THE UNIVERSITY
of EDINBURGH

Specialised global methods for binocular and trinocular stereo matching

Luis Alberto Horna Carranza

Doctor of Philosophy

Institute of Perception, Action and Behaviour

School of Informatics

University of Edinburgh

2017

Abstract

The problem of estimating depth from two or more images is a fundamental problem in computer vision, which is commonly referred as to stereo matching. The applications of stereo matching range from 3D reconstruction to autonomous robot navigation. Stereo matching is particularly attractive for applications in real life because of its simplicity and low cost, especially compared to costly laser range finders/scanners, such as for the case of 3D reconstruction. However, stereo matching has its very unique problems like convergence issues in the optimisation methods, and challenges to find matches accurately due to changes in lighting conditions, occluded areas, noisy images, etc. It is precisely because of these challenges that stereo matching continues to be a very active field of research.

In this thesis we develop a binocular stereo matching algorithm that works with rectified images (i.e. scan lines in two images are aligned) to find a real valued displacement (i.e. disparity) that best matches two pixels. To accomplish this our research has developed techniques to efficiently explore a 3D space, compare potential matches, and an inference algorithm to assign the optimal disparity to each pixel in the image.

The proposed approach is also extended to the trinocular case. In particular, the trinocular extension deals with a binocular set of images captured at the same time and a third image displaced in time. This approach is referred as to $t+1$ trinocular stereo matching, and poses the challenge of recovering camera motion, which is addressed by a novel technique we call baseline recovery.

We have extensively validated our binocular and trinocular algorithms using the well known KITTI and Middlebury data sets. The performance of our algorithms is consistent across different data sets, and its performance is among the top performers in the KITTI and Middlebury datasets. The time-stamped results of our algorithms as reported in this thesis can be found at:

- LCU on Middlebury V2 (https://web.archive.org/web/20150106200339/http://vision.middlebury.edu/stereo/eval/).
- LCU on Middlebury V3 (https://web.archive.org/web/20150510133811/http://vision.middlebury.edu/stereo/eval3/).
- LPU on Middlebury V3 (https://web.archive.org/web/20161210064827/http://vision.middlebury.edu/stereo/eval3/).
- LPU on KITTI 2012 (https://web.archive.org/web/20161106202908/http://cvlibs.net/datasets/kitti/eval_stereo_flow.php?benchmark=stereo).

- LPU on KITTI 2015 (https://web.archive.org/web/20161010184245/http://cvlibs.net/datasets/kitti/eval_scene_flow.php?benchmark=stereo).
- TBR on KITTI 2012 (https://web.archive.org/web/20161230052942/http://cvlibs.net/datasets/kitti/eval_stereo_flow.php?benchmark=stereo).

Acknowledgements

I would like to thank my supervisor Robert Fisher for his detailed guidance and support during the PhD studies. The current thesis would not have been possible without the support from “CONACYT” the National Council for Science and Technology of Mexico, which is funded by the great people of Mexico. The quality of this thesis was greatly improved thanks to the help of my colleagues Radim Tylecek, Matthew Pugh, Steven McDonagh, and Marcelo Saval. The stereo sequences “corridor” and “outside” were captured with the help of Raluca Scona. The feedback from Vitorio Ferrari and Rik Sarkar was thorough, direct, and valuable to improve and steer the direction of my research. Finally, I would like to thank the examiners Paul Siebert and Maurice Fallon for their feedback to improve the final version of this thesis.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified. The material presented in this thesis has contributed to published work. I hereby declare my author contributions to the related publications:

3D plane labeling stereo matching with content aware adaptive windows. L. Horna and R.B Fisher. *International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISAPP), 2017 [37].*

Conceived and designed a 3D plane labelling algorithm; conceived and designed content aware adaptive windows; performed the experiments; wrote the paper.

Plane labeling trinocular stereo matching with baseline recovery. L. Horna and R.B Fisher. *International Conference on Machine Vision Applications, 2017 [38].*

Conceived and designed a motion recovery algorithm; conceived and designed trinocular matching cost function; performed the experiments; wrote the paper.

Luis Alberto Horna Carranza

Table of Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

Using visual information (i.e images) to sense depth is a common feature found in visual systems in nature e.g. humans, cats, dogs, etc. Depth understanding gives much benefit when interacting with the environment, and thus developing algorithms that provide a similar capability is part of the fundamental purpose of computer vision, which aims to develop algorithms that allow computers to measure, interact, or understand the world from images. A very important subset of such algorithms are those that recover depth information from pairs of images, which is considered a low level component for computer vision applications such as 3D reconstruction or autonomous navigation.

Recovering depth using two or more images is known as the stereo matching problem, which can be briefly stated as finding the image correspondences from two or more images. In its simplest form the core issue in stereo matching using two images (left I_l and right I_r views of a scene) is finding the correspondences from each pixel in image I_l to I_r. The displacement that separates two corresponding pixels is known as disparity, which can be either a 1D or 2D displacement (for the sake of simplicity only 1D disparity is considered). However, stereo matching has its very unique problems like convergence issues in the optimisation methods, and challenges to find matches accurately due to changes in lightning conditions, occluded areas, noisy images, etc. It is precisely because of these challenges that stereo matching continues to be a very active field of research.

1.1 3D plane labelling stereo matching

The underlying assumption of stereo matching is that a pixel has a perfect correspondence in the other. The simplest case is assuming integer disparity. However, it is a fact that not all objects are at discrete integer depths, and furthermore such a basic assumption has an implicit bias towards fronto-parallel surfaces, which makes it difficult to recover slanted and curved surfaces.

One possible approach to obtain real valued disparity (i.e. sub-pixel accuracy) is to assume that surfaces can be modelled as a set of small local planes that when evaluated produce real valued disparity. Using planes to estimates disparity has at least two important advantages: the estimated disparity is real valued, and planes better model slanted surfaces. However, using planes presents at least three problems: how to explore a 3D space, how to assign a 3D plane, and how to propagate good plane assignments.

Thus to estimate disparity using 3D planes the stereo matching problem has to be stated as finding a 3D plane per pixel that when evaluated produces the optimal correspondence (using some logical criteria) of two pixels (e.g. left→right). This can be interpreted as assigning a label to each pixel, and therefore this type of procedure is effectively a 3D plane labelling stereo matching algorithm.

1.2 Problem statement

The main topic to address in this thesis is the estimation of real value disparity maps by solving an optimisation problem whose objective is to assign the optimal 3D plane labelling per pixel. Although assigning 3D planes per pixel allows us to recover sub-pixel disparity and handle slanted surfaces its implementation requires to address the following problems:

- 3D space exploration: Estimating the parameters of plane $a*x+b*y+c$ (i.e $(a,b,c) \in \mathbb{R}^3$) is a very large space to traverse and therefore some criteria is needed to explore it smartly.
- 3D plane bias: Assigning 3D planes introduces a strong bias towards planes, which requires a mechanism to limit it.
- Hypothesis generation: 3D plane labelling implies that multiple plane hypotheses need to be generated and evaluated somehow.

In addition to these problems a 3D labelling algorithm still has to deal with the common stereo matching problems such as:

- Textureless regions/surfaces: Such areas requires developing criteria to limit error in disparity estimation.
- Adaptive cost function: Evaluating the similarity/dissimilarity of two pixels requires developing criteria to adaptively change the pixel matching function based on both image content and underlying 3D surface.

1.3 Thesis objective and hypothesis

The proposed research aims to develop a disparity estimation algorithm that can incorporate the benefits of optimisation techniques to compute a disparity map, include a mechanism that allows 3D plane labelling, disambiguates incorrect matches in the matching process, handles perspective distortions, and propagates disparity measures in occluded/textureless/self-similar areas when possible. In order to achieve such objectives the proposed research will attempt to validate the following hypothesis:

A 3D plane labelling algorithm that exploits the underlying 3D structure and image entropy to generate an adaptive matching window, and uses hypothesis generation as a propagation scheduler that is capable of estimating real valued disparity maps by smartly exploring a 3D search space, and generating multiple hypotheses from a single initial hypothesis.

To validate this claim the proposed algorithm is evaluated using the Middlebury, and KITTI (2012, 2015) benchmarks, where the proposed approach is among the top performing results.

1.4 Original contributions

The proposed approach first addresses the issue of a 3D labelling for binocular stereo matching with the following contributions (see chap.5):

- Content aware adaptive window aggregation: Reduces error and loss of details.

- Use of a cost function that imposes local hypothesis uniqueness: Helps to handle textureless surfaces, and does not required higher order interactions in a MRF.
- Use of a cost function that penalises disparity values outside a defined search range: Prevents invalid disparity values assignment.
- Use of a single global hypothesis per disparity plane: Eliminates the need to update multiple hypotheses.
- Use of a hypothesis generator that acts as a propagation scheduler: Helps to do a search in a 3D space.

The proposed approach is extended for the trinocular case, in particular the trinocular extension deals with a binocular set of images captured at the same time and a third image displaced in time. This approach is referred as to $t+1$ trinocular stereo matching, and poses the challenge of recovering camera motion, which is addressed by a innovative technique we call baseline recovery (see chap.6).

1.5 Thesis overview

The main topic addressed in this thesis is that of binocular/trinocular stereo matching using 3D plane labelling. The theoretical foundations and basic optimisation algorithms used for stereo matching are covered in chapter 2. The most relevant work in the state of the art stereo/optimisation algorithms is covered in chapter 3. In chapter 4 we develop a fronto-parallel stereo matching algorithm that is able to recover slanted surfaces along the epipolar direction, as well we develop local cues that improve the classical pairwise energy minimisation stereo matching approach. Our contributions for binocular stereo matching are covered in depth in chapter 5 where the performance of our approach in both indoor and outdoor environment are evaluated. In chapter 6 our approach is extended to work with three images, but most importantly we develop a novel algorithm to recover camera pose known as “baseline recovery”. Finally, in chapter 7 the implementation details of our stereo matching and baseline recovery algorithms are discussed in terms of practical considerations, e.g floating point issues, exploiting modern hardware architecture and parallel computing.

Chapter 2

Stereo matching background

This chapter presents the basic principles used to solve the stereo matching problem. Although posing the stereo matching problem as search for the optimal correspondence/disparity is simple, it requires one to address three basic issues:

1. How are images transformed to do the matching?

2. How are images compared?

3. How is a match computed?

From these three questions the first one is a very well studied geometric problem for which conditions and solutions exist. In this chapter only the most relevant parts of the geometric problem are covered, since an extensive discussion is outside the scope of this thesis. The other two questions have gone through significant progress for several years, and there still remain areas that are very actively researched, e.g. how to handle noisy images, radiometric differences in the images and models to solve the stereo matching problem. In particular this chapter covers the most used similarity/dissimilarity functions and optimisation algorithms.

The theory and algorithms described in this chapter covers dense passive stereo matching, i.e. static images, therefore stereo matching algorithms such as those using structured light, random patterns or matching sparse keypoints will not be covered, although many of the algorithms described here may very well be applied in those situations.

2.1 Historical context

The history of the stereo matching problem is related to photogrammetry. The most relevant work can be traced back to the 19^{th} century including the the work of Sturms and Hauck (in 1883) [22] who developed models to establish a relationship between projective geometry and photogrammetry. Around the same period of time Finsterwalder [20] developed the first algorithm for position resection using correspondences from stereo pairs. This type of approach led to the development of stereo comparators for measurement such as the one from Pulfrich [20]. The next breakthrough contribution in photogrammetry was made by Schut (in 1957) with the introduction of the coplanarity condition for stereo images [26]. Gilbert Hobrough developed the first stereo matching correlator [47], which was implemented as an electronic device to assist with the creation of maps.

The modern approaches for stereo matching can trace their origins back to Marr-Poggio in 1976 who introduced the cooperative algorithm for estimation of disparity, but most importantly introduced the uniqueness and smoothness constraints to estimate a dense disparity map [63] using synthetic images. The TINA 3D vision system (in 1988 [77]) is one of the earliest binocular stereo matching applications for robotics and interaction with the environment, which shows the importance of 3D perception in computer vision.

The decades of 1990 and 2000 saw important progress in stereo matching (e.g. pixel cost similarity [8, 115]) due to the availability of enough compute power, and the introduction of efficient algorithms for inference in Markov Random Fields particularly Graph Cuts [95] and message passing [44, 114]. This brief historical overview consists only of binocular approaches, which is the main topic of our research, covering the history of other stereo approaches in out of scope of the current thesis. However, for further information about multi-view stereo see [22, 11, 26].

From this historical context is clear that using images for measurement or interaction with a 3D environment has been a very active area of research over a long period. This can be attributed to the simple idea of using images to recover a 3D description, but at the same time due to the complexity of the algorithms required to accomplish this task.

2.2 Notation used

The algorithms presented below evaluates all possible disparity planes $d \in \mathbb{Z}$ at each image pixel $p \in \mathbb{Z}$. When referring to the optimal disparity plane assignment the value $D_p \in \mathbb{Z}$ is used. When referring to all possible disparity values at p, the value $d_p \in \mathbb{Z}$ is used. This chapter uses the following notation:

- Disparity plane is a 1D integer displacement along the horizontal direction at each pixel.
- Disparity plane and label are used interchangeably unless otherwise stated.
- $dis(p,q) : \mathbb{R}^2 \rightarrow \mathbb{R}$ is distance function between pixels p and q in the same image.
- $M^t_{pq}(d_p) : \mathbb{R}^n \rightarrow \mathbb{R}^n$ computes a message from pixel p to q, at iteration t for disparity d_p, and n is the number of disparity planes.
- $w_{pq} \in \mathbb{R}$ is a weight factor between two pixels.
- These abbreviations are used: *LBP* (Loopy Belief Propagation), *TRW*-S (Tree Re-Weighted), *CG* (Graph Cuts).

2.3 Image rectification

The first step to solve a problem is to define constraints to simplify it. In the case of the stereo matching problem addressed the current research assumes that two cameras (e.g. left and right) are separated by a baseline $\overline{CC'}$ as in fig.2.1.

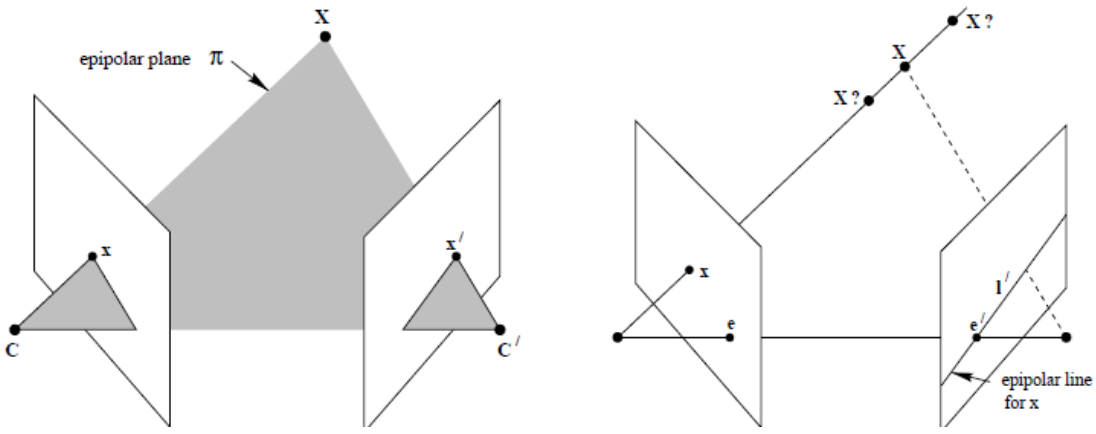
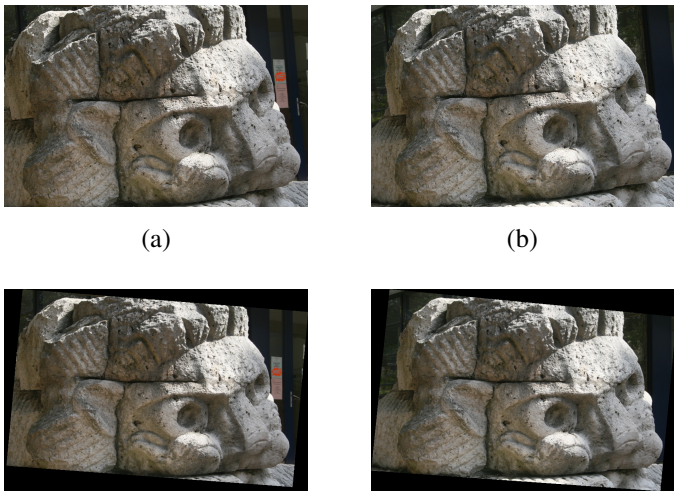


Figure 2.1: Epipolar constraint from [30].

In fig.2.1 (left part), C is the left camera, C' the right camera, and X is a point in 3D space with x and x' being the projection on the left and right image camera planes. Notice that X, x, and x' lie in the same plane π, which is known as the epipolar plane. Note in fig.2.1 (right part) that the intersection of the epipolar plane with the image plane forms a line l' known as the epipolar line, which connects the epipoles (projections of the camera centres) e and e' with the points x and x'.

The existence of the epipolar line creates a very useful condition in which a point that lies in the epipolar plane must also lie in the epipolar lines of both left and right cameras. This is known as the *epipolar constraint* and in the case of left and right cameras, it limits the area of the image where the search for matching points must be done, i.e. finding the correspondence from $x \rightarrow x'$.

To apply the *epipolar constraint* the rotation and translation relative from left to right camera (or right to left, order is not relevant) must be known, which can be obtained once cameras have been calibrated using standard algorithms such as those described in [30]. Using the camera calibration makes it possible to compute the epipolar lines and to do a search for a match. However, the search in the original image is computationally expensive and thus the scanlines of two images are aligned, which is known as image rectification.



(a) (b)

(c) (d)

Figure 2.2: Image rectification: (a) left original image; (b) right original image; (c) left rectified; (d) right rectified; data set from [69].

The process of image rectification is better described as applying a transformation such that the epipoles lie at infinity, which in practical terms means that the epipolar

lines become parallel. However, it is not enough just to map the epipoles at infinity as this alone does not guarantee that the images will have the same scaling or shearing. Thus image rectification also requires the estimation of an optimal image transformation such that scaling and shearing are almost the same. The best example of this procedure is [57], which addresses all of these issues.

Fig.2.2 shows left (fig.2.2a) and right (fig.2.2b) images before rectification, while fig.2.2c and fig.2.2d is the rectified image pair, which shows how scaling is identical and rotation has been removed. Finally, it is worth mentioning that to compute the image rectification it is not always necessary to know the camera pose, but instead sparse image point matches (e.g. $x \rightarrow x'$) can be used to compute the fundamental matrix F (see [30] for full details) such that $xFx' = 0$ (i.e. x and x' lie in the same epipolar plane), and then F can be used to rectify the images using [57].

2.4 Pixel matching cost

An important part of the stereo matching process is assigning a matching cost for a pair of pixels (x, x'), assuming that images have been rectified. The function that evaluates similarity or dissimilarity is known as the cost function, which compares each potential pixel match either using the image intensity or a transformed version of the image. For the sake of simplicity the examples given in this section only use fronto parallel planes to evaluate the matching cost, and image intensity is used to describe the principles behind pixel cost functions, and the disparity plane is assumed to be integer valued. Fig.2.3 shows two scanlines and possible left→right image matches.

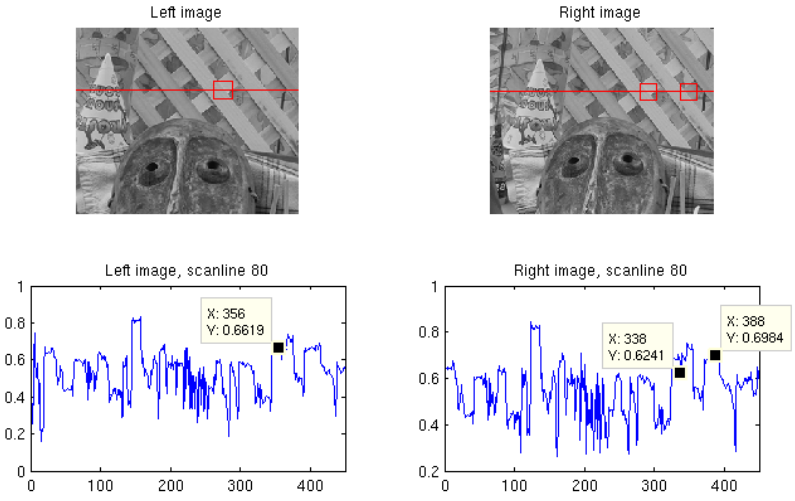


Figure 2.3: Intensity profile and possible pixel matches.

There are two possible ways to evaluate the matching cost in this example (fig.2.3): one possibility is to compare each pixel (*pixelwise matching cost*) in the scanline, which has the evident drawback of being too sensitive to image noise or brightness changes, another option is to make use of a neighbourhood window (*block matching cost*) around the pixel to compute the matching cost (red windows in fig.2.3). It becomes clear that the resulting matching cost may give ambiguous results (two similar windows in fig.2.3), with more than one possible match having low cost. Another possibility is that no match actually exists due to occlusions. A third issue is noise in the image producing low matching costs that are incorrect. One approach to obtain better matches is to increase the window size, which could alleviate the ambiguity issue to a small degree, because the ambiguous regions can always be larger than the window size used. The matching cost is usually stored as 3D array, as in fig.2.4, where each slice represents the cost of each image pixel at disparity d.

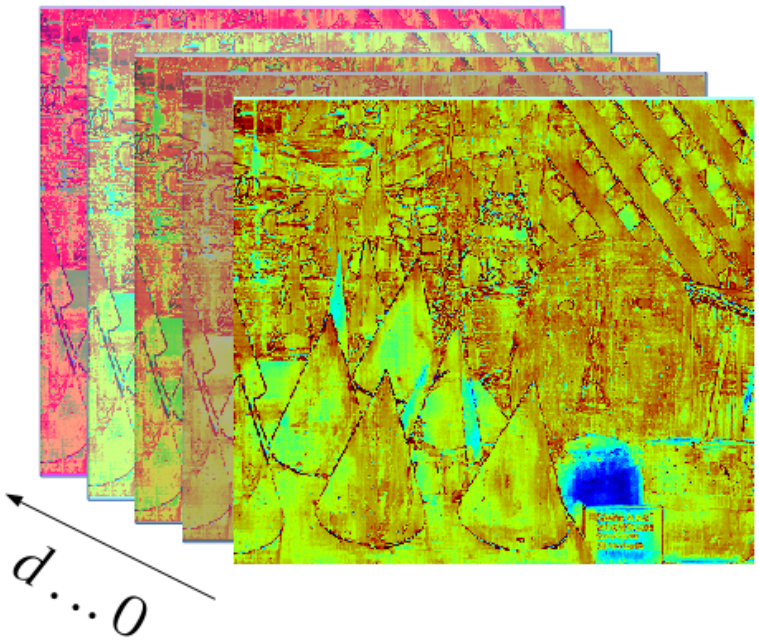


Figure 2.4: 3D cost array.

