algorithm), which takes the majority of time after the matching cost stage. The CNN case of the matching cost stage is most critical, and it could be improved with the steps detailed by the authors of the architecture [17] (also explained in Chapter 3). Implementing this will result, in the best case, in an execution time of 95 seconds when using the KITTI images, which is still slow for this type of application. So, this tells us that the architecture itself can be improved, and new, more recent architectures could be the solution to the execution time issue, for example, the one detailed in [23], which is able to produce accurate results in less than a second of GPU, using a matching network. Anyway, it is certain that its hard to find an architecture that has a good trade off between performance and time complexity. Now, focusing on the Semi Global Matching algorithm, its execution time can be reduced by making the implementation less memory consuming [24], and even consider to use other types of hardware (e.g GPU, FPGA).

When it comes to the predicted disparity maps, interpolation techniques could be used to fill in the gaps that could not be predicted [18], and even recover the leftmost columns of the final disparity. Also, for the CNN configuration, an additional cost aggregation step is suggested in the literature [17], for example, a cross based cost aggregation technique [19]. Going even further, the CNN could be trained with the truck images for patch comparison, therefore expecting a better performance of the matching function. In addition, preprocessing of the images could be done, to make the illumination conditions equal, for example with histogram equalization of the input images [25].

Last but not least, it has been proven that parameter setting of the Semi Global Matching algorithm is crucial in the output disparity map result. It is important to develop a way in which the P_1 and P_2 parameters are set automatically, using for example the image intensity gradient [18]. This way, the best result could be obtained immediately, instead of having to do a trial and error phase.

Bibliography

[1] Woodtech: A Red to Green Company, “Logmeter”, 2017, https://www.woodtechms.com/logmeter.

[2] Open Source Computer Vision (OpenCV), “Opencv modules”, 2021, https://docs.opencv.org/4.5.2/index.html.

[3] H. Laga, L. V. Jospin, F. Boussaid, and M. Bennamoun, “A Survey on Deep Learning Techniques for Stereo-based Depth Estimation,” jun 2020.

[4] H. Hirschmüller and D. Scharstein, “Evaluation of cost functions for stereo matching,” in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2007.

[5] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, “Vision meets robotics: The KITTI dataset,” *International Journal of Robotics Research*, 2013, http://www.cvlibs.net/datasets/kitti/.

[6] D. Scharstein and R. Szeliski, “A taxonomy and evaluation of dense two-frame stereo correspondence algorithms,” *International Journal of Computer Vision*, vol. 47, no. 1-3, 2002.

[7] R. A. Hamzah and H. Ibrahim, “Literature survey on stereo vision disparity map algorithms,” 2016.

[8] B. Tippetts, D. J. Lee, K. Lillywhite, and J. Archibald, “Review of stereo vision algorithms and their suitability for resource-limited systems,” *Journal of Real-Time Image Processing*, 2016.

[9] X. Xiang, M. Zhang, G. Li, Y. He, and Z. Pan, “Real-time stereo matching based on fast belief propagation,” *Machine Vision and Applications*, 2012.

[10] M. Adams, Elective Course: “Robotics, Sensing and Autonomous Systems,” *Faculty of Physical and Mathematical Sciences, Universidad de Chile*, 2021.

[11] A. Nordmann, “Epipolar geometry,” *Wikimedia Commons*, 2007, https://commons.wikimedia.org/wiki/File:Epipolar_geometry.svg.

[12] C. Loop and Z. Zhang, “Computing rectifying homographies for stereo vision,” *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 1999

[13] S. Birchfield and C. Tomasi, “A Pixel Dissimilarity Measure That Is Insensitive to Image Sampling,” Tech. Rep. 4, 1998.

[14] K. Zhou, X. Meng, and B. Cheng, “Review of Stereo Matching Algorithms Based on

Deep Learning," *Computational Intelligence and Neuroscience*, 2020.

[15] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "The KITTI Vision Benchmark Suite," 2021, http://www.cvlibs.net/datasets/kitti/eval_stereo_flow.php?benchmark=stereo.

[16] S. Zagoruyko and N. Komodakis, "Learning to Compare Image Patches via Convolutional Neural Networks," apr 2015.

[17] J. Zbontar and Y. Lecun, "Computing the Stereo Matching Cost with a Convolutional Neural Network," tech. rep.

[18] H. Hirschmüller, "Stereo processing by semiglobal matching and mutual information," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2008.

[19] K. Zhang, J. Lu, and G. Lafruit, "Cross-based local stereo matching using orthogonal integral images," *IEEE Transactions on Circuits and Systems for Video Technology*, 2009.

[20] S. Hermann and T. Vaudrey, "The gradient - A powerful and robust cost function for stereo matching," in *International Conference Image and Vision Computing New Zealand*, 2010.

[21] M. Brown, "Multi-view Stereo Correspondence Dataset," 2007, http://matthewalunbrown.com/patchdata/patchdata.html.

[22] D. Scharstein and R. Szeliski, "High-accuracy stereo depth maps using structured light," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2003.

[23] W. Luo, A. G. Schwing, and R. Urtasun, "Efficient Deep Learning for Stereo Matching," tech. rep

[24] H. Hirschmüller, M. Buder, and I. Ernst, "MEMORY EFFICIENT SEMI-GLOBAL MATCHING," in *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 2012

[25] T. T. San and N. War, "Local stereo matching under radiometric variations," in *Proceedings - 18th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing, SNPD 2017*, 2017.