In order to compute the matching cost of two pixels/windows from left and right images, there are several proposed measures, for instance: Sum of Absolute Differences (SAD), Sum of Square Differences (SSD), Normalised Cross Correlation (NCC), see [35] for a full comparison. The SAD and SSD are measures of dissimilarity that only use pixel intensity, and thus suffer from variations of image brightness and contrast. Note that they should give a low value when the regions matched are similar,

however this does not give enough information about how well two areas match because a low value could be produced even if the two areas are different, e.g. a large surface with the same intensity value produces the same score at all positions. NCC is a similarity measure that provides bounded values to indicate how well two areas match, a value of $+1$ indicates a perfect match while a value of -1 indicates that the areas are perfectly anti-correlated.

To make the measures insensitive to pixel intensity values, two transforms can be used, e.g. the Rank and Census transforms from [115]. Both of them use a neighbourhood around the pixels, where each pixel is given a value taking into account its relative order with respect to the local intensities of its neighbours. After pixels are transformed a similarity/dissimilarity measure can be used to match either each pixel or window around a pixel. The choice of a pixelwise or window cost has two well known issues [82]:

- Pixelwise matching cost suffers from issues in textureless areas and is too sensitive to noise.
- Block matching cost suffers from the "fattening effect", which means the foreground is overgrown.

One possible way to overcome some of these issues is to improve the raw pixel cost by doing aggregation [112] using the reference image (e.g. left) as a guide. Assuming that the raw matching cost $C_p(d_p)$ (e.g. NCC) has been computed, then using an aggregation algorithm the reference image is used as a guide to filter the raw matching cost producing the aggregated cost $\hat{C}_p(d_p)$. This is better described as follows:

$$\hat{C}_p(d_p) = \sum_{q \in N(p)} S(p,q) R(p,q) C_q(d_p) \tag{2.1}$$

where:

$$S(p,q) = \left\{ \frac{1}{Z_s} e^{-\frac{dis_S(p,q)}{\sigma_s}} \mid Z_s = \sum_{q \in N(p)} e^{-\frac{dis_S(p,q)}{\sigma_s}} \right\} \tag{2.2}$$

$$R(p,q) = \left\{ \frac{1}{Z_r} e^{-\frac{dis_R(p,q)}{\sigma_r}} \mid Z_r = \sum_{q \in N(p)} e^{-\frac{dis_R(p,q)}{\sigma_r}} \right\} \tag{2.3}$$

In eq.2.1 $\hat{C}_p(d_p)$ is the aggregated $C_q(d_p)$ similarity/dissimilarity function evaluated at the same disparity d_p and positions p and q, under the assumption that neighbouring pixels at the same disparity level and intensity colour might have similar costs.

The $S(p,q)$ function is the spatial term, and $dis_s(p,q)$ measures the distance (e.g. L_1) in the spatial domain. The $R(p,q)$ function is the image range term and $dis_r(p,q)$ measures the distance in the intensity/colour domain, e.g. L_1 distance.

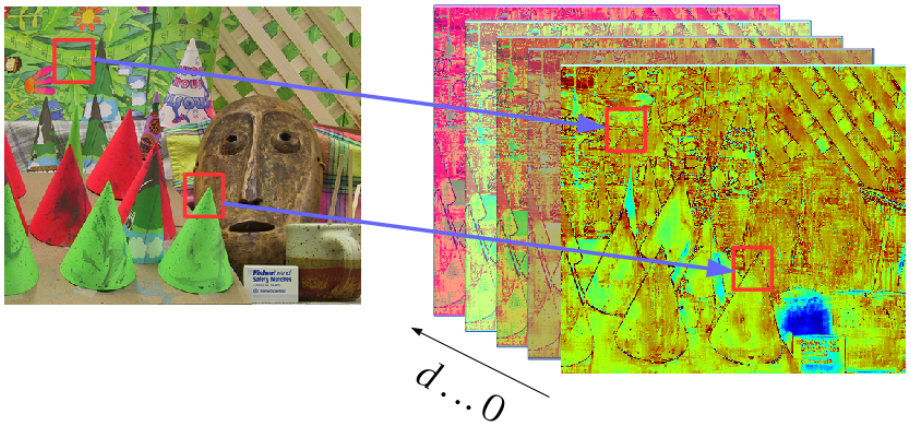


Figure 2.5: Aggregation process.

Fig.2.5 shows the inputs to the aggregation process, where pixels that lie in the same disparity (i.e. fronto parallel plane) are assumed to have a similar cost, and thus adding the cost from neighbouring pixels (red boxes in fig.2.5) based on their intensity similarity or distance may improve the cost and reduce noise in the cost. Aggregation can also be described as a joint bilateral filter, or an adaptive filter (i.e. adaptive window).

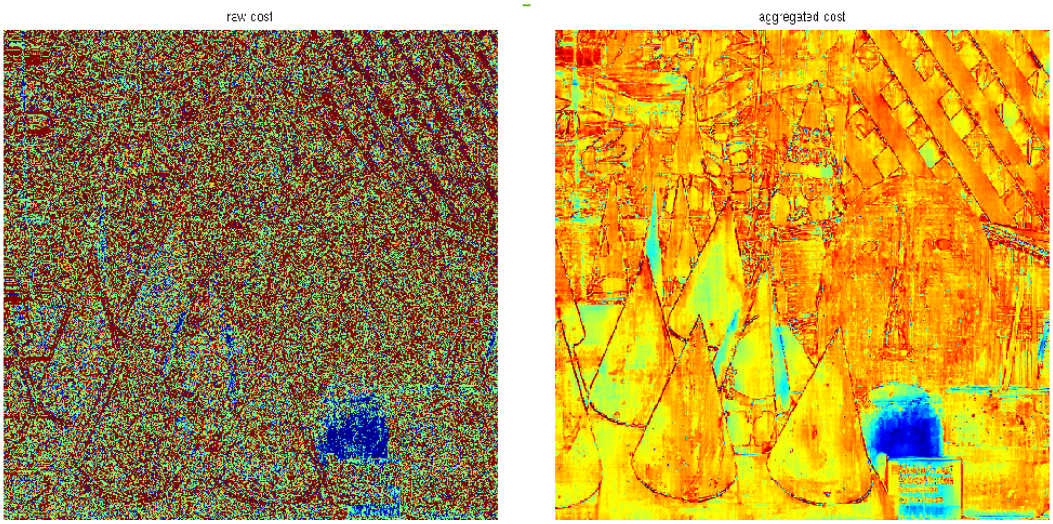


Figure 2.6: Raw vs Aggregated cost; cost slice computed at disparity 80.

To better illustrate the result of aggregation fig.2.6 shows the comparison of raw pixelwise cost at disparity $d = 80$ using SSD (with the images in fig.2.7) and the result

after doing aggregation using a 41×41 neighbourhood. The aggregated cost has less noise and thus may be more robust to noise. Also note that edges in the aggregated cost appear sharp and aligned with the colour image in fig.2.6 due to the adaptive weights in the aggregation window, which can help to match sharp details.

(a) left (b) right

Figure 2.7: Cones reference images [82].

The cost functions and adaptive windows described are the most widely used, and represent the bare minimum required to understand the stereo matching cost computation. It's worth mentioning that there are other cost function such as mutual information (*MI*) [34] or dense descriptors such as DAISY [90], but they were not analysed here.

2.5 Stereo matching as an optimisation problem

In order to assign an optimal disparity plane to each pixel two things are required. First, a model describing the stereo matching problem is needed, i.e. states how matches are evaluated and what constraints are put in place. Second, a mechanism to fit the model is needed, in other words an algorithm that can be used to find the optimal assignment.

To model the stereo matching problem as an optimisation, one possibility is to state it as minimisation problem, where the objective is to find the disparity plane assignment that minimises a function (energy minimisation), which measures the cost of assigning a disparity value taking into account the matching, smoothness and occlusion costs; a commonly used model[45] is:

$$\hat{D} = \arg\min_{D} E(D) \tag{2.4}$$

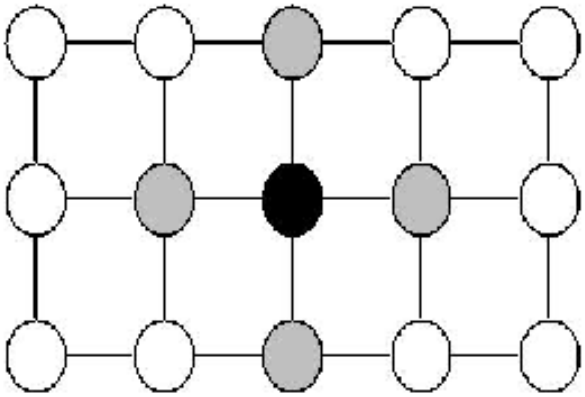


Figure 2.8: MRF representing eq.2.5 without the occlusion term.

where $E(D)$ is the current cost of the disparity assignment D (energy of the whole image), and is given by:

$$E(D) = \sum_{p}^{NumP} \{E_{data}(D_p) + E_{smooth}(D_p) + E_{occ}(D_p)\} \tag{2.5}$$

$$E_{smooth}(D_p) = \sum_{q \in N(p)} V_{pq}(D_p, D_q) \tag{2.6}$$

In eq.2.5 there are three terms: $E_{data}(D_p)$ is the data term (i.e. pixel cost), $E_{smooth}(D_p)$ is the smoothness term that evaluates how well does a disparity value fit in its neighbourhood (e.g. 4-connected), $E_{occ}(D_p)$ is a penalty for disparity planes whose assignment would generate an occlusion. Note that $NumP$ is the number of pixel in the reference image. This type of model makes the assumption that each possible combination of disparity planes per pixel is evaluated during minimisation, which due to the presence of the smoothness and occlusion terms makes it an NP problem. One consequence of the smoothness term is that it introduces the need to represent the relationship of neighbours, which is conveniently achieved by representing eq.2.5 as Markov Random Field (MRF). Each node in the MRF is a pixel, which can have several states assigned (i.e. disparity planes) and is connected to its neighbours by the MRF edges. Once eq.2.5 is represented as an MRF (e.g. fig.2.8) the minimisation problem is equivalent to assigning the most likely state to each node, i.e. do inference in a MRF.

Despite eq.2.5 being an NP problem there are two possibilities for solving it or at least finding an approximate solution:

1. Relax the model so that it can be solved.

2. Use an algorithm to find an approximate solution.

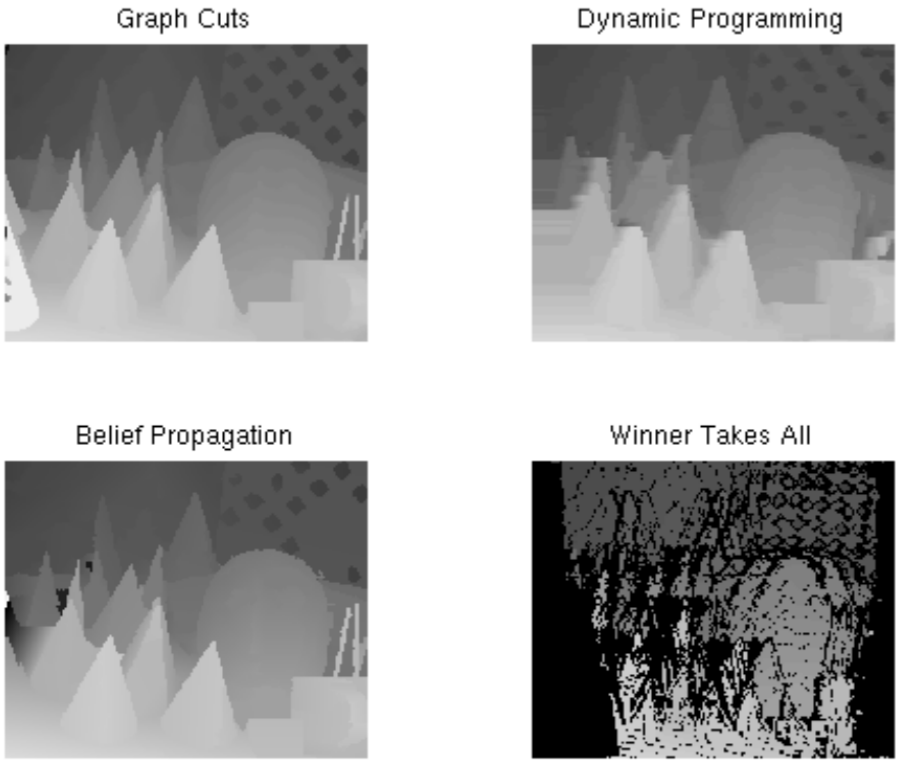


Figure 2.9: Global vs Local methods (cone image from [82]). Graph Cuts [83], Belief propagation [110], Dynamic Programming [100]. Image shows optimal assignment for each algorithm type.

Indeed, these two possibilities are widely used and give rise to a classification of the optimisation algorithm (according to [82]) depending on how the minimisation problem is solved. Thus the optimisation algorithms used can be classified into two main categories:

- Local algorithms: Make the assumption that each pixel/block is matched independently and only the disparity plane with the lowest cost is selected, which is known as winner takes all (WTA), e.g. the modern implementation of MSSM by [14]. The second assumption is that only neighbours in the same 1D path are considered (e.g. same scanline). One example algorithm is Dynamic programming, which gives exact results along the 1D path [82].

- Semiglobal algorithms: Make the assumption that neighbouring pixels are connected (e.g. 4-connected), but finds an exact solution for multiple overlapped

1D paths, which finds an approximation the global minimum. Examples include [34, 19].

- Global algorithms: Make the assumption that neighbouring pixels are connected (e.g. 4-connected) and finds an approximation of the global minimum, because eq.2.5 is an *NP* problem. Examples include Graph Cuts ([45, 101, 36]), Belief propagation ([43, 110, 113]), Simulated Annealing ([4, 86]) or genetic algorithms [7].

In practical terms the major difference between global and local methods is the resulting quality of the disparity map. Global methods will produce disparity maps that approximately minimise eq.2.5. Local methods tend to produce disparity maps that reduce the cost of disparity, but are not good at producing smooth disparity maps. Fig.2.9 shows a comparison of local and global methods; Graph Cuts [83], Belief propagation [110], Dynamic Programming [100], and WTA.

2.5.1 Global optimisation

This thesis makes use of global optimisation, therefore it's relevant to give a brief overview of the most popular algorithms used to minimise eq.2.5. For the sake of simplicity the occlusion term is dropped to make examples and explanations easier to understand. There are two major families of optimisation algorithms to do inference in an MRF: graph cuts and message passing.

2.5.1.1 Graph cuts

Disparity estimation can be seen as a labelling problem where disparity values are labels assigned to certain regions. Then estimating disparity can be stated as finding the optimal labelling that minimises eq.2.5 represented as a MRF. An important observation is that the computational complexity of solving the multiple labelling problem is *NP*, which could make the problem of finding a disparity map intractable. However, if the problem is only finding the solution to a binary labelling problem then a solution can be found in polynomial time [27]. Exploiting this idea [95] states the multiple labelling problem as a series of binary labelling problems where the optimal labelling is found individually for each of the labels. Finding the optimal binary labelling is stated as trying to find the max flow from label S (source node) to label T (sink node), where the nodes (pixels with an initial disparity labelling) are connected (initially) to both

S and T with t-links and each node N_i is connected to its 4-neighbours with n-links. Fig.2.10 illustrates this situation for the 1D case.

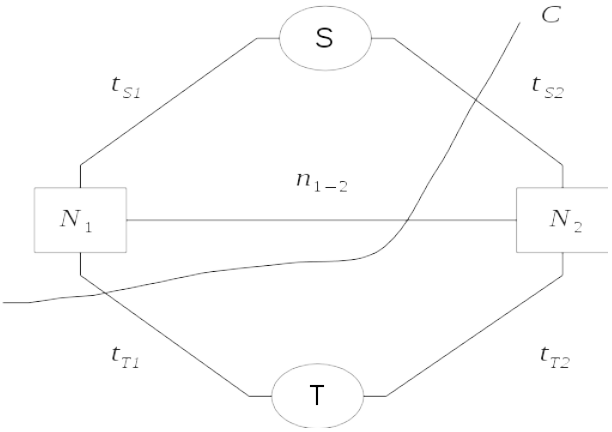


Figure 2.10: Cut C leaves N_1 with S and N_2 with T.

The cut that leaves each node N_i with exactly one t-link is the cut that separates the regions S and T. However, three points need to be addressed: 1) how are t-links and n-links assigned weights?, 2) how is the cut found? and 3) is there any guarantee that the cut is optimal?. The most important contribution of [95] is the answer to those three questions, where two algorithms to find the optimal labelling are given as:

- α-β swap: find an optimal label swap move that reduces energy, see fig.2.11(b).
- α-expansion: find an optimal move that expands label α, see fig.2.11(c).

a) Initial labelling b) $\alpha-\beta$ swap c) α expansion

Figure 2.11: Swap and expansion moves (source [95]) for the general case of multiple labels α, β and γ.

These two algorithms impose a restriction on the smoothness term, which must be either a semi-metric (α-β swap) or a metric (or more generally sub-modular for α-expansion). The conditions that must be satisfied are:

$$V(\alpha,\beta) = 0 \iff \alpha = \beta \tag{2.7}$$

$$V(\alpha,\beta) \neq 0 \iff \alpha \neq \beta \tag{2.8}$$

$$V(\alpha,\beta) = V(\beta,\alpha) \geq 0 \tag{2.9}$$

$$V(\alpha,\beta) \leq V(\alpha,\gamma) + V(\gamma,\beta) \tag{2.10}$$

An important characteristic of these two algorithms is that each is guaranteed to find a labelling whose energy is within a distance from the global optima. Finally, if the smoothness term does not meet the conditions above, then the problem of finding the optimal cut can be done using QPBO [79]. However, this comes at the cost of leaving some nodes unlabelled [79], i.e. the algorithm was unable to find a label.

2.5.1.2 Message Passing

For the sake of simplicity we only describe the message passing equations for Loopy Belief Propagation (*LBP*), and the MRF is assumed to be 4-connected. In the rest of this section let p, q and s be three nodes in an MRF such that $s \leftrightarrow q \leftrightarrow p$. Let d_p be the possible label assignment for pixel position p, and let b_p be a vector that contains the min-marginals (beliefs) of node p for each possible label d_p.

Each component of b_p is computed as in eq.2.11, which is done at the end of T iterations.

$$b_p(d_p) = C_p(d_p) + \sum_{q \in N(p)} M_{qp}^{T}(d_p) \tag{2.11}$$

where $M_{qp}^{t}(d_p)$ is given by eq.2.12 (update equation). The objective is to find an optimal assignment d_q for each node p has label d_p. Here C_p and V_{pq} are used since we a dealing with stereo matching.

$$\begin{aligned} M_{qp}^{t}(d_p) = \arg\min_{\hat{M}_{qp}^{t}(d_p)} \Big\{ & C_q(d_q) + V_{pq}(d_p, d_q) \\ & + \sum_{s \in N(q) \setminus p} M_{sq}^{t-1}(d_q) \Big\} \end{aligned} \tag{2.12}$$

An important observation in eq.2.12 is that no order is imposed on how messages M_{sq}^{t-1} are used to update M_{qp}^{t} (at $t = 0$, all entries in M_{sq}^{0} are set to zero). Fig.2.12 shows how the message passing actually looks like from an implementation point of view.

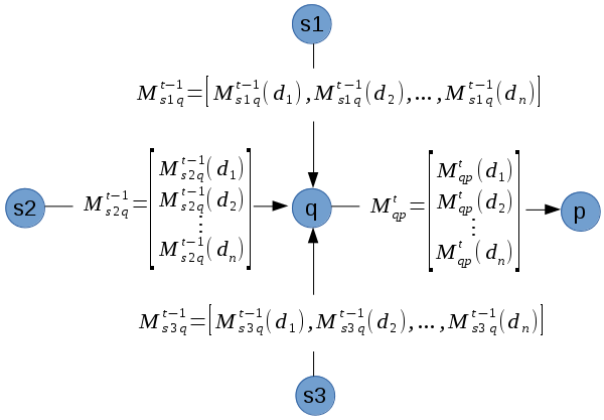


Figure 2.12: Message passing in a 4-connected graph.

Message passing is done iteratively for T times or until certain convergence criteria have been met [114, 21, 44]. The final assignment of label $\hat{D}_p$ is selected according to eq.2.13, which selects the label (disparity plane) with the lowest cost.

$$\hat{D}_p = \arg\min_{d_p} b_p(d_p) \tag{2.13}$$

The fact that *LBP* imposes no order in the message passing makes it prone to oscillation during energy minimisation [114], i.e. the label assignment can increase and decrease energy across multiple iterations, which is not a desirable feature for an optimisation algorithm. This problem is addressed by *TRW*-S, which imposes a sequential order for message passing (also done iteratively) by decomposing the graph into trees, see [44] for a full discussion on how messages are updated. Fig.2.13 shows how a 4-connected graph is decomposed into trees. Notice that the message passing is divided into two stages: forward and backward message passing, which ensures that each edge is covered exactly once. Once forward and backward passes have been completed they are followed by an averaging operation [44], which ensures that a node receives all messages from its neighbours. Reading the solution is however not as straightforward as in *LBP* and must be performed in a sequential way [44], e.g. following the forward pass convention.

The most important advantage of *TRW*-S over *LBP* is that energy is always guaranteed to decrease, and the label assignment is also guaranteed to have a lower energy when compared to both *LBP* and *GC*. See [44] for a full discussion on the optimality properties of *TRW*-S. Finally, note that the Semiglobal algorithm [34] is a special case of message passing [19] in which beliefs are computed along independent 1D paths (e.g. vertical, horizontal and diagonal) and then the beliefs are added up before

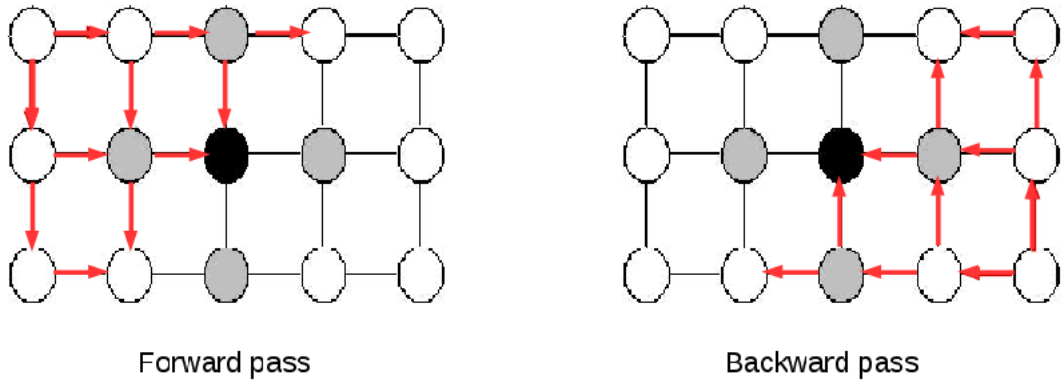


Figure 2.13: TRW-S message passing in a 4-connected graph.

selecting the disparity assignment.

2.6 Stereo matching pipeline overview

After the concepts of matching cost and optimisation techniques have been introduced it remains to explain how a complete stereo matching algorithm usually works. Fig.2.14 shows a typical stereo matching pipeline, where the computation of pixel cost and optimisation are the most important parts of a basic stereo matching algorithm.

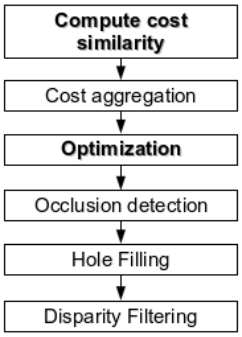


Figure 2.14: Typical stereo matching pipeline.

Once an optimised disparity plane assignment has been computed for both left and right images, it's followed by a post-processing stage (fig.2.14) usually consisting of:

1. Occlusion detection: Once the local stereo matching process creates left and right disparity maps, they are cross checked to detect occlusions, and invalid disparity plane assignments are removed.

2. Hole filling: Occlusions detection can leave holes, which may need to be filled in depending on what application is given to the disparity map.

3. Disparity filtering: The resulting disparity map can be noisy, and thus may need to be refined and filtered to improve its quality.

2.7 Disparity plane assignment evaluation

After a disparity plane assignment has been estimated and post-processed, the result is expected to be accurate, consistent and smooth. These three characteristics of the disparity plane assignment can be briefly defined as follows:

- Accuracy: Measures how close is the disparity map produced to the ground truth, which can be commonly accomplished by measuring the number of pixels whose disparity value difference is under some threshold, number of pixels whose disparity is wrong in occluded areas, root mean square of the original 3D positions and those obtained from the disparity map, or the average error.

- Consistency: Evaluates how close do the images induced by the disparity map approximate left and right views. Fig.2.15 shows two images from [90], where the image induced by the disparity map has regions (green circles) that do not exist in the original image, i.e. the regions are inconsistent.

- Smoothness: Evaluates how well does a disparity value fits a neighbourhood, which can be embedded in the optimisation algorithm. Additionally it can also measure the rate of change of disparity changes at a local scale. While having an accurate and consistent disparity map should be enough, it does not prevent the disparity map from changing the underlying shape of objects, e.g. fine details that become coarse or disappear.

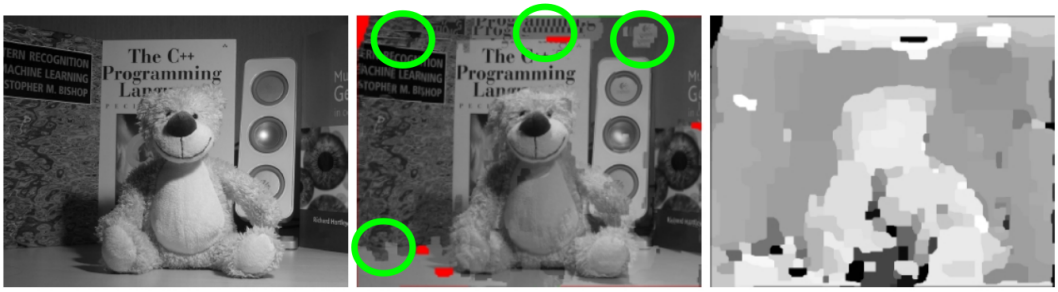


Figure 2.15: Inconsistencies in image induced by disparity map, left: reference image, centre: induced by disparity map (green circles show inconsistency), right: disparity map (source [90]).

2.7.1 Stereo data sets

The evaluation of the stereo matching algorithm accuracy is fundamental to validate the assumptions made and proposed improvements. In this thesis two types of scenes are used: indoor and outdoor environments. The reasons to use these types of scenes can be summarized as follows:

1. Fine details are better evaluated indoors (e.g. thin poles, sharp edges) due to very well controlled lighting conditions.
2. Realistic light conditions are better evaluated in outdoor environments.
3. The algorithm should be able to deliver similar results in both environments. This ensures that no over-fitting has occurred.

To evaluate these two types of scenes we use two very well known data sets:

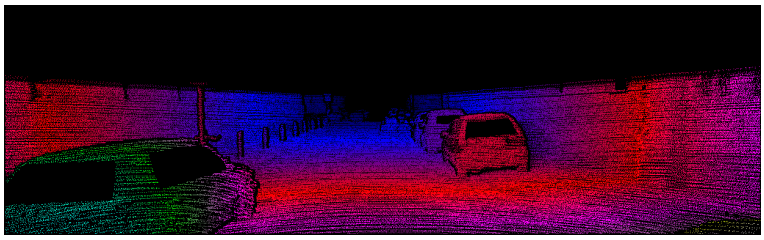
Middlebury: The Middlebury data sets version 1 and 2 (see [82]) were the de facto stereo matching benchmarks for many years. The main reason was the highly accurate groundtruth (static scenes captured using structured light), and nearly perfect light conditions. However, it was limited by a maximum precision of quarter of a pixel, and limited number of images. The version 3 [81] of this data set has addressed these issues by introducing true sub-pixel precision with up to 0.032 disparity residual error and included challenging light conditions. This data set consists of indoor images (see fig.2.16), which are rich in details making them ideal for testing the capability of a stereo matching algorithm to recover fine details. However, one disadvantage of the data set is the limited number of images in its benchmark, which could lead to over-fitting.

KITTI: The KITTI [24, 66] data sets have become the new de facto stereo matching benchmark for outdoor environments. The data sets consist of stereo images captured at 30 frames per second by a car driving through city streets. The groundtruth depth maps are obtained using a LIDAR (with up to 2 cm of error) and then mapped to disparity (see fig.2.17). Unlike the Middlebury data sets the light conditions and motion blur create challenging/realistic conditions that are necessary to evaluate the robustness of a stereo matching algorithm. However, the disparity groundtruth in this data set is sparse and only contains information for the lower half of the image, the disparity groundtruth is not perfectly aligned with the intensity images, and the scenes are only of cars and roads.

(a) table (b) groundtruth

Figure 2.16: Middlebury V3 “Table” image.

(a) image “15”



(b) groundtruth

Figure 2.17: KITTI 2012 image “15”.

These two data sets are complementary as they provide scenes with fine details (Middlebury) and challenging scene conditions (KITTI). Both types of scenes are needed to evaluate the stereo matching algorithm result’s accuracy, robustness, and consistency. For instance, an algorithm could be tuned to work well with indoor environments (e.g. Middlebury), but its performance in outdoor environments could be poor due to the algorithm’s inability to handle lighting changes (e.g. KITTI). The inverse case is also a cause of concern as an algorithm that is robust to challenging lighting conditions is prone to having low performance in indoor scenes with fine details (i.e. fine details could be confused with noise). The leader tables of the Middlebury and KITTI data sets show this problem, as algorithms that are ranked highly in the Middlebury table (in particular for V2) tend to have a low ranking in the KITTI ta-

ble. In some instances algorithms are ranked similarly in both data sets when using different parameters, which shows that these results do not generalise and are not consistent. It is important to note that the official evaluation metrics of both Middlebury V3 and KITTI data sets are the same, i.e. percentage of all/non-occluded bad pixels above a disparity error threshold, and average disparity error. The Middlebury data set also provides additional metrics such as root mean square error, and sub-pixel error metrics. The KITTI data set by contrast includes metrics for reflective areas (i.e. cars), and errors are only evaluated using an integer threshold.

2.8 Stereo data capture

The capture of stereo images, is very simple as it only requires two cameras separated by a baseline. The devices that can be used to capture stereo images can be diverse. However, the quality of the images will be different due to image resolution, quality of the sensor, noise, and calibration. The following list provides a brief overview of data capture alternatives:

Webcam: Any webcam can be mounted to create a low cost stereo rig (e.g. The Minoru webcam `www.minoru3d.com`). However, the image quality and framerate could be a limiting factor in the stereo matching results.

DSLR: DSLR cameras generally provide better quality than webcams which makes them ideal for stereo matching. However they are either limited to single image capture or low framerates, and thus better suited for static scenes (e.g. `www.di4d.com`).

PointGrey: The PointGrey (`www.ptgrey.com`) cameras provide a wide range of models suitable for real time applications. The KITTI data set was captured using this type of camera.

BumbleBee: The BumbleBee (`www.ptgrey.com`) cameras are a popular choice for real time applications as they come enclosed in a box containing the stereo rig, which is pre-calibrated. There are binocular and trinocular versions available.

MultiSense: The MultiSense (`http://carnegierobotics.com/`) cameras are a popular choice for for real time applications as they come enclosed in the same box and additionally have a hardware implementation of semi-global matching.

2.9 Stereo matching issues

The main components of the stereo matching problem have been described in the previous sections. Thus far it has been assumed that each pixel has a perfect match, and that the optimisation algorithms works perfectly. However, under real conditions the stereo matching algorithms commonly suffer from the following issues:

- Occlusions: There is no guarantee that all points in left/right images will have a match, or occlusions are correctly detected.
- Textureless/self-similar regions: These regions will have multiple matches, which creates ambiguity, e.g. a large surface with constant intensity produces multiple matches with the same matching cost.
- Fattening Effect: Window based methods suffer from errors near depth discontinuity where foreground objects contaminate the background (The larger the window the larger the effect).
- Intensity changes: Images captured with different cameras might have differences in the intensities.
- Subpixel accuracy: Matching pixels/blocks using integer disparities results in discretisation artifacts.
- Perspective distortions: Changes in perspective create distortion on the surfaces seen in left and right images.

Note that changing window size, shape and cost function helps to reduce some of the issues.

2.10 Summary

This chapter has described the basic principles, components and issues of a stereo matching algorithm that have to be understood before proposing a new stereo matching algorithm. In the rest of this thesis we propose improvements and develop solutions for some of stereo matching algorithm issues described previously.

Chapter 3

Literature review

The latest advances in stereo matching have primarily focused on algorithms that can estimate sub-pixel accurate disparity maps. The focus on this domain has come mostly from the limitation of stereo matching algorithms that only produce integer valued disparity maps, which can't handle thee following scenarios:

- Recovering depth from slanted surfaces.
- Recovering depth from curved surfaces.
- Efficiently explore a 3D search space.

Addressing these three problems not only has the advantage of obtaining a more accurate disparity map, but also enables new types of application, such as:

- Novel view rendering: using the recovered depth, it is possible render the scene/view from a new viewpoint.
- Unrectified stereo matching: using two images from different view points and estimating a 3D plane and transformation that produces an optimal 3D labelling.
- Scene flow estimation: using image pairs at time t and $t+1$ to estimate disparity maps and optical flow simultaneously.

New aggregation algorithms have been developed to reduce the computational time needed, as well as improved accuracy in the estimated disparity map.

In this chapter the most recent and relevant work is presented. In particular, three major areas are covered: Recent advances in aggregation algorithms, classification of 3D plane labelling algorithms, areas of improvement, and those not currently addressed.

3.1 Aggregation algorithms

So far the aggregation algorithm described in chapter 2 works using an $N \times N$ window. However, this has at least three major problems:

- Computational complexity is $O(n^2)$: This potentially makes its use impractical for large window sizes. The adaptive window used is non-separable.
- Window size is fixed: This implies that all pixels are treated equally, which may not necessarily be the case such as in small surfaces on thin objects.
- Aggregation is computed locally: This means that only the pixels in the adaptive window are used, but distant pixels that may belong to the same surface are not used at all.

Reducing the computational complexity of the aggregation process is of great importance for both real-time applications and practical points of view. There have been three approaches used to reduce the complexity of aggregation:

Recursive filtering: Aggregation can be interpreted as a bilateral filtering, for instance [109] develops an approximation using recursive Gaussian filtering. This has the advantage of completely removing the window size dependency from the computational complexity, and depending exclusively on the number of pixels in the image, which can potentially allow it to be used in real time applications. One important characteristic of this type of filtering is that it assumes that the range term can be expressed as a recurrent equation and that the filtering can be split along vertical and horizontal directions for later combination. The recursive Gaussian filter along the horizontal direction can be expressed as:

$$\hat{C}_p(d_p) = (1-\alpha)\cdot C_p(d_p) + R(p,p-1)\cdot\alpha\cdot\hat{C}_{p-1}(d_p) \tag{3.1}$$

where α is a recursion factor, $p-1$ is the previous neighbour (along the aggregation direction), which implies that eq.3.1 computes only the forward pass. To complete the recursive filtering a backward pass is done. The process to do vertical aggregation is then the same.

Separated aggregation: An obvious solution is to perform aggregation in separated directions, which reduces complexity to $O(n)$; for instance [48]. This process works by first aggregating the cost along a horizontal scanline and then

using the aggregated cost again to do vertical aggregation. It has to be noted that this type of aggregation is not the same as the joint bilateral filter from [112], but it produces similar results (see [48]), and performance at a reduced computational cost. Separated aggregation can be expressed as follows:

$$\hat{C}_p(D_p) = \left\{ \frac{1}{Z_v} \sum_{q \in N_v(p)} e^{-\frac{|I_p - I_q|}{\sigma_r}} C_q^h(D_p) \middle| Z_v = \sum_{q \in N_v(p)} e^{-\frac{|I_p - I_q|}{\sigma_r}} \right\} \tag{3.2}$$

$$C_q^h(D_p) = \left\{ \frac{1}{Z_h} \sum_{s \in N_h(q)} e^{-\frac{|I_s - I_q|}{\sigma_r}} C_s(D_p) \middle| Z_h = \sum_{s \in N_h(q)} e^{-\frac{|I_s - I_q|}{\sigma_r}} \right\} \tag{3.3}$$

Guided filtering: In [31] the problem of doing aggregation is stated as finding the optimal linear transformation that best approximates the results of bilateral filtering, while also removing the window size dependency from the computational complexity. This algorithm has also been used for other applications such as fast weighted median filter, and fast bilateral filtering. The assumption in [31] is that the output of a general bilateral like filter (i.e. guided filter) is represented in two different ways:

$$q_i = p_i - n_i \tag{3.4}$$

$$q_i = aI_i + b \tag{3.5}$$

where eq.3.4 assumes the output of the guided filter is modelled as removing noise n_i (or image guide) from the input pixel p_i, and this process can be approximated as the linear transformation in eq.3.5, with I being the reference image. Estimating the coefficients a and b is then stated as an optimisation problem:

$$(\hat{a}, \hat{b}) = \arg\min_{(a,b)} \sum_i (aI_i + b - p_i)^2 + \varepsilon a^2 \tag{3.6}$$

Then [31] shows that $\hat{a}$ and $\hat{b}$ can be computed as:

$$\hat{a} = \frac{cov_{n\times n}(I,p)}{var_{n\times n}(I) + \varepsilon} \tag{3.7}$$

$$\hat{b} = \overline{p_{n\times n}} - \hat{a}\overline{I_{n\times n}} \tag{3.8}$$

where p is the raw cost, and $(\overline{p_{n\times n}}, \overline{I_{n\times n}})$ are the raw cost mean and reference image mean computed in a $n \times n$ window. In a similar way $cov_{n\times n}(I,p)$ and $var_{n\times n}(I)$ are computed. The guided filter is easily extended for cost aggregation by making $q_i = \hat{C}_p(D_p)$ (aggregated cost), $p_i = C_p$ (raw cost), and ε acts like a smoothness weight. Finally, the $O(n^2)$ complexity of bilateral filtering is

removed by computing the covariance and mean using box filtering, which is rotationally asymmetric, but can be fixed by using a Gaussian filter[31].

From these three approaches the work from [31] best approximates the results of bilateral filtering, because aggregation is done in a $N \times N$ window rather than in different directions and then combined. Although these algorithms address the important issue of time complexity, they are still computed using a constant size local window, which should not necessarily apply to all of the image.

The idea of changing the window shape to the image intensity has two major variants:

Cross based aggregation: The aggregation algorithm of [119] estimates the size and shape of an adaptive window based on intensity/colour similarity. The window size estimation is divided into two stages, horizontal and vertical arm estimation, as seen in fig.3.1.

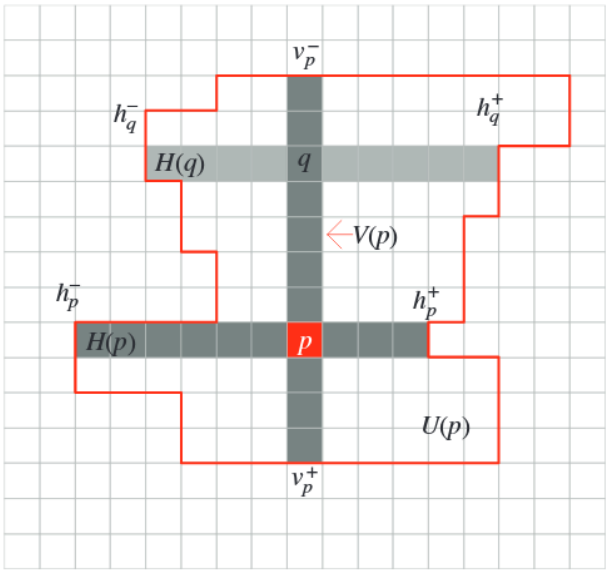


Figure 3.1: Adaptive window estimation, source[119].

In fig.3.1, p and q are two pixels, $H(p)$ is the horizontal neighbourhood, and $V(p)$ is the vertical neighbourhood. An important observation is that the size of the neighbourhoods are not regular, in fact the endpoints are given by $[h_p^-, h_p^+, v_p^-, v_p^-]$ and are computed by finding the last pixel whose intensity similarity is above a predefined threshold. Notice that $U(p)$ is the adaptive window (i.e. neighbour-

hood). The cost aggregation is then computed by:

$$\hat{C}_p(D_p) = \sum_{s \in U(p)} C_p(D_p) \tag{3.9}$$

In eq.3.9 *no adaptive weight* is used, which is done under the assumption that the adaptive window only includes the pixels from the same surface. This makes cross base aggregation dependent entirely on the initial estimation of the neighbourhoods $H(p)$ and $V(p)$. Cross based aggregation is done in two stages (horizontal and vertical) which are efficiently done using integral images[31]. This makes the computational complexity only dependent on the pixel in the image rather than the size of the window.

Segment based aggregation: The main idea of [43] is to do superpixel-segmentation (e.g. using [1]), and then do aggregation in each segment. This implicitly assumes that super-pixels are likely to have the same disparity, and thus the aggregated cost is given by:

$$\hat{C}_p(D_p) = \sum_{q \in Seg(p)} C_q(D_p) \tag{3.10}$$

In eq.3.10 p is the pixel and $Seg(p)$ is the segment to which it belongs. All pixels in a segment have the same cost, which at first may seem incorrect. However, segment aggregation was developed to deal with 3D labelling rather that just using fronto parallel planes, e.g. [24, 106, 66].

All the algorithms described up to this point have in common that aggregation is done within a window/neighbourhood, which can leave some pixels outside the cost aggregation. In an attempt to address this issue a new group of algorithms has been developed that are known as *Non-Local Aggregation Cost* algorithms (*NLAC*).

Non-Local Aggregation Cost: The first *NLAC* algorithm[108] computes the aggregated cost in a tree, which is done using the Minimum Spanning Tree (*MST*) of the reference image[1]. The image is then represented by a tree T, where each node v_p is a pixel. At each node v_p the aggregated cost that is later aggregated in the parent node is given as follows:

$$C_{v_p}^{A\uparrow}(D_p) = C_{v_p}(D_p) + \sum_{\phi(P(v_q)=v_p)} S(v_p, v_q) \cdot C_{v_q}^{A\uparrow}(D_p) \tag{3.11}$$

[1]The *MST* is normally computed using Kruskal's algorithm.

where $C_{v_p}(D_p)$ is the raw cost at v_p, $P(v_q)$ is parent node of v_q, and $\phi(P(v_q) = v_p)$ is the set of nodes for which v_p is the parent node. Notice that the recursive definition of eq.3.11 implies that the cost is aggregated starting at the leaves of T. In [108] using this fact it is shown that the aggregated cost at each node v_p is recovered as follows:

$$\hat{C}_{v_p}(D_p) = S(P(v_p), v_p) \cdot C^{A\uparrow}_{P(v_p)}(D_p) + [1 - S^2(P(v_p), v_p)] \cdot C^{A\uparrow}_{v_p}(D_p) \quad (3.12)$$

The computational complexity of *NLAC* algorithm is linear [108] in the number of nodes present in the MST. As is the case of window based aggregation, *NLAC* has been extended to use segments as in [64].