Annexes

Error tables

A.1. Error table for Middlebury dataset

	Configuration								
	BT			Prefiltered BT			CNN		
Error (%) - Image name	mism	occ	ov	mism	occ	ov	mism	occ	ov
Aloe	5.16	12.49	17.65	4.10	14.20	18.30	9.92	14.98	24.90
Baby1	7.70	12.77	20.47	1.87	5.72	7.59	12.47	6.55	19.02
Baby2	11.63	9.01	20.64	1.52	6.29	7.81	19.65	7.97	27.62
Baby3	15.06	8.71	23.77	3.55	6.82	10.37	14.40	7.86	22.26
Bowling1	31.52	28.92	60.44	6.63	15.71	22.34	27.95	30.70	58.65
Bowling2	23.79	10.29	34.08	2.05	6.74	8.79	21.86	21.29	43.14
Cloth1	0.37	1.10	1.47	0.22	1.05	1.27	1.17	1.43	2.60
Cloth2	5.28	6.48	11.76	2.09	5.90	7.99	7.93	5.39	13.33
Cloth3	2.26	1.57	3.83	1.72	1.91	3.63	4.93	2.80	7.72
Cloth4	3.30	7.34	10.65	1.53	8.08	9.61	3.96	7.70	11.66
Flowerpots	33.78	7.46	41.24	7.06	7.28	14.34	34.41	21.94	56.35
Lampshade1	14.08	19.52	33.60	6.65	18.01	24.66	20.94	22.84	43.78
Lampshade2	35.46	34.04	69.50	5.71	25.02	30.72	26.17	33.30	59.46
Midd1	30.51	24.37	54.88	9.53	37.93	47.47	35.28	26.14	61.43
Midd2	21.02	40.15	61.17	8.89	35.21	44.10	40.92	24.63	65.55
Monopoly	13.02	40.13	53.15	6.84	12.62	19.46	15.18	12.43	27.61
Plastic	18.16	57.68	75.85	11.61	50.26	61.88	44.60	24.63	69.23
Rocks1	4.63	7.11	11.74	1.92	3.81	5.73	6.12	3.43	9.55
Rocks2	3.65	5.76	9.41	1.96	2.11	4.07	5.27	2.35	7.63
Wood1	8.31	17.98	26.29	1.38	10.77	12.16	14.37	18.02	32.39
Wood2	11.00	24.66	35.67	1.44	3.79	5.22	12.62	16.44	29.06
Mean	14.27	17.98	32.25	4.20	13.30	17.50	18.10	14.90	33.00
SD	11.11	14.97	22.52	3.25	13.27	16.22	12.66	10.00	21.63

Table A.1: Error for each image on Middlebury dataset with the three configurations implemented. "mism" stands for mismatch error, "occ" for occlusion error and "ov" for overall error. The threshold for correct matches between ground truth and prediction is 1 pixel (without scaling).

A.2. Error table for KITTI dataset

	Configuration								
	BT			Prefiltered BT			CNN		
Error (%) - Image name	mism	occ	ov	mism	occ	ov	mism	occ	ov
9	3.73	14.47	18.20	2.20	18.92	21.13	5.04	21.81	26.84
68	6.41	21.32	27.73	1.94	16.50	18.44	7.04	13.72	20.76
89	3.70	11.42	15.12	1.70	20.70	22.40	6.74	25.57	32.31
98	8.46	23.43	31.89	4.61	37.29	41.90	12.92	20.03	32.95
110	5.99	28.85	34.84	2.34	26.13	28.47	3.87	15.43	19.29
116	4.48	15.54	20.01	2.12	19.11	21.23	8.20	13.21	21.41
117	12.59	28.16	40.75	4.13	49.45	53.57	5.01	29.24	34.24
118	8.30	41.13	49.43	3.45	36.84	40.29	9.28	37.07	46.35
126	10.69	61.25	71.94	1.95	33.72	35.68	6.14	64.99	71.12
139	3.36	15.47	18.83	1.70	20.89	22.59	3.14	9.46	12.59
141	7.78	41.61	49.38	3.32	44.19	47.51	12.17	41.26	53.42
146	2.45	10.90	13.35	1.53	15.11	16.64	2.80	5.95	8.75
151	7.43	25.91	33.34	3.22	23.08	26.31	7.54	23.44	30.98
155	7.00	28.54	35.54	2.75	30.99	33.74	8.04	23.66	31.69
162	5.87	20.74	26.61	2.28	41.44	43.72	4.03	19.09	23.12
165	7.84	30.82	38.66	2.16	37.33	39.49	12.23	30.73	42.96
179	7.49	21.38	28.87	6.21	37.46	43.67	10.02	18.39	28.42
180	11.58	33.32	44.90	5.47	49.71	55.18	12.59	42.90	55.49
181	12.13	69.46	81.58	2.20	24.87	27.07	2.55	12.84	15.38
193	9.13	28.70	37.83	3.75	36.48	40.22	12.11	33.22	45.34
Mean	7.32	28.62	35.94	2.95	31.01	33.96	7.57	25.10	32.67
SD	2.94	15.29	17.61	1.31	10.88	11.81	3.53	13.91	15.82

Table A.2: Error for each image on KITTI dataset with the three configurations implemented. “mism” stands for mismatch error, “occ” for occlusion error and “ov” for overall error. The threshold for correct matches between ground truth and prediction is 3 pixels.