A common characteristic of all the aggregation algorithms reviewed thus far is that their efficient implementation depends entirely on the assumption that either all pixels belong to the same disparity plane, which limits their application to either doing fast aggregation on a large number of disparity planes, or working exclusively with super-pixels. Another problem with these algorithms is that many of them work under the assumption that the estimated window/neighbourhood or MST matches perfectly the underlying 3D surface, which may not necessarily be the case, and also may be prone to errors in cost aggregation due to errors during the initial segmentation.

3.2 3D labelling algorithms

In chapter 2 it was assumed that the 3D labelling was done using only fronto parallel planes at the cost of being unable to recover slanted or curved surfaces. However, great progress has been made in recent years to address these two issues. Most of the work has centred around assigning a 3D slanted plane that better represents slanted or curved surfaces, thus a plane D_p at pixel p has two parameters: a 3D unit normal vector $\hat{n}_p = (n^x_p, n^y_p, n^z_p)$ and disparity d_p. The disparity of pixel $q = (x_q, y_q)$ using D_p is given by:

$$D_p(q) = a * x_q + b * y_q + c \quad (3.13)$$

where $a = -\hat{n}^x_p/\hat{n}^z_p$, $b = -\hat{n}^y_p/\hat{n}^z_p$ and $c = (\hat{n}^x_p * x_q + \hat{n}^y_p * y_q + \hat{n}^z_p * d_p)/\hat{n}^z_p$ as in [9]. This new definition of the disparity plane can be easily used in eq.2.5 (from chapter 2).

Fig.3.2 shows the concept of using a 3D slanted plane. One consequence of using a 3D plane is that when evaluated within a window (left image of fig.3.2) it changes the shape of the corresponding window on the target image, i.e. right image in fig.3.2.

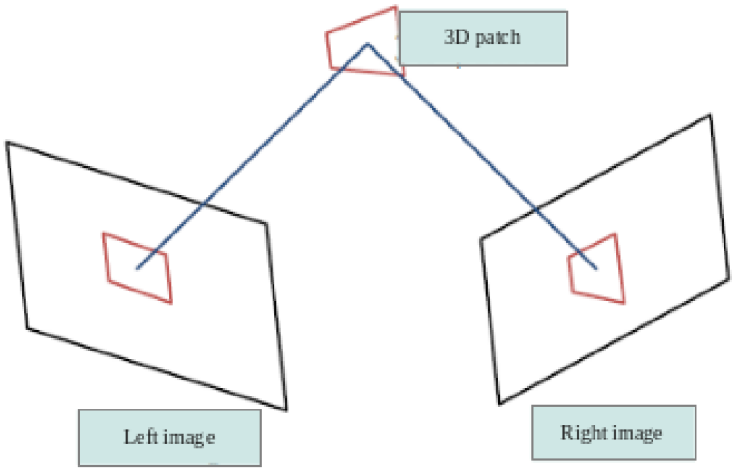


Figure 3.2: Slanted 3D plane.

It is clear that using a slanted 3D plane is far more complex than using a simple fronto parallel plane, since it requires solving two fundamental problems:

- ***Plane hypothesis***: A mechanism is needed to select a number of hypotheses to be evaluated by the optimisation algorithm.

- ***3D space exploration***: Eq.3.13 shows that the plane has 3 degrees of freedom and therefore requires a mechanism to traverse the large search space.

Depending on how these two problems are addressed, a 3D plane labelling algorithm is classified into fixed plane inference (FPI) and dynamic plane inference (DPI). These algorithms are better characterised as follows:

3.2.1 Fixed Plane Inference

FPI algorithms make an initial disparity estimation and then extract the planes, which are then used during the inference process to re-estimate disparity.

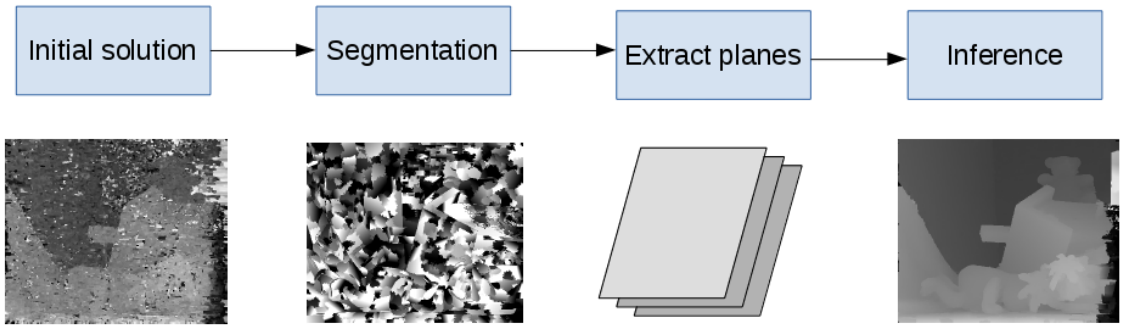


Figure 3.3: Common pipeline of FPI algorithms.

Fig.3.3 shows a typical FPI algorithm: estimate an initial disparity map, oversegment the reference image (e.g. fig.3.4 left), extract/fit planes from each image segment (e.g. using RANSAC), and finally use the extracted/fitted planes to do inference. There are two possible ways to organise the plane hypotheses, one is to evaluate all extracted planes for all pixels during inference, and the other is to extract several planes per image segment and only evaluate the planes within its segment during inference.

(a) reference image (b) over segmentation

Figure 3.4: Teddy image oversegmentation using SLIC.

Among FPI algorithms we can find [104] that uses a second order smoothness prior in a MRF. The most important contribution noted is the use of higher-order cliques in the MRF to model the smoothness and using the *QPBO* variant of the α-expansion *GC* algorithm to address the non-submodular nature of the proposed smoothness term. This algorithm was developed to perform novel viewpoint rendering, which required sub-pixel accurate disparity maps in order to produce high quality images.

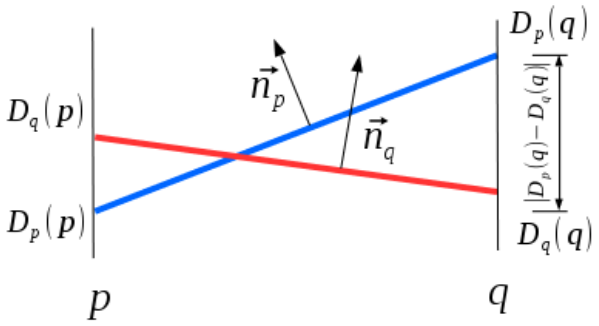


Figure 3.5: Smoothness term used that measures the curve's deviation from the tangent plane [73].

Due the possibility of *QPBO* returning unlabelled nodes, a solution is proposed in [73], which consists in using a variation of the $L1$ smoothness term (see. fig.3.5) that, according to [73], results in less non-submodular terms, and removes the need of using higher order cliques. However, the most relevant contribution is the use of TRW-S to do inference and proving that the smoothness term used can be computed in linear time by using the distance transform to compute the messages as in [21]. A common characteristic of both [73] and [104] is that they use a MRF where each node is a pixel.

Other relevant work includes [43] which follows the standard FPI pipeline, and is among the historically most important since it introduces the basic concepts of FPI algorithms, as well as making the contribution of representing the MRF using segments rather than pixels in an effort to handle textureless surfaces. One problem of doing inference using *GC*, *LBP* or TRW-S is the high computational time required either due to the smoothness term used or the number of iterations required to converge to a solution.

The computational time problem is addressed in [85] by grouping pixels (and finding a bounding box) from an initial disparity map to extract a set of planes, which are used to find the maximum and minimum disparity (for each plane in its corresponding bounding box). A local search range is sampled at regular real values using fronto-parallel planes and the optimal disparity plane is found using SGM, first per bounding box and then for all pixels by combining the different disparity plane assignments. Although [85] uses fronto parallel planes to do inference, it is still capable of recovering real valued disparity due to the smart search strategy used to sample the 3D space.

The FPI algorithms discussed thus far described all assume that the initial disparity estimate and extracted planes are correct, which may not necessarily be the case and therefore could make FPI susceptible to errors in initial segmentation, disparity esti-

mation, and plane extraction. Unlike [85], the FPI algorithm from [106] estimates a 3D plane labelling per image segment, computes occlusions, refines image segmentation, and reduces the computational time requirements by developing a block-coordinate descent algorithm. The seemingly intractable *NP* problem is approximated by alternating three steps: (1) jointly estimating occlusions and segments, (2) estimating segment boundaries, and (3) estimating/refining the initial plane assignments. Using segments instead of planes has the advantage of labelling several pixels at the same time.

FPI algorithms using disparity planes per segment have been successfully used to compute scene flow. For instance [96] jointly computes disparity, optical flow, and segmentation, which is accomplished by making the assumption that a 3D scene can be decomposed into 3D moving planes. Using 3D moving planes (see [96]) requires that each segment must be assigned a 3D plane, rotation, and translation (i.e. the 3D label has three components), which are first computed from an initial estimate of disparity, and optical flow to generate a number of hypotheses. Additional 3D moving planes are obtained from neighbouring segments. Once a number of hypotheses is estimated per segment, the optimal 3D plane is inferred using the QPBO variant of *GC*.

Although using planes allows the recovery of sub-pixel accurate disparity, there are problems inherent to stereo matching that are not necessarily eliminated by using 3D planes, for instance problems with textureless or reflective surfaces. The work from [28] addresses these issues by using 3D models (e.g. cars) composed of planes, and adding a constraint that prevents segments that belong to the same object from having disparity values that are not consistent with the underlying 3D models. Using 3D objects as a way to generate disparity plane hypotheses is achieved by doing a semantic segmentation of the image (see [28] for the specifics), which makes the performance of [28] dependent on the initial semantic segmentation, and a set of scene specific models.

3.2.2 Dynamic Plane Inference

DPI algorithms initialise each pixel with one or more planes, usually a random 3D plane hypothesis, and then propagate them based to neighbours/regions under the assumption that neighbours/regions are likely to have the same plane. Then, in a refinement stage, the planes are improved, thus the planes are dynamically updated. DPI algorithms commonly have the following structure:

1. Create an initial random solution (or set of solutions/hypotheses) D_p at each pixel p.

2. Propagate spatially using a propagation scheduler, and update D_p.

3. Propagate from views (i.e. evaluate left disparity planes in the right image. and update D_p.

4. Refine current estimate of D_p.

5. Repeat for several steps or convergence.

The random initialisation can be done by using a uniform distribution for disparity values, where the values must come from a defined search range, and the initial normal vector $\hat{n}$ is usually generated by sampling a unit sphere. DPI algorithms have developed strategies to perform a search in a 3D space despite being a continuous search space.

In [9] the foundations for DPI algorithms are first introduced, where the two main contributions are adapting the patch match algorithm [5] and developing a technique to do a 3D search.

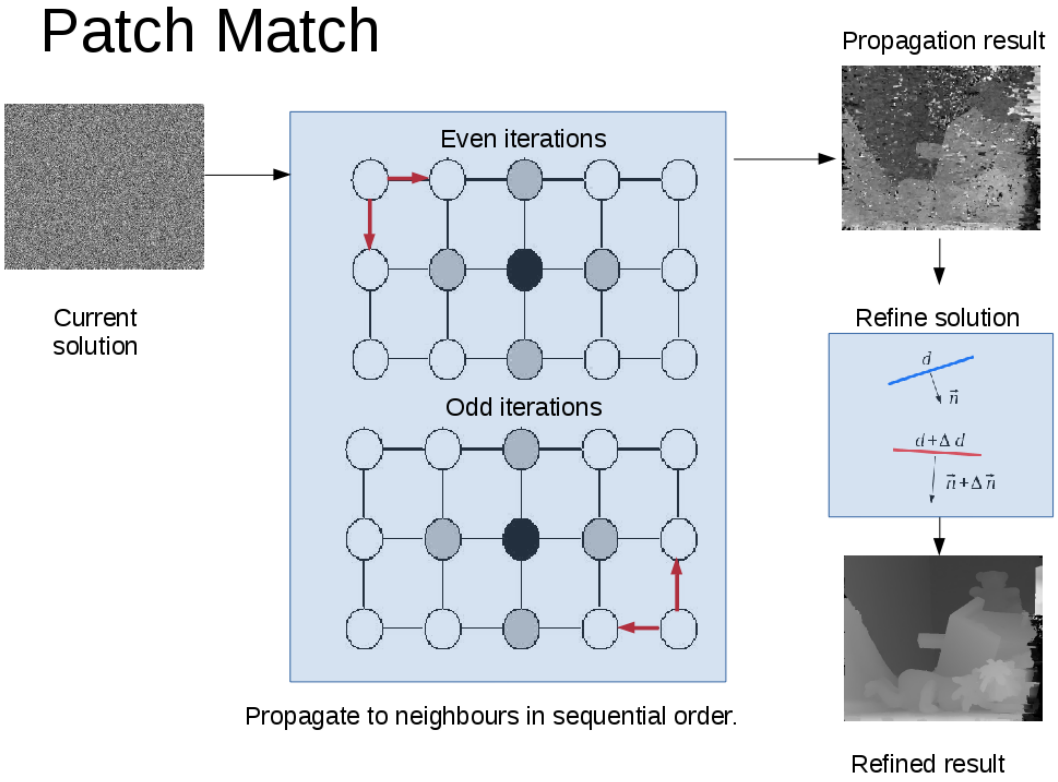


Figure 3.6: Patch Match stereo algorithm.

Fig.3.6 describes the general idea of [9] where the reference image is represented as a 4-connected MRF, and estimates the optimal assignment of a disparity plane per

node, i.e. pixel. The plane initialisation in [9] is done using a random plane per segment, which is then followed by a spatial propagation stage. The spatial propagation is done by first evaluating the pixel cost at a pixel p using its current disparity plane assignment and the planes from neighbouring pixels s and q (e.g. up and left planes). The disparity plane with the "best" score is kept. The propagation is done sequentially starting at the top left corner (i.e. forward raster scan order). The current disparity plane assignment is then refined locally by adding random variations. Additional views can be used to generate a new plane hypothesis by projecting the results of one view to another. i.e. view propagation [9]. The spatial propagation, refinement, and view propagation is done iteratively, where the propagation scheduler alternates its order, e.g. at odd iterations the spatial propagation is done in forward raster scan, and in even iterations is done in backward order scan. This change in propagation direction is done to cover all the MRF with trees [9].

The work from [9] does not impose any smoothness constraint on the resulting disparity plane assignment, i.e. it is a local optimisation algorithm. The algorithm in [6] (patch match belief propagation) addresses this issue by introducing a smoothness constraint in a 4-connected MRF. The main idea in [6] is to have multiple labels (or particles) per pixel and infer the optimal 3D label assignment using belief propagation. During the spatial propagation stage, it evaluates the cost of the current set of 3D planes at a pixel p and its neighbours s and q (as in [9]), which means that multiple 3D planes are propagated instead of only a single 3D plane. The refinement stage is done as in [9] for each of the 3D plane hypotheses per pixel. The algorithm from [6] also works iteratively following the same conventions from [9], but the resulting disparity maps are more accurate [6].

Both [9] and [6] have a sequential update algorithm, which, as shown in [88] (Locally Shared Label), is possible to approximate by using a hypothesis generator that can act like a propagation scheduler (fig.3.7 shows this basic idea).

In fig.3.7 each pixel is first initialised with a random disparity plane (the current hypothesis). Then each hypotheses is transformed into a series of disjoint disparity plane hypotheses (hypotheses generation) with each hypothesis covering an area of NxN pixels (see [88]). It is precisely this hypothesis generation process that allows [88] to propagate planes without the need of the sequential scheduler of [9, 6]. The way hypotheses are generated allows the use of a smoothness term that is submodular, which makes it possible to do inference using GC $\alpha-$expansion, and according to [88] the 3D labelling is then equivalent to solving multiple disjoint $\alpha-$expansions.

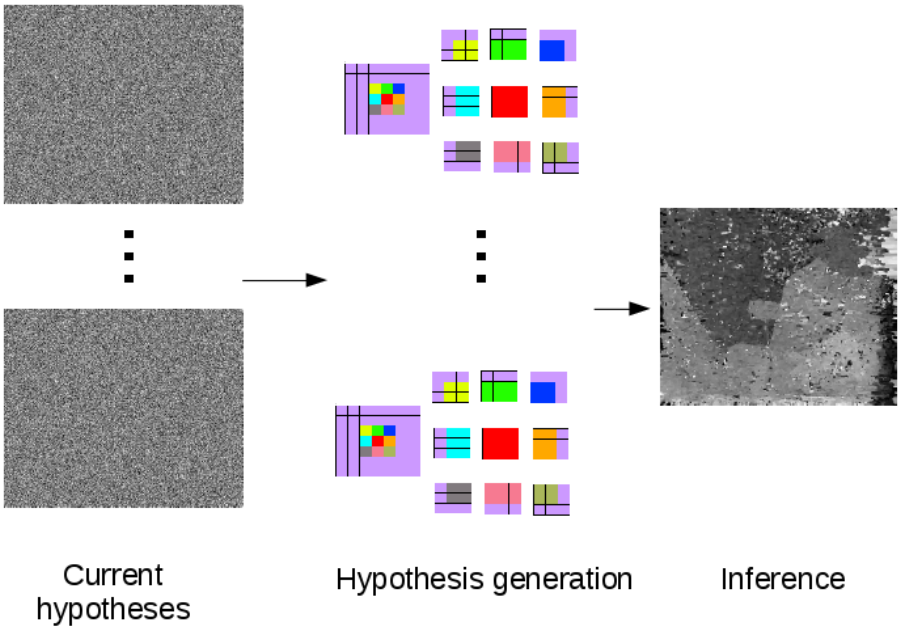


Figure 3.7: Locally shared label stereo algorithm. See text for explanation.

Using a smoothness constraint to regularise the disparity plane assignment requires the use of several hypotheses per pixel to prevent the optimisation algorithm from being trapped in a local minimum. In [32, 117] the problem of adding regularisation to [9] by only using a single plane hypotheses per plane is addressed by assuming that the smoothness term can be decoupled from the data term. The optimisation algorithm alternates between minimising the data term and smoothness term separately. The propagation and refinement stages are identical to that of [9] with the main difference being the introduction of a separated regularisation stage.

The DPI algorithms described up to this point work by assigning a 3D plane per pixel. This requires multiple evaluations of a pixel cost function that is expensive because of the intensive use of interpolation. The work in [55] reduces the compute time by using image segments (as in fig.3.8) and giving each segment a number of 3D disparity plane hypotheses, where the aggregation algorithm from [31] is used. Unlike [6], the propagation stage is done by randomly selecting a pixel (and its 3D labels) from each of its neighbouring segments, and then the best disparity planes are selected just as in [6], and then refined. Although segments are used in [55] the MRF is still 4-connected, and belief propagation is used for inference. The result of using a 4-connected MRF is that each pixel in a segment might be assigned a different plane, which is a desirable characteristic since it does not impose a hard constraint on all the pixels in a segment having the same plane, and thus avoids losing small details.

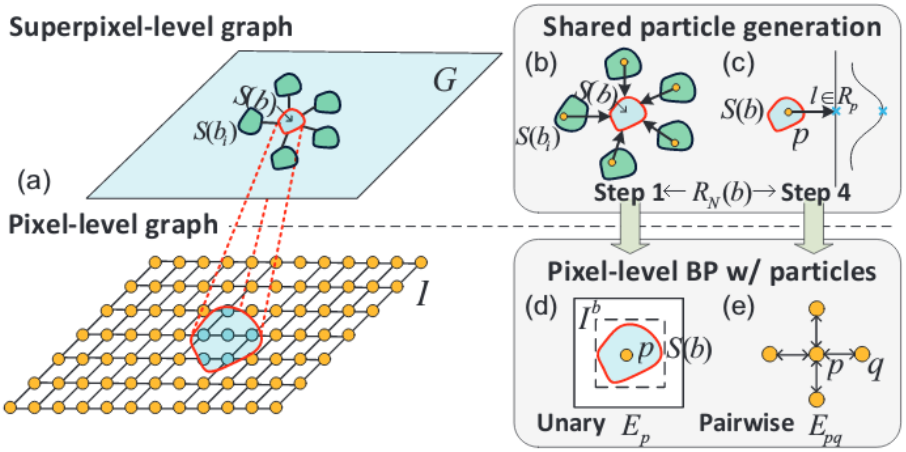


Figure 3.8: Segment based DPI [55]. Segments are overlaid over a 4-connected MRF to replace per pixel cost with segment cost.

3.2.3 FPI and DPI comparison

The main difference between DPI and FPI algorithms is that DPI algorithms have mechanism that allow them to propagate and refine the disparity plane assignments, which is ideal for overcoming errors in initial disparity map estimation. Additionally, DPI algorithms have proven to be efficient when traversing a continuous search space, whereas FPI algorithms are faster since they do not have the propagation and refinement stages. However, common issues in both DPI and FPI algorithms include the use of large adaptive windows (commonly using [112]) or segment [55] based similarity cost to compute the pixel cost function, which can result in a strong bias towards large planes. However, both FPI and DPI algorithms have become the state of the art for stereo matching (e.g. [96, 54, 75, 28]), since they can recover slanted and curved surfaces with great accuracy.

3.3 Disparity map regularisation

The assumption that the disparity plane assignment per pixel fits its neighbourhood is usually enforced using a prior known as the smoothness term. There are two basic smoothness terms commonly used (see [95]), which introduce a penalty V_{pq}:

- Potts model: interactions of a pixel and neighbours are piecewise constant.

$$V_{pq}(D_p, D_q) = \begin{cases} K & \text{if } D_p(p) \neq D_q(q) \\ 0 & \text{otherwise} \end{cases} \tag{3.14}$$

- Linear model: interactions of a pixel and neighbours are piecewise linear.

$$V_{pq}(D_p, D_q) = |D_p(p) - D_q(q)| \tag{3.15}$$

As shown in [95], both of these terms are submodular, which makes it possible to use $\alpha-$expansion. Furthermore, the linear and Potts models can be efficiently computed by message passing algorithms using the distance transform [21]. However, both eq.3.14 and eq.3.15 consider that all edges in the MRF have the same weight, which ignores intensity edges (in the reference image) that may align with depth edges. This new consideration is commonly addressed using the following models [21]:

- Weighted Potts model:

$$V_{pq}(D_p, D_q) = \begin{cases} U_{pq} & \text{if } D_p(p) \neq D_q(q) \\ 0 & \text{otherwise} \end{cases} \tag{3.16}$$

where

$$U_{pq} = \begin{cases} 3\lambda & \text{if } |I(p) - I(q)| < \tau \\ \lambda & \text{otherwise} \end{cases} \tag{3.17}$$

In eq.3.17 λ controls the influence of the smoothness term.

- Weighted linear model:

$$V_{pq}(D_p, D_q) = w_{pq}|D_p(p) - D_q(q)| \tag{3.18}$$

In eq.3.18 w_{pq} is a weight between nodes (e.g. image gradient). Note that eq.3.18 can still be computed using the distance transform [21], while eq.3.16 is still submodular and can be used with *GC* [45]. However, eq.3.18 has the disadvantage of the linear model not performing well at depth edges, which is addressed using:

$$V_{pq}(D_p, D_q) = w_{pq} min(|D_p(p) - D_q(q)|, K) \tag{3.19}$$

Eq.3.19 is known as the truncated linear model and is meant to model edges and smooth surfaces, and can also be efficiently computed using the distance transform [21], but it is no longer submodular, and to be used with *GC* it requires to use QPBO.

The smoothness terms described thus far were all meant to be used with integer valued disparities, and also are unable to model slanted surfaces. Modern 3D plane labelling algorithms have developed the following smoothness terms to handle slanted or curved surfaces:

- ***Normal vector regularisation***: In [6] the smoothness term is used to regularise the normal vectors.

$$V_{pq}(n_p, n_q) = w_{pq}(|n_p \cdot (x_q - x_p)| + |n_q \cdot (x_p - x_q)|) \tag{3.20}$$

  where x_q and x_p are 3D points. n_p and n_q are the normal vectors from disparity planes D_p and D_q.

- ***Linear model for 3D planes***: The work from [73] introduces a 3D label smoothness term that can be computed using a modified version of the distance transform of [21], but is not sub-modular.

$$V_{pq}(D_p, D_q) = w_{pq}|D_p(p) - D_q(p)| \tag{3.21}$$

- ***Symmetric linear model for 3D planes***: The contribution from [88] is a 3D label smoothness term that is submodular and thus can be used with $\alpha-$expansion.

$$V_{pq}(D_p, D_q) = w_{pq} min(|D_p(p) - D_q(p)| + |D_q(q) - D_p(q)|, K) \tag{3.22}$$

These three smoothness models are the most widely used for 3D plane labelling, e.g. [96, 6, 88, 73, 66, 55, 32, 117, 106].

3.4 Related stereo matching algorithms

The main topic of this thesis is the development of a DPI algorithm to estimate subpixel accurate disparity maps. However, it is worth mentioning that there are other stereo matching approaches that can also estimate sub-pixel disparity but using fronto-parallel planes. Among these approaches we can find:

Space carving: This approach is commonly used when there are multiple images available from different points of view. The basic idea consists in tessellating the 3D search space using voxels, and then iteratively leave the voxels (i.e. the photohull) that have a consistent projection across multiple images (i.e. they are both visible and minimise/maximise pixel cost function. This type of approach is indeed a generalization of stereo matching for multiple views. Some examples include [50, 49] where the algorithm presented works by finding the pixels that have the best matching cost, across multiple views as well as including a visibility constraint, but no smoothness assumptions are made about the scene. In [46]

the space carving approach is enhanced by using global optimization to improve the quality of the 3D reconstruction. However, [50],[49], and [46] produce discrete depth values due to the use of voxels. The work from [17] addresses the problem of discrete voxels by finding the 3D mesh that produces the lowest reprojection, pixel cost, and visibility errors, which is optimised using level sets. Using a mesh has the advantage of recovering normal vectors, and potentially using the mesh for interpolation.

Phase-based matching: An intuitive way of modelling the problem of stereo matching is to reformulate it as a doing signal matching in the frequency domain. This in practice means that potentially matching image blocks are compared using correlation in the frequency domain, which in practice means that the Fast Fourier Transform (FFT) is applied per image block (an expensive operation), but has the advantage of being able to estimate sub-pixel accuracy . In [80] the phase-based algorithm is initialised using the result of a local stereo matching method, and the initialisation is further refined by computing the optimal sub-pixel offset. The work from [67] and [16] address the issue of high computational cost by implementing the FFT and frequency domain matching in a *GPU* and $FPGA$, but [67] still uses an initialisation to compute the sub-pixel offset, whereas [16] does the whole matching process without prior initialization. Unlike the algorithms discussed thus far (i.e. local methods) the proposed solution from [23] includes a smoothness term to regularize the disparity map. In [105] wavelet transformations are used to develop an uncalibrated stereo matching algorithm that can work in multi-scale while obtaining sub-pixel accurate disparity maps. Note that all these phase-based algorithms have the implicit assumption of fronto-parallel planes and therefore suffer from the same problems.

Scale-space tracking: This technique consists in solving the matching problem using a scale-space pyramid where the matching is done first at a lower scale, and subsequently refined at a higher scale. The algorithm from [42] combines stereo matching with shape from shading and solves the correspondence problem using scale-space tracking. Improved results are obtained by reducing the ambiguity of shape from shading with stereo cues. The approach from [84] presents an active stereo matching algorithm that makes use of the scale-space approach to estimate disparity and then generate 3D models. The work from [103] describes a generalized mathematical model of scale-space tracking, its relation

to smoothness assumptions and optimization techniques to solve the matching problem. In particular [103] shows that the proposed space-scale tracking approach produces sub-pixel accurate disparity maps, because of the underlying assumption of deformable sheets used. In [14] the popular Multi-Scale Signal Matching algorithm is implemented in a modern CPU exploiting vector instructions to accelerate the computation of a disparity map for real-time applications. The work in [87] uses the scale-space approach for 3D reconstruction of retinal images. In [13] the scale-space approach is extended to take into account the estimation of optimal scale for matching, while in [41] novel window estimation approaches are developed to update the scale-space approach with modern aggregation techniques.

CNN-based matching: This approach consist of training a convolutional neural network (*CNN*) to evaluate the similarity of two potentially matching windows/patches. This approach was recently introduced by [97], where the training process consist in taking matching windows/patch, then applying some shearing and contrast changes to make the trained network robust to changes in perspective and light conditions. The output of the *CNN* is a descriptor that is passed to a fully connected neural network to compute the patch/window similarity. The disparity maps produced by [97] are top performing results (at the time of this thesis submission). The work in [116] presents a training methodology to compare windows/patches for large baseline stereo matching applications, which is accomplished by combining the output of several convolutional neural networks. In [60] improves the performance of [97] by penalizing the error in the disparity estimated rather that the patch similarity. All works mentioned so far compute the matching cost using *CNN*, which means that the *CNN* produces a cost that is cleaner and less ambiguous than traditional approaches, but still relies on cost aggregation and *MRF* inference to estimate a disparity map and handle textureless regions, occlusions or uniqueness assumptions.

3.5 Analysis and main areas of improvement

The common areas of improvement for both DPI and FPI algorithms can be categorised as follows:

1. ***Disparity refinement and outliers removal***: A mechanism to remove areas of

outliers in the disparity map is needed. This will improve the initialisation for disparity propagation and will allow a more accurate disparity estimate. In this same line of improvement, further refinement of a disparity estimate is needed in order to take into account distortions induced by the different perspectives of both images.

2. ***Disparity propagation in occluded areas***: A mechanism is needed to fill the areas where occlusions exist, but only if the surface is similar to nearby regions whose disparity is known. An important observation here is that another mechanism is also needed to prevent the propagation if the region is "different" enough from nearby areas where disparity is known. Note the term "similar/different" also implies the need to develop another mechanism to make such classification.

3. ***Disparity estimation in textureless/self similar regions***: This situation requires a method to detect the high incidence of ambiguities in textureless regions and select only the measures that are in areas that correspond to boundaries. A similar situation occurs with textures that are self similar, where a large number of ambiguities are commonly present. This also suggests that the same method used for textureless regions might also be used for self-similar regions.

4. ***Adaptive windows estimation***: Most of the work on adaptive windows has either focused on investigating how to assign weights or change the window shape. However the window size is either fixed or just a limit for the maximum size. It would be interesting to investigate how to combine cost functions based on image or underlying 3D content.

5. ***Adaptive smoothness term***: The smoothness terms from [6, 73, 88] all assume that either the variation in disparity or normal vector agreement should have the same penalty. The work from [45] is a classic example that shows how important is to give a different penalty to depth edges and smooth surfaces, which suggest that a similar approach could be extended to 3D plane labelling.

3.6 Summary and research direction

This chapter has covered the most relevant areas in modern stereo matching algorithms. In particular the DPI algorithms are currently of great interest mostly because of the following reasons:

1. ***3D search strategy***: DPI algorithms have proven efficient in traversing a continuous 3D space.

2. ***Robustness***: DPI algorithms are able to deal with noisy and imperfect initialisation.

3. ***Propagation scheduler***: DPI algorithms are able to exploit spatial neighbours to both propagate and traverse 3D space.

4. ***Sub-pixel accuracy***: DPI algorithms produce sub-pixel accurate disparity maps.

For these reasons, this research will focus on developing a new DPI algorithm that can match the state of the art performance while introducing techniques such as adapting the window size based on underling 3D surface, reduce errors in textureless areas, and generating disparity plane hypothesis that are not sequential. In addition to the areas of improvement of section 3.5, it is worth mentioning that the cost functions and smoothness term presented in this chapter suffer from the following problems:

- Large adaptive windows or segments: Cause loss of fine details.

- Strong plane bias: The assumption that all 3D surfaces in a scene are large planes may not apply for all pixels in the image.

- Update numerous hypotheses: DPI algorithms usually keep many hypotheses, which increases computational space and time requirements.

- Pixel centre bias when using large adaptive windows: Aggregation is prone to error resulting in incorrect labelling of a surface due wrong pixel adaptive weight computation.

- Higher order cliques used to model "unique" matches: Increase computation time and make algorithms more sequential.

Although, the primary goal of this thesis is to develop a new DPI algorithm, we will first develop and fronto-parallel based approach to handle these problems (chapter 4), and later extend it to become a DPI algorithm (chapter 5 and chapter 6).

Chapter 4

Using local cues to improve dense stereo matching

In this chapter we present a stereo matching algorithm that uses fronto-parallel planes, but is able to estimate sub-pixel accurate disparity maps with up to half a pixel of precision. The contributions presented are summarised as follows:

- Reducing edge overgrowth by using a V_{pq} that combines the Potts model with a truncated linear model (*TLM*).
- Reducing the fattening effect by using a function C_p that combines a pixel-wise cost with an aggregated cost.
- Maintaining horizontal slanted planes (e.g. floors) by including information on the surface direction as an additional data term.

These three contributions address the most common problems present in stereo matching algorithms that use fronto-parallel planes. We stress that the approach used in this chapter does not make use of sophisticated techniques such as smart normal vector space traversing [9] or 3D labelling [73],[88], yet our results still achieve top performance on integer disparity accuracy, while also obtaining competitive results on the half of a pixel accuracy benchmark. The results presented in this chapter were achieved during November 2014.

The concept of including local cues consists of locating possible depth edges and surface directions. These local cues are computed from image content and underlying 3D structure and allow our stereo matching algorithm to achieve high performance. The proposed algorithm will be referred to as stereo matching using Local Cues (*LCU*).

4.1 Stereo matching model

In this chapter the stereo matching algorithm described uses two rectified images (left I_l and right I_r views of a scene) and finds the approximate optimal correspondences for each pixel in image I_l to I_r. The optimal disparity[1] assignment is computed using global optimisation. The disparity maps are computed under the assumption that the 3D scene can be represented using fronto-parallel planes. The optimal disparity assignment is found minimising eq.4.1.

$$\hat{D} = \arg\min_{D} E(D) \tag{4.1}$$

where $E(D)$ is the cost of the disparity assignment (energy):

$$E(D) = \sum_{p}^{NumP} \{\alpha C_p(D_p) + \sum_{q \in N(p)} V_{pq}(D_p, D_q)\} \tag{4.2}$$

In eq.4.2, D is the disparity map aligned to a reference image (e.g. left). D_p encodes the relative displacement/disparity of the pixel at position p (e.g. in the left image) with respect to another image (e.g. right image). C_p is a function that measures the similarity/dissimilarity of two pixels (e.g. $I_l(p)$ and $I_r(p+D_p)$), and $NumP$ is the number of pixels in the image. $N(p)$ is a neighbourhood around p, q is a neighbour of p, V_{pq} (*smoothness* term) is a function that evaluates how well the disparity at position p compares to its neighbours, and α is a constant used to weight the importance of the similarity term. Eq.4.1 is typically minimised using either Graph Cuts (*GC*), Loopy Belief Propagation (*LBP*) or Tree Re-weighted with Sequential (TRW-S) update (see chapter 2 for further details). In this chapter TRW-S is used for minimisation unless otherwise explicitly stated. TRW-S is used as it has theoretical guarantees to find lower energy configurations than both *LBP* and *GC* [44].

Disparity could be a 2D displacement, but for the sake of simplicity we will assume throughout this chapter that images have been rectified, and disparity only happens in the horizontal direction and is integer valued (but the approach could be extended for sub-pixel disparities).

In this chapter, we will focus on the effect $C_p(D_p)$ and $V_{pq}(D_p, D_q)$ have on the disparity map obtained. Our experimental results will show that using a similarity function C_p that combines pixel-wise and block matching costs, modifying V_{pq} to use

[1] The displacement along a search area is known as disparity.

local information, and including information about surface direction, will produce the top integer disparity performance on the Middlebury benchmark[2] (version 2).

4.2 Basic model assumptions

The proposed approach follows advances in high performing stereo matching algorithms, which have focused on:

- Adaptive windows, e.g. [112, 29, 9, 65, 33, 39, 48, 107].
- An adaptive smoothness term for *LBP* and *GC*, e.g. [25, 98, 102, 10, 6, 74, 59, 118].

The use of adaptive windows is motivated by the assumption that neighbouring pixels (within a certain distance) with similar colour might have the same disparity value, and thus the cost might be similar as well. Adaptive windows produce disparity maps that have depth discontinuities that are sharp and are aligned mostly to areas where intensity discontinuities are in the reference image. The adaptive window used is [112]:

$$\hat{C}_p(d_p) = \sum_{q \in N(p)} S(p,q) R(p,q) C_q(d_p) \tag{4.3}$$

where:

$$S(p,q) = \left\{ \frac{1}{Z_s} e^{-\frac{dis_s(p,q)}{\sigma_s}} \mid Z_s = \sum_{q \in N(p)} e^{-\frac{dis_s(p,q)}{\sigma_s}} \right\} \tag{4.4}$$

$$R(p,q) = \left\{ \frac{1}{Z_r} e^{-\frac{dis_r(p,q)}{\sigma_r}} \mid Z_r = \sum_{q \in N(p)} e^{-\frac{dis_r(p,q)}{\sigma_r}} \right\} \tag{4.5}$$

where $C_p(d_p)$ and $C_q(d_p)$ is the similarity/dissimilarity function evaluated at the same disparity d_p and positions p and q, under the assumption that neighbouring pixels at the same disparity level and intensity colour might have similar costs. $S(p,q)$ is the spatial term (in the current chapter $S(p,q) = \frac{1}{n^2}$, n is the number of pixels in the window) and $dis_s(p,q)$ measures the distance (e.g. L_1) in the spatial domain. $R(p,q)$ is the image range term and $dis_r(p,q)$ measures the distance in the intensity/colour (L_1 distance in *RGB* colour space in the current chapter) domain.

The smoothness term used is assumed to be the common truncated linear model (TLM) ([89, 21]):

[2] and top average performance on version 3

$$V_{pq}(d_p, d_q) = \lambda w_{pq} \omega(d_p, d_q, K) \tag{4.6}$$

where λ controls the influence of the smoothness term, w_{pq} commonly depends on the image gradient (e.g. [51]), and $\omega(d_p, d_q, K) = min(|d_p - d_q|, K)$. The desired effect is that neighbours with similar intensities/colours have a greater contribution than those that are different. Eq.4.6 assumes a piece-wise smooth surface but not well defined edges.

4.3 Problems with fronto-parallel windows and TLM

In this section we discuss some of the problems present when using eq.4.3 as the similarity cost function, and using the TLM ($min(|d_p - d_q|, K)$ in eq.4.6) as the smoothness term in a global optimisation framework. We also analyse the effect that each of these parts has in eq.4.1, since eq.4.3 and eq.4.6 are the approaches most commonly used in the literature (e.g. [95], [89], [21]). The main issues addressed in the current chapter are in three categories:

- Fattening effect (FE) due to the use of windows/aggregation [82].
- Overgrowth effect (OG) due to the use of TLM and a pixel-wise function.
- Bias towards fronto-parallel surfaces.

(a) venus image (b) venus disparity

Figure 4.1: Venus colour image and disparity map.

When using message passing algorithms to minimise eq.4.1 it is common to use the TLM (e.g. [21, 111, 76, 62, 12]) to estimate smooth surfaces, but at the expense of

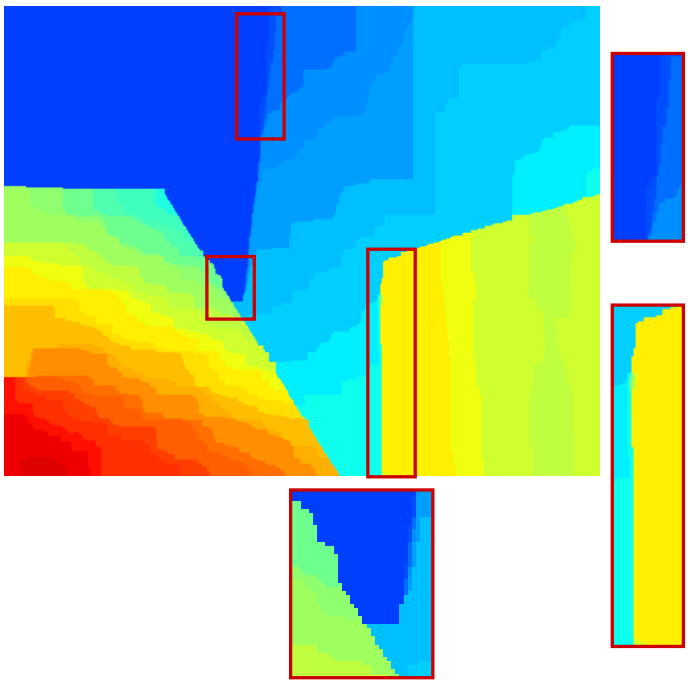


Figure 4.2: Venus image [82]. Red boxes show where disparity is overgrown. Parameters used $\alpha = 30$, $\lambda = 4$ and $K = 4$. Disparity values displayed using false colour for visualisation purposes.

slightly overgrown edges. Fig.4.1 shows the venus image and corresponding disparity map while fig.4.2 shows the noticeable effects of overgrown edges.

Fig.4.2 was obtained using TRW-S as the global optimisation algorithm (left and right images were up-scaled before computing the cost to obtain up to half of a pixel accuracy), the pixel similarity cost ($C_p(d_p)$) used is Birchfield [8] over RGB (pixel-wise cost), weights are computed using eq.4.7 (similar to [51]):

$$w_{pq} = \frac{1}{Z_n} e^{-|I_{ms}(p) - I_{ms}(q)|} \mid Z_n = \sum_{q \in N(p)} e^{-|I_{ms}(p) - I_{ms}(q)|} \tag{4.7}$$

where I_{ms} is a grey level image after applying the quick shift segmentation [94, 93] (eq.4.7 will be used for the weights w_{pq} in the rest of the chapter). Notice how the disparity map is smooth, but near depth discontinuities the surface is slightly overgrown. By contrast fig.4.3 shows the result of using the Potts model as the smoothness term, which shows that edges are sharper. However, the disparity map now looks piece-wise constant and noise patches are present.

One possible way to overcome some of the issues seen in fig.4.2 and fig.4.3 is to improve the raw pixel cost by doing aggregation [82] using the reference image (e.g. left) as a guide. This is done using eq.4.3, which is known to perform well on smooth

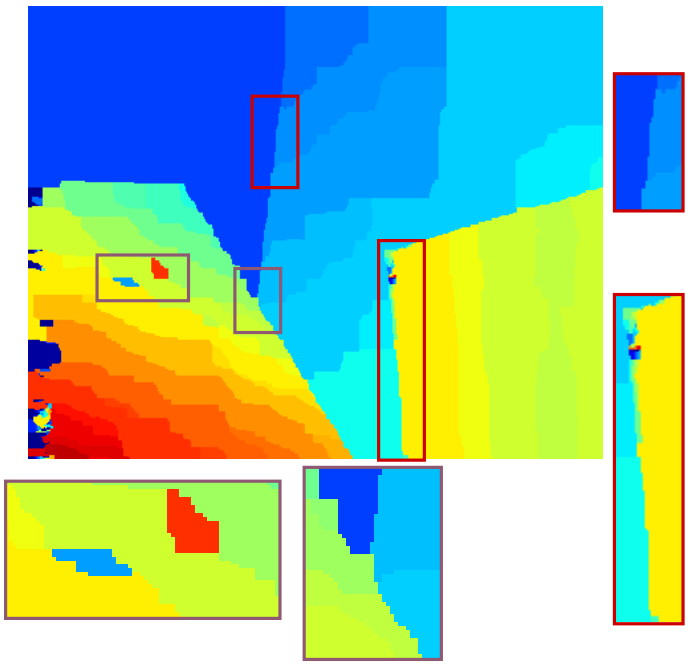


Figure 4.3: Venus image. Red boxes show where disparity over growth is reduced. The magenta boxes shows where disparity is no longer smooth or still overgrown. Parameters used $\alpha = 30$, $\lambda = 4$ and $K = 1$.

surfaces and near to boundaries (e.g. [112], [9]). Fig.4.4 shows the result of using eq.4.3 with a 41×41 window (large enough to reduce noise in the raw cost); it can be seen that overgrowth (OG) is reduced in certain areas (red boxes) while some new artifacts appear (magenta boxes). This error is known as the fattening effect (FE) [82]. We will refer to OG[3] as the problem caused as result of using the TLM with a pixel-wise cost and FE as the problem caused by computing the similarity/dissimilarity cost using windows/aggregation. Another unwanted effect of eq.4.3 is the introduction of a strong bias towards fronto-parallel surfaces. In fig.4.5 (right), it can be seen how the slanted plane (magenta box) has been erroneously subdivided into non-smooth fronto-parallel subregions.

[3]This is commonly referred to as over-smoothing, but we are mostly interested in its effect at depth discontinuities.

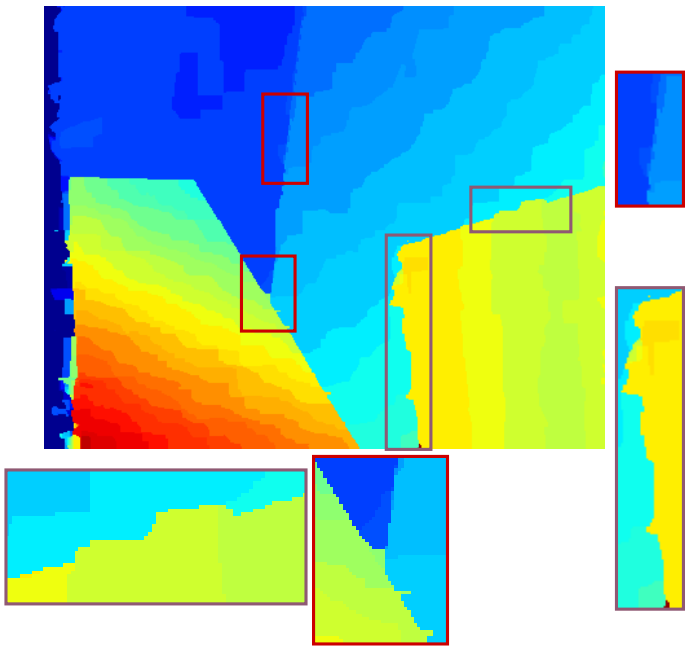


Figure 4.4: Venus image. Red boxes show where disparity overgrowth has been reduced by using eq.4.3. The magenta boxes show the fattening effect (FE). Parameters used $\alpha = 30$, $\lambda = 4$ and $K = 4$.

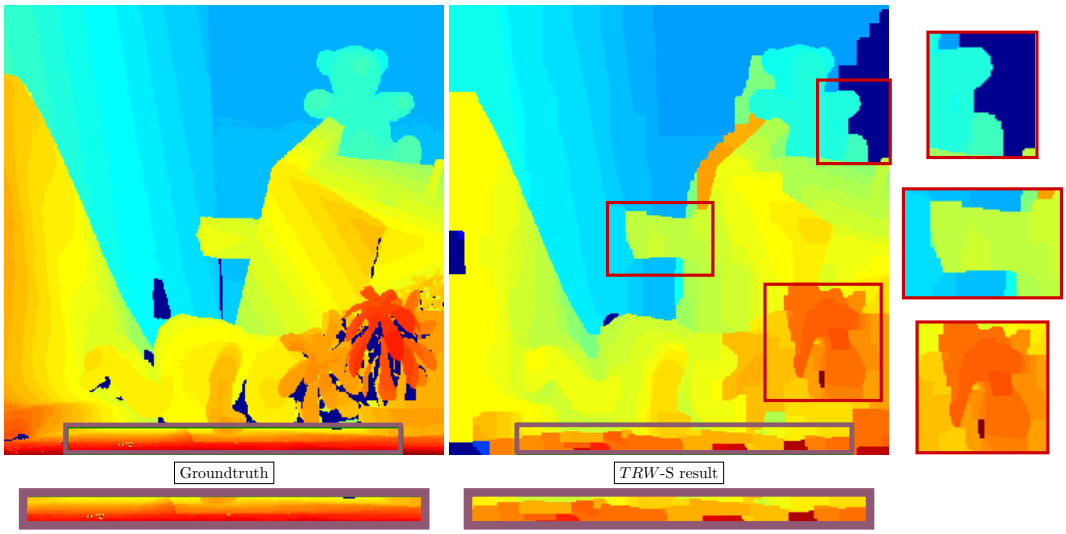


Figure 4.5: Teddy image [82] (see fig.3.4) using eq.4.3. Left magenta box groundtruth, right magenta box shows the slanted plane is flattened. Red boxes show the effects of the FE. Parameters used $\alpha = 45$, $\lambda = 2$ and $K = 4$.

4.4 Improving the stereo matching model

The previous sections have described what issues are commonly found in stereo matching algorithms using fronto-parallel windows. To address these problems we have improved the model from eq.4.2 using the following new ideas:

- **Improved edge model**: The proposed edge model uses image content to adjust the truncation constant in the TLM such that it adapts better to potential depth discontinuities.
- **Pixelwise and block matching cost**: The proposed cost combines a pixelwise cost and block matching cost (aggregated using eq.4.3), sec.4.5.4.
- **Computation of local cues**: We introduce the concept of local cues to address the issue of slanted surfaces along the epipolar line (sec.4.5.2 and sec.4.5.3).

The resulting disparity map after using our innovations is further improved by doing post-processing:

- Occlusion detection (sec.4.7.1).
- Hole filling (sec.4.7.1).
- Disparity filtering (sec.4.7.1).

4.4.1 Combined Potts Linear edge model

As noted previously, the smoothness term (referred to from now on as the *edge model*) using the TLM creates problems near boundaries when $K > 1$, and problems on smooth surfaces when $K = 1$. It would be desirable to retain the good properties of the truncated linear edge model (visible when $K = 1$ at edges, $K > 1$ on smooth surfaces) while reducing the side effects. The edge model that combines the Potts ($K = 1$) and truncated linear edge model ($K > 1$) will be referred to as the Combined Potts Linear edge model (CPL), which can be expressed as follows:

$$V_{pq}(d_p, d_q) = \begin{cases} w_{pq}\lambda\omega(d_p, d_q, K) & : |I(p) - I(q)| < \tau \\ w_{pq}\lambda\omega(d_p, d_q, 1) & : \text{otherwise} \end{cases} \tag{4.8}$$

where w_{pq} and λ are defined in eq.4.6. This model resembles that from [45] and [34], but the main difference is that we encourage smooth linear variations on surfaces with

similar intensities, and force evaluation of the Potts model at possible edges. We will see that this idea reduces the *OG*. One disadvantage of using the original image directly is that it can contain areas where the gradient is very noisy, which can potentially introduce changes in disparity that do not really exist.

4.4.2 Improved similarity cost

One consequence of the FE is the loss of detail (see fig.4.5) caused by a block matching cost [82]. To handle this issue we modify eq.4.3 to include a pixel-wise cost, which gives eq.4.9.

$$\hat{C}_p(d_p) = \alpha C_p^1(d_p) + \sum_{q \in N(p)} S(p,q)R(p,q)C_q^2(d_p) \tag{4.9}$$

where:

- $\hat{C}_p(d_p)$ replaces $\alpha C_p(d_p)$ in eq.4.2, $C_p(d_p)$ from eq.2.11 and eq.2.12 (see chapter 2).
- $C_p^1(d_p)$ is a pixel-wise similarity measure (e.g. Birchfield pixel wise cost).
- $C_p^2(d_p)$ can be any other matching measure (e.g. Normalised cross correlation).
- α regulates how much influence the pixel-wise cost has.
- $S(p,q)$ and $R(p,q)$ are defined in eq.4.4 and eq.4.5.

4.5 Computing local cues

Locating a possible depth discontinuity using the image gradient can lead to problems in areas where the gradient is noisy (e.g. a high frequency texture). The second issue with the current model is that there is no way to handle slanted surfaces. To overcome these two shortcomings, the following two local cues are included:

- Possible depth discontinuities, computed using a function similar to eq.4.3, but applied to the reference image, and its gradient.

- A data term providing information based on the epipolar line direction (e.g. horizontal), which is computed using independent message passing in horizontal and vertical directions.

4.5.1 Image and gradient filtering

The purpose of filtering the reference image and gradient is two-fold. First, the grey level of the reference image is filtered to remove noise in the gradient estimation (eq.4.10). Second, eq.4.11 is applied to the gradient magnitude to remove false edges (at least potentially false edges) so that eq.4.8 can handle high frequency textures (at least to some degree).

The adaptively smoothed grey level image is given by:

$$\hat{I}_g(p) = \sum_{q \in N(p)} S(p,q) R(p,q) I_g(q) \tag{4.10}$$

The adaptively filtered gradient magnitude is given by:

$$F_p = \sum_{q \in N(p)} S(p,q) R(p,q) |\nabla \hat{I}_g(q)| \tag{4.11}$$

where:

- $|\nabla \hat{I}_g(q)| = |\nabla_x \hat{I}_g(q)| + |\nabla_y \hat{I}_g(q)|$
- $\nabla_x \hat{I}_g(q)$ and $\nabla_y \hat{I}_g(q)$ are obtained using the Sobel gradient operator.
- $S(p,q) = \frac{1}{n^2}$ for a n×n window and $R(p,q)$ is given by eq.4.3.

4.5.2 Independent Horizontal and Vertical Message passing (*IHVMp*)

We now consider how to improve the recovery of slanted surfaces that are parallel to the epipolar direction (eg. ground planes). A cue to provide information about this direction can be obtained after approximately optimising eq.4.1 using IHVMp (*BP* on each row and column), where only one message pass round is needed since independent horizontal and vertical paths form a tree, in other words eq.2.11 is modified as follows:

$$b_p^{hv}(d_p) = \hat{C}_p(d_p) + \sum_{q \in N_v(p)} M_{qp}^{v,1}(d_p) + \sum_{q \in N_h(p)} M_{qp}^{h,1}(d_p) \tag{4.12}$$

where $\hat{C}_p(d)$ comes from eq.4.9, $M_{qp}^{v,1}(d_p)$ is defined as $M_{qp}(d_p)$ (eq.2.12) using vertical neighbours $N_v(p)$ and $M_{qp}^{h,1}(d_p)$ uses horizontal neighbours $N_h(p)$. Doing IHVMp is similar to Semi Global Matching (SGM, [34]), but the main difference resides in the smoothness term. Since the message passing is independent, we consider the result coming from eq.4.12 a local cue.

For the horizontal and vertical directions the smoothness term is V_{pq} in eq.2.12 is replaced by eq.4.13 and eq.4.14. In eq.4.13 a constant is used instead of a data dependent weight to give more importance to the horizontal direction, which is done under the assumption that there might be slanted surfaces in the horizontal direction.

$$V^h_{pq}(d_p,d_q) = 0.25\lambda U^{dir}_{pq}(d_p,d_q) \tag{4.13}$$

$$V^v_{pq}(d_p,d_q) = w_{pq}\lambda U^{dir}_{pq}(d_p,d_q) \tag{4.14}$$

where:

$$U^{dir}_{pq}(d_p,d_q) = \begin{cases} \omega(d_p,d_q,K_h) & : |F_p - F_q| < \tau_{hv} \\ \omega(d_p,d_q,K_l) & : \text{otherwise} \end{cases} \tag{4.15}$$

where $K_h > K_l > 0$, and F_p comes from eq.4.11. This allows larger disparity changes in areas where the intensity image gradient is similar (i.e. smooth surfaces), and small variations at possible depth discontinuities (i.e. sharp edges).

4.5.3 Integrating local cues

Now that we have the local cues that will provide information to handle noisy images/textures and provide some information on slanted surfaces, they are included in our approach. The first step is to update eq.4.8, which then becomes:

$$V_{pq}(d_p,d_q) = \begin{cases} w_{pq}\lambda\omega(d_p,d_q,K) & : |F_p - F_q| < \tau_{cpl} \\ w_{pq}\lambda\omega(d_p,d_q,1) & : \text{otherwise} \end{cases} \tag{4.16}$$

where $K > 1$. To include the information on slanted surface in the horizontal direction eq.4.9 is updated as follows:

$$\begin{aligned} \hat{C}_p(d_p) = \alpha C^1_p(d_p) + \frac{b^{hv}_p(d_p) - \min_{r,d_r}(b^{hv}_r(d_r))}{\max_{r,d_r}(b^{hv}_r(d_r)) - \min_{r,d_r}(b^{hv}_r(d_r))} \\ + \sum_{q\in N(p)} S(p,q)R(p,q)C^2_q(d_p) \end{aligned} \tag{4.17}$$

where $b^{hv}_p(d_p)$ is normalised to prevent it from having too much influence on the data term.

4.5.4 Similarity cost

The last remaining part is to select a similarity cost function for use in eq.4.9. The proposed cost function is meant to ***recover fine details***, ***provide information on small textureless areas***, and ***deal with small radiometric differences***. The following three different costs have been selected:

- Birchfield (Bt) pixel-wise cost function on *RGB* (as in [45]) or intensity gradient space, ***recovers the fine details***.
- Enhanced normalised Cross Correlation (ENCC)[78] on the grey level image (using a 3×3 window), ***provides information on small textureless areas***).
- Census transform with Hamming distance (Ct_p)[115] on the grey level image (using a 9×9 window), ***deals with small radiometric differences in the images***).

The census transform cost is computed as follows:

$$\hat{Ct}_p(d) = \min\{Ct_p(d-0.5), Ct_p(d), Ct_p(d+0.5)\} \tag{4.18}$$

The window is shifted by half of pixel to the left and right to reduce the FE of using a window. Each of these costs is meant to provide information at a different scale. Now that the similarity costs are defined, $C_p^1(d)$ and $C_p^2(d)$ (from eq.4.9) are defined by eq.4.19 and eq.4.20:

$$C_p^1(d) = \begin{cases} \alpha Bt_p(d) + ENCC_p(d) \text{ for IHVMp in eq.4.9} \\ \alpha Bt_p(d) \text{ for } TRW\text{-S in eq.4.9} \end{cases} \tag{4.19}$$

$$C_p^2(d) = \hat{Ct}_p(d) \tag{4.20}$$

4.6 Improvements to the cost function and CPL

The previous sections proposed modifications to improve the stereo matching model from eq.4.2. The introduced *CPL* edge model enables recovery of both sharp edges and smooth surfaces as shown in fig.4.6 where the improved structure boundaries (red boxes) is observable when compared to fig.4.5. The second contribution is the combination of pixelwise and block matching to reduce error in textureless areas. As shown fig.4.7 the almost textureless wall has the expected disparity, whereas the same area in fig.4.6 the upper red box has a low and incorrect disparity value.

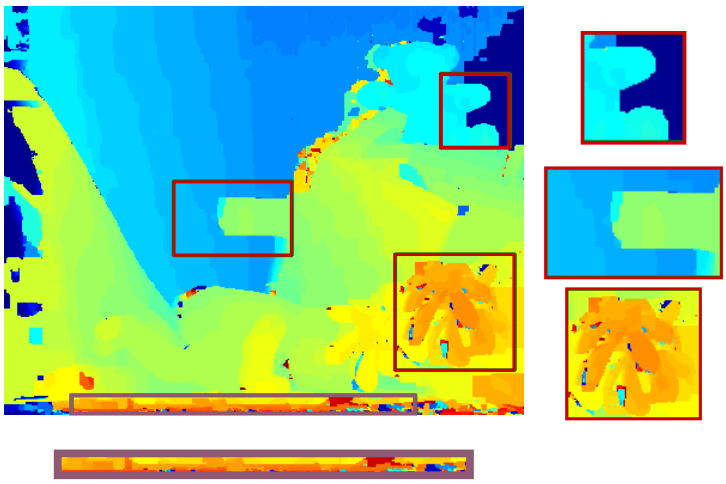


Figure 4.6: Teddy disparity map using CPL and pixelwise cost. Parameters used $\alpha = 45$, $\lambda = 2$, $K = 4$, window size $= 41 \times 41$.

The result in fig.4.7 uses the classic TLM edges with the similarity cost from sec.4.5.4, and this results in overgrown edges (especially around the teddy bear) and loss of large portions of the ground plane.

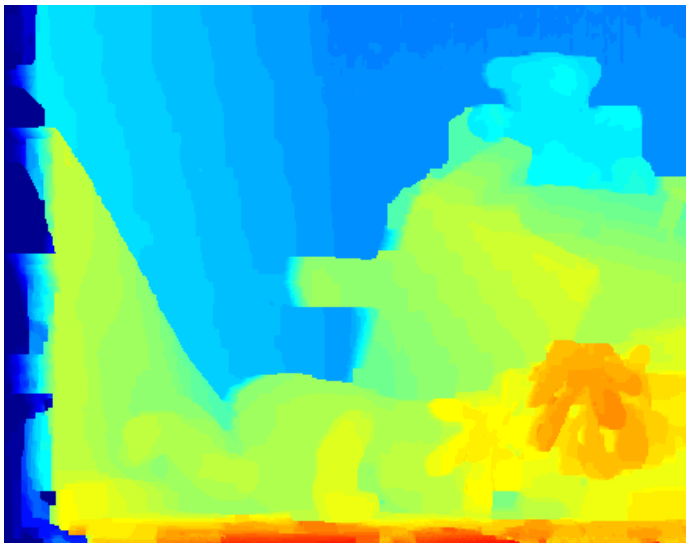


Figure 4.7: Teddy disparity map using TLM.

Some of the TLM issues are reduced in fig.4.8 by using the CPL edge model and cost function the resulting disparity map has sharper, better defined edges, and smooth surfaces. The CPL edge model so far has been able to recover sharp edges, keep smooth surfaces, and handle textureless regions. The proposed CPL edges and cost function are still unable to properly recover the slanted ground floor (i.e. slanted surfaces) as

seen in fig.4.8.

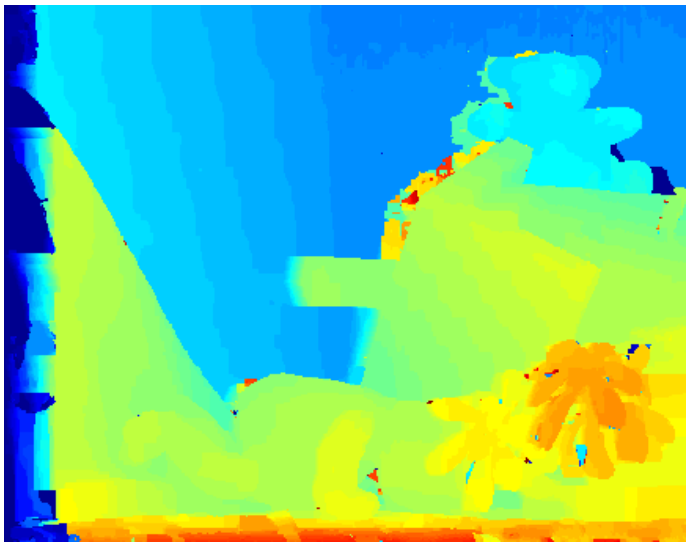


Figure 4.8: Teddy disparity map using CPL.

The proposed approach introduced the concept of computing local cues for slanted surfaces (parallel to the epipolar lines) by using $IHVMp$. The result of including this cues is shown in fig.4.9, which now has sharp edges, smooth surfaces, and keeps slanted surfaces. It is important to note that all examples in the previous figures show the left disparity maps which have ***NOT*** been post-processed and thus show artifacts such as regions of disparity with high either variance or low disparities in ***occluded*** regions, which is more noticeable when using the CPL model.

Fig.4.10 shows the final result of our algorithm after occlusion detection and post processing (see sec.4.7.1 for further details). The result in fig.4.10 still has marked quantisation artifacts (e.g. the stair case effect on the background wall) caused by the use of fronto-parallel planes to compute the matching cost.

The examples using the Teddy and Venus images provide qualitative evidence of our algorithm performance. Numerical evaluation is now performed to determine the quantitative improvement of the proposed algorithm $CPL+IHVMp$.

Tab.4.1, shows the percentage of wrong pixels (integer accuracy) on the standard 23 Middlebury images before doing any hole filling and median filtering, i.e. using the ***raw disparity map***, errors are evaluated only on non-occluded areas. Notice that CPL provides better results over using each cost alone with a fixed K. The column **pw+TLM** evaluates the simple pixelwise RGB Birchfield cost with the TLM in the edges. The column **agg+TLM** evaluates the census transform+Hamming dis-

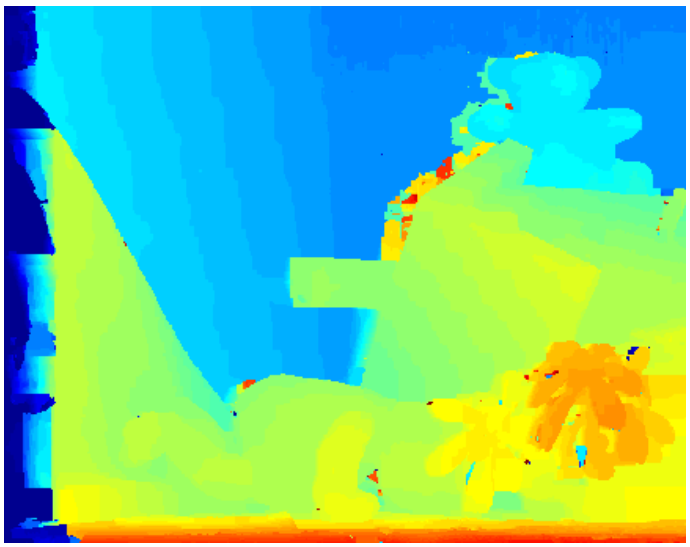
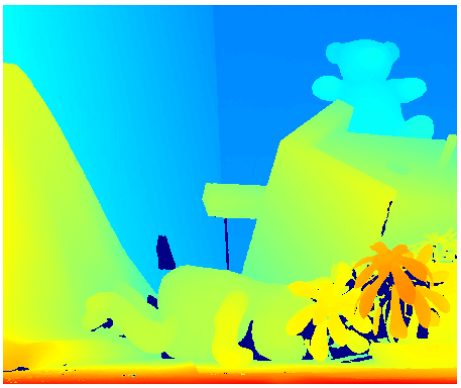
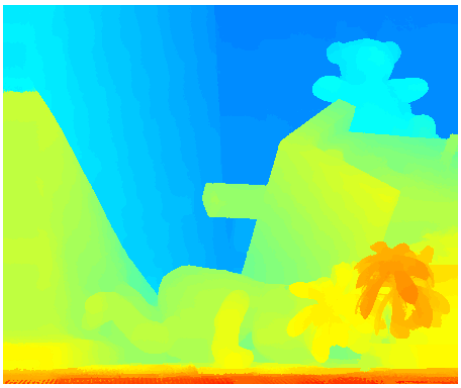


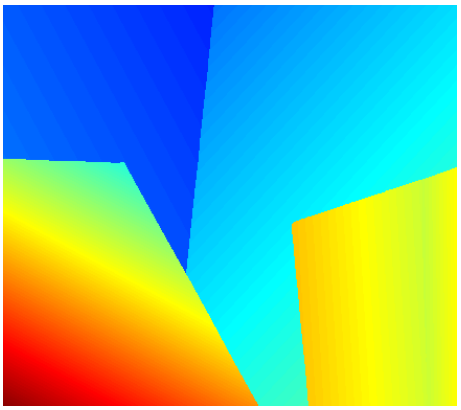
Figure 4.9: Teddy disparity map using $CPL + IHVMp$.



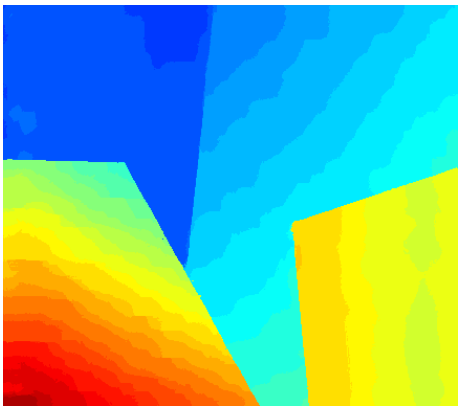
(a) teddy groundtruth



(b) teddy result



(c) venus groundtruth



(d) venus result

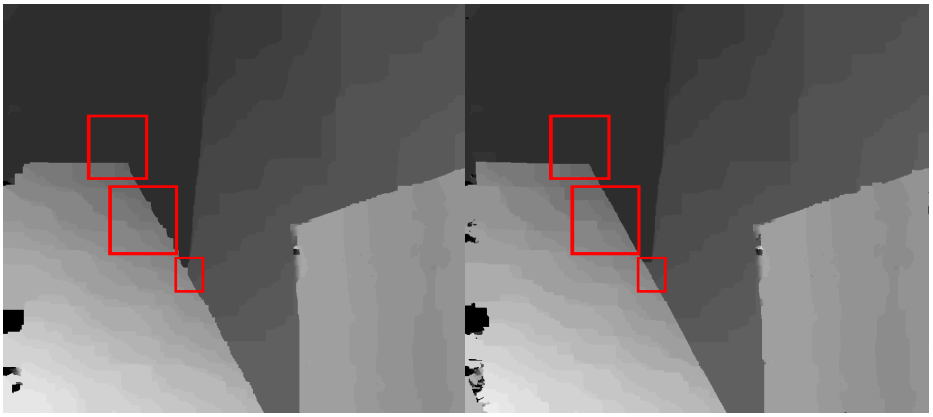
Figure 4.10: Result comparison.

Image	%err. pw+TLM	%err. agg+TLM	%err. CPL+IHVMp
Flowerpots	37.21	**18.15**	19.65
Baby1	9.18	5.73	**2.89**
Baby2	16.80	10.80	**5.64**
Baby3	15.46	6.63	**6.45**
Art	23.57	25.73	**10.29**
Aloe	5.85	18.90	**3.65**
Books	18.30	15.41	**12.24**
Cloth1	**0.15**	4.61	0.20
Cloth2	6.20	10.12	**2.58**
Cloth3	1.97	6.06	**1.27**
Cloth4	3.51	4.44	**1.50**
Dolls	14.10	17.16	**7.58**
Lampshade1	23.94	16.94	**6.96**
Lampshade2	47.91	**10.17**	13.56
Laundry	22.00	23.42	**16.59**
Moebius	13.33	19.27	**9.72**
Wood1	6.98	9.73	**3.12**
Wood2	24.79	9.00	**1.03**
Bowling1	44.89	19.16	**11.20**
Bowling2	22.87	14.18	**9.97**
Rocks1	8.42	7.64	**3.99**
Rocks2	7.66	4.93	**2.49**
Reindeer	18.22	21.21	**7.10**
Avg.Error	17.10	13.02	**6.94**

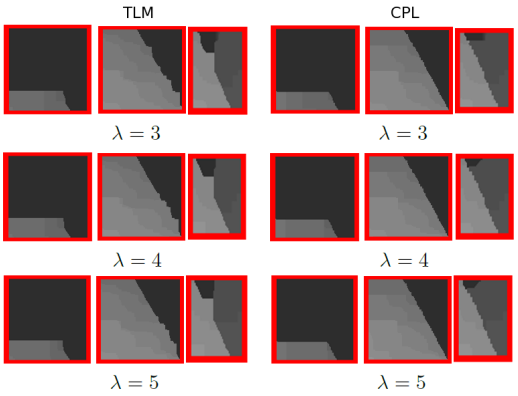
Table 4.1: Error % comparative table. pw+TLM (pixel-wise cost + TLM), agg+TLM (census+Hamming distance+aggregation), CPL+IHVMp.

tance+aggregation, i.e. block matching. The column **pw+TLM** evaluates our proposed similarity (using the pixel-wise and block matching costs) function, the *CPL* model, and local cues including $IHVMp$. The results obtained show the improvement from the pixel-wise and block matching costs contributions, and the *CPL* edge model. This suggests that in the future we should investigate better strategies to localise possible depth discontinuities, as the *CPL* edge model performance depends entirely on the edge location.

4.7 Effect of λ and K on CPL



(a) TLM $\lambda = 3$ (b) CPL $\lambda = 3$

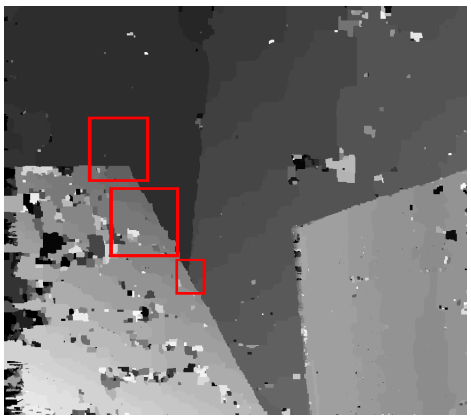


(c) Change of λ in TLM (d) Change of λ in CPL

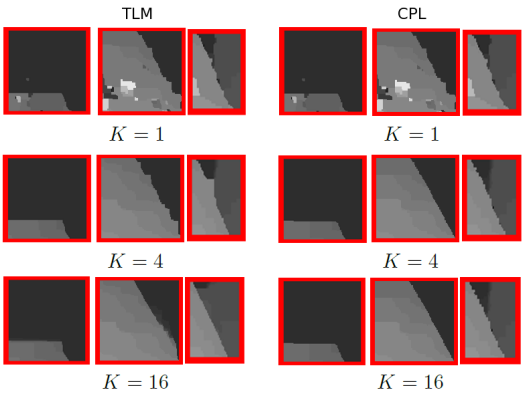
Figure 4.11: Effect of changing λ (venus image), $\alpha = 45$, $K = 4$, similarity function is Birchfield. First column TLM, second column CPL. Red boxes show areas where overgrowth has happened in TLM, but it's either non-existent or reduced in CPL.

To evaluate how the CPL edge model behaves when λ and K change, we compare our results to the classic TLM. The motivation to use CPL is to reduce the overgrowth at edges; for this reason its performance is better exposed when using a pixel-wise similarity measure (Birchfield RGB dissimilarity measure is used). The following evaluations only use the Birchfield dissimilarity measure and IHVMp was not used.

The trade-off between a smooth disparity map and sharp edges is commonly balanced by adjusting λ. Increasing λ gets cleaner results at the expense of edges. Fig.4.11 shows how CPL suffers less from overgrowing edges when increasing λ. In particular it can be seen that CPL is able to keep most of the edges sharp, while edges using TLM overgrow in all directions.



(a) Venus image with $K = 1$



(b) Change of K in TLM (c) Change of K in CPL

Figure 4.12: Effect of changing K (venus image), $\alpha = 45$, $\lambda = 2$, similarity function used is Birchfield. First column TLM, second column CPL. Red boxes show areas where overgrowth has happened in TLM, but it's either non-existent or reduced in CPL.

One possibility to improve the quality of the disparity map is to increase K, which allows more disparity values to be tested. Increasing K using the TLM might end up overgrowing (fig.4.12b). However, using CPL (fig.4.12c) reduces this issue, because CPL does not give the same K to possible depth discontinuities.

Fig.4.12 shows that increasing K in TLM has overgrown the edges, but has been able to produce a smooth surface. It's important to stress that CPL is able to keep edges sharper, but it also is more susceptible to initial edge positions. This could indicate that another possible use of CPL might be in a disparity refinement stage since it's able to keep sharp edges, while keeping smooth surfaces.

4.7.1 Occlusion Detection and Hole Filling

Disparity is computed for both left and right views which allows occlusion detection. Occluded pixels are detected by doing left↔right cross checking according to eq.4.21 (where D_l and D_r are left and right disparity maps).

$$\hat{D}_l(p) = \begin{cases} 0 \text{ if } |D_l(p) - D_r(p + D_l(p))| > 1 \\ 0 \text{ if } p + D_r(p + D_l(p)) \text{ outside image boundaries} \\ D_l(p) \text{ otherwise} \end{cases} \tag{4.21}$$

There may be spurious disparity values (in occluded areas) that pass the cross check test. To detect spurious disparities the probability distribution is computed (i.e. compute the histogram and normalise it) of each disparity present in both the left and right disparity maps, and an n-point (n is 18% of the disparity search range) Parzen window is applied to smooth the probability distribution. Then, a search finds the lowest disparity (*minD*) that is above a threshold (0.0056). The choice of n and *minD* were found to be suitable empirical values. Normally spurious disparity values are lower than *minD*. Next, regions containing disparity with less than 20 pixels are removed. Then, regions smaller than 45 pixels are removed from the left and right disparity map occlusion borders[4]. The occlusion detection and spurious disparity removal steps create holes thet need to filled in. Hole filling takes into consideration the four non-hole[5] neighbours of the missing pixel (up, left, right, down), and uses the minimum disparity value of them. At image borders, where not all four neighbours are available the disparity D_p is selected according to (eq.4.22). The colour and spatial distance are used to measure the similarity of the neighbouring pixels with known disparity the pixel hole, an additional reward is given to the neighbour pixel with the lowest disparity value.

$$D_{p_{hole}} = D_{\hat{q}_{nonhole}} \tag{4.22}$$

$$\hat{q}_{nonhole} = \max_{q_{nonhole}} e^{-3S_{colour}(W(p),W(q)) - dis(p,q) + MDR} \tag{4.23}$$

where:

[4]When disparity is positive these areas are the horizontal positions in [0, size of search range] in the left disparity map, and right horizontal positions in [width-size of search range, width] in the right disparity map.

[5]We call it non-hole rather than non-occluded because removing spurious disparities creates holes that are not necessarily occlusions.

- p is a pixel with missing disparity value, q is a neighbour with a disparity value.
- $W(p)$ and $W(q)$ are $n \times n$ windows around p and q.
- $S_{colour}(A,B) = \frac{1}{w^2}\sum_{i\in\{RGB\}}\sum_{j}^{w^2}|A_j^i - B_j^i|$
- $dis(p,q)$ is the normalised distance (i.e. $[0,1]$) of the neighbour. The image height (when the neighbour is up/down) or image width (when the neighbour is left/right) is used to normalise.
- MDR is the Minimum Disparity Reward. $MDR = 0.51$ if the non-hole pixel q is the second lowest disparity value (similar to [34]) and is a left/right neighbour, otherwise $MDR = 0.50$.

Filling in holes may create a streaking noise effect in the disparity map. To remove this problem we use the weighted median filter from [61] using I_l as a guide with a window of 11×11. Notice that there are other hole filling alternatives such as anisotropic diffusion [15], image in-painting [52], or colourisation techniques [51, 40] that could be adapted to our problem. However, they only take into account colour/intensity similarity, whereas hole filling in disparity needs to use disparity values that are from the background to fill in occluded areas.

4.7.2 Overall disparity estimation process

Algorithmically, the disparity estimation process can now be summarised as follows:

1. Up-scale images by a factor of 2x using bilinear interpolation (to reduce FE from census transform of 9×9).
2. Set τ_{hv} and τ_{cpl} for eq.4.15 and eq.4.16 as $\tau_{hv} = 2\tau_{cpl}$ to reduce the sensitivity of IHVMp to noisy gradients.
3. Set the constants K, K_l, and K_h in eq.4.16 and eq.4.15 as a function of the search range[6] ($K = \min\{floor(0.1 \times SearchRange) - 0.1, 20\}$ when using TRW-S, $K_l = 3$ and $K_h = 12$ when using $IHVMp$).
4. Compute $ENCC_p(d)$ and $\hat{Ct}_d(p)$ and linearly map them to $[0,1]$ to make it suitable for minimisation and to avoid negative contributions.

[6] This is done under the assumption that the cost similarity will have less noise when the disparity search range is small.

5. Compute pixel similarity cost and aggregation according to eq.4.19 and eq.4.20. For time considerations, cost aggregation is done using a two-pass algorithm like in [99].
6. Truncate pixel-wise cost using a threshold of $15/255$ for RGB, and $3/255$ for gradient images. This is done to reduce noise in the cost.
7. Do IHVMp (with CPL) to compute slanted hypothesis and normalise the contributions according to eq.4.17.
8. Do T times message passing using TRW-S (with CPL) using eq.4.17 as the cost to optimise eq.4.1, we set $T = 40$.
9. Compute occlusions via cross-checking, compute lowest disparity value, and eliminate spurious disparities.
10. Fill in occlusions using eq.4.22.
11. Downscale disparity maps, using linear interpolation and round disparities.
12. Use the weighted median filtering from [61], with the non-scaled colour image I_l as the guide to reduce streaking noise.

Fig.4.13 shows a simplified diagram of the proposed algorithm LCU described above. The purple boxes denote the input images. The blue boxes denote known optimization techniques. The orange boxes denote our original contributions in the cost function (pixel-wise and block matching combined), computing the growth limit for the proposed CPL model, and computing the local cues using $IHVMp$. Finally, the green box denotes our postprocessing technique to detect spurious disparities and fill in holes.

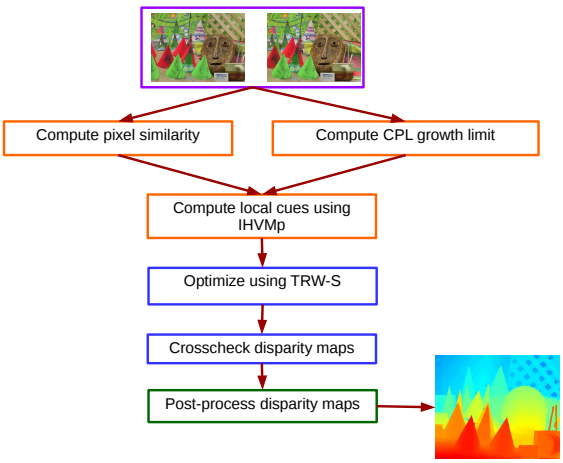
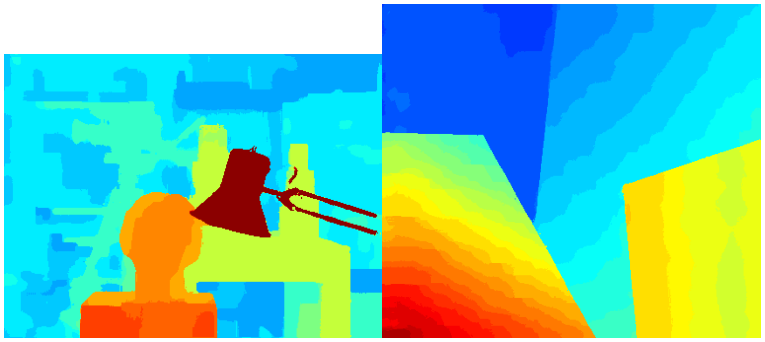


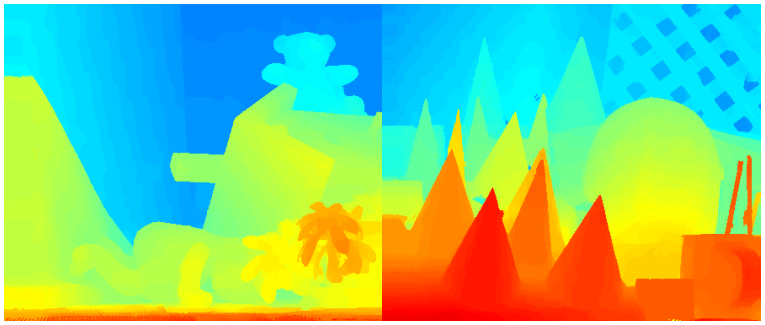
Figure 4.13: LCU algorithm diagram.

4.8 Experimental results

Performance evaluation is done using the Middlebury benchmark[82] (version 2 and 3). The four evaluation images[7] (tsukuba, venus, teddy, cones) were used to adjust the parameters. The tables showing results will contain the following columns: % bad pix. evaluates the percentage of pixels whose disparity is above a predefined threshold. Additional experiments are performed using another 23 images from the same data set (2005 and 2006 data sets), Avg. Error evaluates average disparity error, rmse evaluates the root mean square error of the disparity map, and Avg. Rank evaluates the average rank of the different metrics defined in [82]. The parameters used are $\alpha = 45$ (TRW-S), $\alpha = 30$ ($IHVMp$), $\lambda = 2$, $\sigma_r = 10/255$ (eq.4.3), $w = 41$ (i.e. 41×41 window) and $\tau_{cpl} = 0.05$ (eq.4.16). Fig.4.14 shows the performance of our algorithm on the four test images (Middlebury v.2) after occlusion detection, hole filling and post processing.



(a) tsukuba (b) venus



(c) teddy (d) cones

Figure 4.14: Results of proposed model after occlusion filling and post processing.

Tab.4.2 shows the top performers of the Middlebury data set v2. Our approach is the top performer in the integer accuracy (1 pixel of error) and competitive in the half of a pixel evaluation (rank 12 out of 152), which is noteworthy since this is only the

[7] Benchmark version 2.

Method	Avg. rank	Avg. % bad pix.
OUR RESULT	**11.1**	**3.89**
TSGO [71]	11.2	4.06
JSOSP+GCP [56]	13.0	4.18
ADCensus [65]	15.3	3.97
AdaptingBP [43]	19.2	4.23

Table 4.2: Table of integer disparity performance on Middlebury v2.

result of up-scaling the image. A more detailed comparison is presented in tab.4.5, where *nonocc* evaluates errors in non-occluded regions, *all* evaluates all pixels in the disparity map, and *disc* evaluates errors close to depth discontinuities.

Method	Avg. rank	Avg. % bad pix.
OUR RESULT	**18.65**	7.20
SubPixSearch [68]	19.40	5.85
SegAggr [72]	20.40	5.54
TSGO [71]	23.15	7.63
PM-Huber[32]	24.95	5.95
PMF [58]	25.20	5.88
GC+LSL [88]	25.65	**5.41**
ADCensus [65]	32.75	8.94
JSOSP+GCP [56]	40.65	9.64

Table 4.3: Average integer and half of pixel performance on Middlebury v2.

An interesting observation in the Middlebury leader table is that algorithms that have a high score on sub-pixel accuracy, have a lower score on integer accuracy. This could indicate that these algorithms (e.g. [88] is the number one on half of a pixel accuracy) are producing good sub-pixel results on smooth surfaces, but are giving lower results near depth discontinuities. Tab.4.3 shows that, when combining the rank scores of the two tables, our algorithm still leads the table. What contributes to this result is the top performance in integer accuracy and a competitive result in half of a pixel accuracy (a more detailed comparison is presented in tab.4.5).

Our approach was also evaluated on the Middlebury benchmark (version 3 [81]). The new data set is more challenging since it contains changes in exposure, lighting conditions, and a larger search range. To handle lighting and exposure differences present in the images, we replaced the *RGB* pixel-wise cost with the Birchfield cost on the intensity gradient of the image (i.e. each pixel is represented by $(\nabla_x I, \nabla_y I)$), $\lambda = 1$ (only for *TRW*-S), and the threshold to detect *minDisp* is set to 0.003 due to the larger search range (the rest of the parameters remain the same).

Metric	OURS	IDR	SGM	SNCC	LPS
%bad pix.	**17.0**	18.1	18.4	21.9	19.2
avg. error	**3.86**	6.35	5.23	7.82	85.8
rmse	**16.2**	21.8	20.0	26.1	156

Table 4.4: Comparative table of results (on non-occluded pixels) on the Middlebury v3 data set, %bad pix. uses a threshold of 2 pixels. Our approach uses only quarter size images, the other methods use half size images. IDR[48], SGM[34], SNCC[18], LPS[85].

Tab.4.4, shows the results of our approach on the new data set, where our algorithm ranks first on average error and rmse, even when only using images of one quarter size, and ranks second on percentage of bad pixels (threshold error of 2 disparity levels at the full size image). A more detailed comparison is presented for the top ranking algorithms on the average error in tab.4.7, and RMSE (tab.4.8) measures, while the detailed average percentage error is shown in tab.4.9) using an error threshold of 2 pixels (disparities).

The experiments and evaluation are performed using images that are one quarter size, whereas other algorithms use half size, yet our algorithm is still a top performer. This means that our approach is a competitive approach at half of a pixel, when scaled to the quarter size image. Finally, fig.4.15 shows the screen capture of the Middlebury 2 and 3 websites, and confirms the claimed performance reported above. All the performance reported in this section was done at the time of results submission in November 2014.

Error Threshold = 1 — Error Threshold... — Sort by nonocc — Sort by all — Sort by disc

Algorithm	Avg. Rank	Tsukuba (ground truth) nonocc	Tsukuba all	Tsukuba disc	Venus nonocc	Venus all	Venus disc	Teddy nonocc	Teddy all	Teddy disc	Cones nonocc	Cones all	Cones disc	Average Percent Bad Pixels
YOUR METHOD	11.1	1.06 17	1.34 8	5.50 15	0.07 2	0.26 15	1.03 2	3.66 14	9.95 34	10.4 13	**1.63** 1	6.87 11	**4.82** 1	3.89
TSGO [143]	11.2	0.87 4	**1.13** 1	4.66 6	0.11 8	0.24 9	1.47 11	5.61 40	8.09 17	13.8 33	1.67 2	**6.16** 1	4.95 2	4.06
JSOSP+GCP [151]	13.0	**0.74** 1	1.34 9	**3.98** 1	0.08 3	**0.16** 1	1.15 3	3.96 16	10.1 35	11.8 19	2.28 17	7.91 31	6.74 20	4.18
ADCensus [82]	15.3	1.07 20	1.48 18	5.73 23	0.09 4	0.25 12	1.15 3	4.10 18	6.22 7	10.9 15	2.42 23	7.25 17	6.95 24	3.97
AdaptingBP [16]	19.2	1.11 23	1.37 11	5.79 25	0.10 6	0.21 8	1.44 10	4.22 20	7.06 15	11.8 20	2.48 27	7.92 33	7.32 32	4.23
CoopRegion [39]	19.5	0.87 6	1.16 2	4.61 5	0.11 7	0.21 6	1.54 15	5.16 32	8.31 21	13.0 27	2.79 45	7.18 16	8.01 52	4.41
CCRADAR [152]	23.8	1.15 26	1.42 16	6.23 39	0.15 20	0.27 16	1.89 25	5.39 35	10.6 40	14.7 44	2.01 3	7.37 19	5.88 3	4.75
RDP [87]	25.2	0.97 11	1.39 13	5.00 11	0.21 42	0.38 33	1.89 25	4.84 24	9.94 33	12.6 24	2.53 31	7.69 23	7.38 33	4.57
MultiRBF [129]	25.3	1.33 50	1.56 23	6.02 34	0.13 12	0.17 3	1.84 22	5.09 30	6.36 8	13.4 31	2.90 53	6.76 9	7.10 29	4.39
DoubleBP [34]	25.9	0.88 8	1.29 6	4.76 9	0.13 13	0.45 51	1.87 24	3.53 13	8.30 20	9.63 8	2.90 52	8.78 63	7.79 44	4.19

(a) Middlebury V2

avgerr (pixels) — Weight

Date	Name	Res	Avg	Austr MP: 5.6 nd: 290 im0 im1 GT	AustrP MP: 5.6 nd: 290 im0 im1 GT	Bicyc2 MP: 5.6 nd: 250 im0 im1 GT	Class MP: 5.7 nd: 610 im0 im1 GT	ClassE MP: 5.7 nd: 610 im0 im1 GT	Compu MP: 1.5 nd: 256 im0 im1 GT	Crusa MP: 5.5 nd: 800 im0 im1 GT	CrusaP MP: 5.5 nd: 800 im0 im1 GT	Djemb MP: 5.7 nd: 320 im0 im1 GT	DjembL MP: 5.7 nd: 320 im0 im1 GT	Hoops MP: 5.7 nd: 410 im0 im1 GT	Livgrm MP: 5.9 nd: 320 im0 im1 GT	Nkuba MP: 5.5 nd: 570 im0 im1 GT	Plants MP: 5.6 nd: 320 im0 im1 GT	Stairs MP: 5.2 nd: 450 im0 im1 GT
11/12/14	LCU	Q	**3.86** 1	4.62 4	3.10 3	3.38 4	3.24 2	10.1 3	2.45 3	**2.71** 1	2.30 3	1.54 13	7.94 5	**5.09** 1	4.64 4	**3.39** 1	**4.95** 1	**5.49** 1
11/11/14	MeshStereo	H	4.62 2	3.78 3	3.59 7	4.20 7	5.84 7	**6.75** 1	**2.26** 1	3.17 4	3.29 8	1.36 7	4.94 2	7.46 5	4.38 3	6.78 4	8.30 7	6.30 2
07/28/14	SGM	H	5.32 3	7.93 7	3.16 4	2.20 2	10.5 10	16.2 5	2.45 3	3.50 7	**2.09** 1	**0.98** 1	7.60 4	5.44 3	5.70 5	5.76 3	6.32 3	10.6 8
07/25/14	SGM	Q	5.38 4	8.06 8	5.29 11	4.20 7	5.22 6	9.65 2	3.69 10	3.33 5	2.77 4	1.45 10	10.2 7	6.63 4	4.33 2	5.67 2	7.83 5	12.4 12
10/07/14	IDR	H	6.35 5	7.58 5	**2.54** 1	2.21 3	16.8 15	24.1 7	3.28 9	3.34 6	2.15 2	0.99 2	13.5 9	5.13 2	5.76 6	6.81 5	6.35 4	8.02 4
09/18/14	SNCC	H	7.82 6	12.1 15	5.16 10	3.54 5	17.4 16	27.1 9	2.95 5	5.92 9	3.51 9	1.25 6	**4.92** 1	9.15 7	7.60 13	11.2 14	8.21 6	8.76 7
07/28/14	SGM	F	7.97 7	8.64 9	2.90 2	**2.01** 1	14.7 13	23.5 6	2.38 2	10.2 12	2.95 5	1.40 9	7.20 3	8.44 6	8.47 14	13.2 16	11.7 12	12.0 11
09/10/14	LAMC_DSM	H	8.98 8	12.5 16	7.89 16	7.57 15	6.36 8	12.9 4	5.29 16	7.82 11	6.93 13	1.45 10	10.1 6	11.5 10	8.68 15	11.0 12	16.1 16	19.4 17
07/28/14	Cens5	H	9.34 9	13.5 17	5.70 13	3.88 6	17.5 17	24.6 8	5.78 17	6.32 10	4.83 11	1.53 12	10.6 8	12.7 11	9.13 17	14.0 17	10.3 10	14.1 13
07/28/14	ELAS	H	10.5 10	9.61 12	5.69 12	5.36 11	12.1 12	82.5 11	3.93 13	5.64 8	4.32 10	2.01 16	17.3 10	12.9 12	5.93 8	8.13 8	12.9 13	8.38 5
07/25/14	SGBM1	Q	14.3 11	8.69 10	8.00 17	7.08 14	3.86 5	102 12	5.28 15	15.4 14	15.8 15	2.19 17	33.2 12	18.3 14	6.08 10	10.3 11	14.6 14	18.3 16
07/25/14	ELAS	F	15.0 12	9.62 13	4.11 8	5.34 10	14.7 14	141 14	3.78 11	10.7 13	6.42 12	2.00 15	18.0 11	13.5 13	6.18 11	8.57 9	28.9 18	10.7 9
07/28/14	SGBM1	H	16.0 13	8.84 11	5.70 13	5.50 12	6.94 9	150 15	3.84 12	18.0 15	15.8 15	1.37 8	33.8 13	22.4 16	7.07 12	10.0 10	10.2 9	16.5 14
07/25/14	SGBM2	Q	17.1 14	7.70 6	6.57 15	6.33 13	3.42 3	133 13	4.92 14	26.5 17	27.0 17	1.87 14	37.1 14	19.4 15	5.77 7	11.1 13	15.6 15	11.5 10
08/31/14	BSM	Q	19.9 15	23.4 18	9.58 18	9.29 16	26.7 18	52.0 10	9.81 18	21.6 16	14.9 14	6.31 18	40.5 15	23.9 17	17.8 18	22.6 18	16.8 17	48.0 18
07/25/14	SGBM1	F	22.8 16	10.3 14	4.76 9	5.18 9	10.7 11	183 16	3.17 8	46.0 18	40.4 18	1.15 4	42.5 16	29.9 18	9.12 16	12.8 15	10.6 11	17.7 15
08/27/14	LPS	F	84.6 17	3.65 2	3.37 5	9.41 17	3.65 4	999 17	3.12 7	3.06 3	3.15 7	1.18 5	999 17	10.8 9	**3.92** 1	7.16 6	8.56 8	8.75 6
08/25/14	LPS	H	85.8 18	**3.37** 1	3.55 6	26.7 18	**2.48** 1	999 18	3.00 6	2.90 2	3.05 6	1.07 3	999 18	10.3 8	5.95 9	7.82 7	6.23 2	7.60 3

(b) Middlebury V3

Figure 4.15: Screenshot confirming claimed performance.

Method	Avg. Rank	Tsukuba			Venus			Teddy			Cones		
		nonocc	all	disc	nonocc	all	disc	nonocc	all	disc	nonocc	all	disc
OUR RESULT	**11.1**	1.06	1.34	5.5	0.07	0.26	1.03	3.68	9.95	10.4	1.63	6.87	4.82
TSGO	11.2	0.87	1.13	4.66	0.11	0.24	1.47	5.61	8.09	13.8	1.67	6.16	4.95
JSOSP+GCP	13	0.74	1.34	3.98	0.08	0.16	1.15	3.96	10.1	11.8	2.28	7.91	6.74
ADCensus	15.3	1.07	1.48	5.73	0.09	0.25	1.15	4.10	6.22	10.9	2.42	7.25	6.95
AdaptingBP	19.2	1.11	1.37	5.79	0.10	0.21	1.44	4.22	7.06	11.8	2.48	7.92	7.32

Table 4.5: Average integer disparity performance on Middlebury v2; nonocc: non occluded pixels, disc: pixels close to depth discontinuities, all: all pixels.

Method	Avg. Rank	Tsukuba			Venus			Teddy			Cones		
		nonocc	all	disc	nonocc	all	disc	nonocc	all	disc	nonocc	all	disc
OUR RESULT	**18.65**	9.03	9.27	11.10	0.76	1.13	4.10	5.52	12.43	14.65	2.81	8.64	7.15
TSGO	23.15	4.18	4.62	8.38	3.77	4.02	6.19	9.16	12.25	19.80	3.38	8.18	7.78
JSOSP+GCP	40.65	11.37	11.92	10.54	4.63	4.84	6.18	7.78	14.35	17.85	4.96	10.76	10.72
ADCensus	32.75	13.94	14.24	13.42	2.07	2.43	4.58	7.35	10.01	15.50	4.50	9.83	9.43
SegAggr	20.40	7.20	7.65	12.95	0.20	0.31	1.89	3.26	5.77	9.51	2.61	7.58	7.50
SubPixSearch	19.40	3.82	4.36	7.93	0.61	1.02	4.55	5.36	8.70	13.95	3.13	8.32	8.40
LAMC-DSM	29.25	5.48	6.14	9.68	0.86	1.35	5.66	5.96	12.50	15.85	3.05	9.66	7.95
PMF	25.20	6.37	6.72	12.04	0.53	0.71	4.72	3.49	7.66	11.00	2.51	7.56	7.27
PMBP	32.00	6.93	7.26	13.51	0.58	0.80	5.01	4.24	10.29	12.25	2.85	7.76	7.95
PM-Huber	24.95	5.31	5.95	11.41	0.61	0.92	5.15	4.46	7.46	13.30	2.43	7.30	7.09
GC+LSL	25.65	3.74	4.15	12.30	0.46	0.62	4.36	3.11	5.45	9.95	3.27	8.27	9.23
ARAP	30.85	5.12	5.61	13.90	0.51	0.70	5.52	4.27	8.59	12.56	2.54	7.64	7.26
SNCC+AM	50.25	6.59	6.99	16.65	0.49	0.78	4.62	7.56	12.35	20.45	3.79	9.86	10.84

Table 4.6: Average combined integer and half of a pixel performance on Middlebury v2; nonocc: non occluded pixels, disc: pixels close to depth discontinuities, all: all pixels.

Image	OURS	IDR	SGM	SNCC	LPS
Austr	4.62	7.58	10.6	8.76	**3.37**
AustrP	**3.1**	2.54	7.93	12.1	3.55
Bicyc2	3.38	2.21	**3.16**	5.16	26.7
Class	3.24	16.8	**2.2**	3.54	2.48
ClassE	**10.1**	24.1	10.5	17.4	999
Compu	**2.45**	3.28	16.2	27.1	3
Crusa	2.71	3.34	**2.45**	2.95	2.9
CrusaP	2.3	**2.15**	3.5	5.92	3.05
Djemb	1.54	**0.99**	2.09	3.51	1.07
DjembL	7.94	13.5	**0.98**	1.25	999
Hoops	5.09	5.13	7.6	**4.92**	10.3
Livgrm	**4.64**	5.76	5.44	9.15	5.95
Nkuba	**3.39**	6.81	5.7	7.6	7.82
Plants	**4.95**	6.35	5.76	11.2	6.23
Stairs	**5.49**	8.02	6.32	8.21	7.6
Avg. Err.	**3.86**	6.35	5.32	7.82	85.8

Table 4.7: Average error on test data on Middlebury v3.

Image	OURS	IDR	SGM	SNCC	LPS
Austr	**18.3**	22.1	22.6	30.2	17.1
AustrP	16.3	**15.6**	17.8	22.8	18.4
Bicyc2	13.9	**10.3**	10.9	15.1	675
Class	19.8	48.3	34.6	50.3	**15.8**
ClassE	**33.7**	55.1	40.5	58	999
Compu	**7.48**	12.8	8.67	10.1	10.9
Crusa	**13.5**	14.9	16.3	20.8	16.2
CrusaP	**12**	13.1	12.5	16.9	17.3
Djemb	5.09	4.04	**4.01**	5	4.17
DjembL	**20.2**	35.5	22	15.7	999
Hoops	22.5	**20.6**	23.6	31.2	147
Livgrm	**15.9**	17.9	17.8	19.1	17.2
Nkuba	**17.5**	27.1	27.5	51.7	52.9
Plants	**22.4**	25.7	26	30.9	27
Stairs	**23.3**	31.2	40.1	32.7	26.4
Avg. RMSE	16.2	21.8	20	26.1	156

Table 4.8: Average RMSE on test data set on Middlebury v3.

Image	OURS	IDR	SGM	SNCC	LPS
Austr	24.7	37.5	40.3	48.6	**6.14**
AustrP	7.59	4.08	4.54	6.98	**5.34**
Bicyc2	11.6	7.49	8.03	9.79	**9.24**
Class	11.9	23.3	22.9	25.7	**7.53**
ClassE	**27.9**	40.6	40.5	46	96
Compu	14	12.8	11.4	12.4	**12.3**
Crusa	19.3	24.5	24.7	36.8	**9.61**
CrusaP	15.8	11.3	10.1	16.6	**9.4**
Djemb	8.1	5.46	5.4	7.25	**5.18**
DjembL	36.1	33.1	29.6	**23.1**	92.4
Hoops	29.1	26	28.5	34.2	**27.4**
Livgrm	**21.3**	21.5	23.9	26.7	24.3
Nkuba	**18.4**	21.7	20	21.8	23
Plants	14.1	15.3	14.2	19.9	**10**
Stairs	23.8	**21.2**	30.9	28.4	25.6
Avg. %bad pix	**17**	18.1	18.4	21.9	19.2

Table 4.9: Average % of bad pixels on test data on Middlebury v3.

4.9 Critical analysis discussion

The results presented so far have shown that the proposed similarity function, edge model, and local cues are able to produce sharp edges, keep slanted planes (along the epipolar direction), reduce the fattening effect, and reduce the overgrowth effect. Although our approach delivers top performing results for its generation (as of November 2014) it still faces the following issues:

- **Balancing pixel-wise**: The proposed cost function performed well with a small search range, as the cost was not too noisy or ambiguous. However, if the search range or image resolution is larger (compared the test images) then the pixel-wise cost becomes very noisy and it is therefore not a trivial task to balance it.

- **CPL edge location**: The edge model used relies on the assumption that intensity and depth edges are aligned. However, if the image is too noisy or the assumption does not hold, then using the *CPL* model will result in a noisy disparity map due to large portions of the image using the Potts model. Fig.4.16 shows a low quality image captured using a webcam. Fig.4.17 shows the errors produced by the *CPL* model due to incorrect edge location, whereas fig.4.18 shows the TLM

model recovering a plausible disparity value as no edges are assumed to exist.

- **Balancing local cues**: The proposed model used the local cues computed using $IHVMp$ that contain information about slanted surfaces in the epipolar direction, which are then combined with the similarity cost. When doing inference (we use TRW-S) and depending on the smoothness term weight value, it can cause the local cues to be irrelevant and the slanted surfaces could be lost, which suggest that a mechanism to balance the local cues is needed. Fig.4.19 shows the reference image, fig.4.20 shows the groundtruth, whereas fig.4.21 shows our result (they look almost identical). However, fig.4.22 shows that the slanted surface has almost being replaced by a fronto-parallel plane despite using the proposed local cues.

- **Half of pixel accuracy**: The proposed approach can only compute disparity maps that are accurate up to half of a pixel by up-scaling the images and computing disparity using fronto-parallel windows. This limitation has resulted in the rank of our algorithm in dropping in the Middlebury benchmark from 2^{nd} place in November 2014 to 16^{th} as the time of submission of this thesis. The main reason behind this is the lack of true sub-pixel accuracy and handling of slanted surfaces.

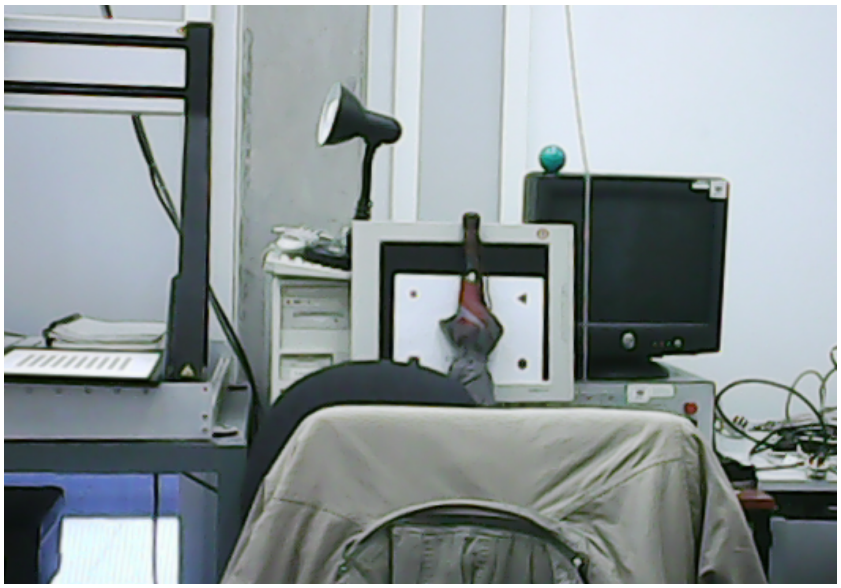


Figure 4.16: test image “lab01” (left reference).

Figure 4.17: Disparity map test image “lab01” (using CPL). Red boxes show failures of the CPL model (too large or low disparity values); Green boxes show correct estimation.

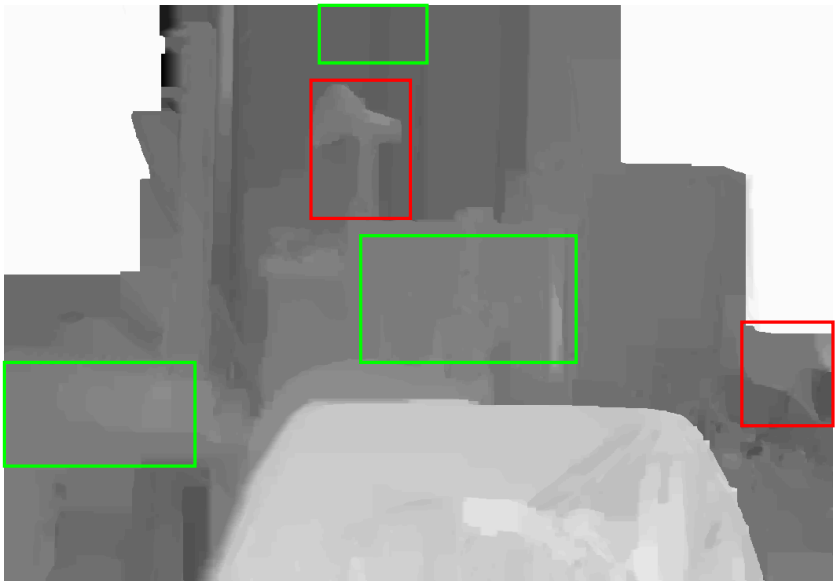


Figure 4.18: Disparity map test image “lab01” (using TLM). Red boxes show failures of the TLM model (blurred details); Green boxes show correct estimation.

Figure 4.19: Middlebury test image “Wood1” (left reference).

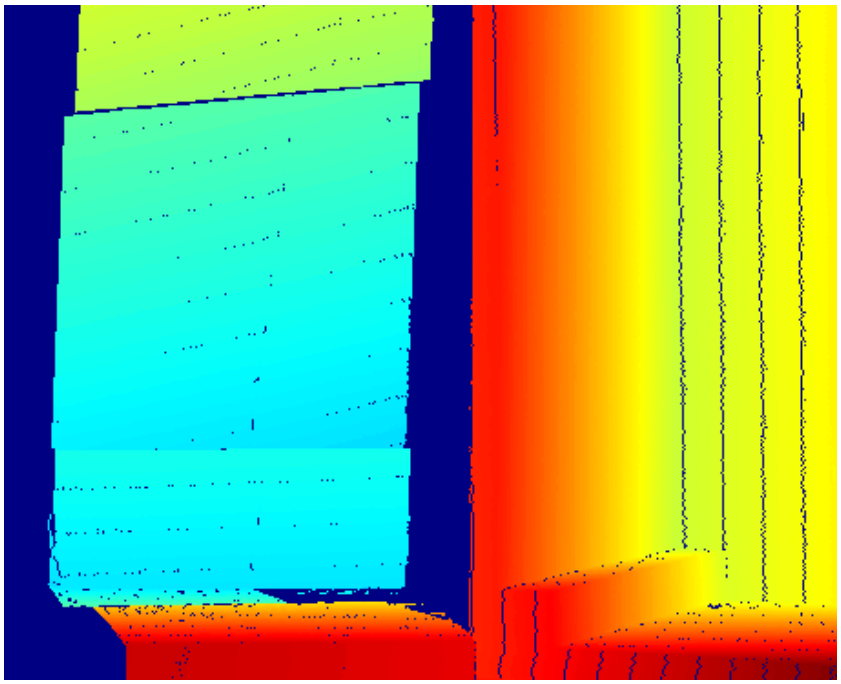


Figure 4.20: Groundtruth disparity map test image “Wood1” (Middlebury data set).

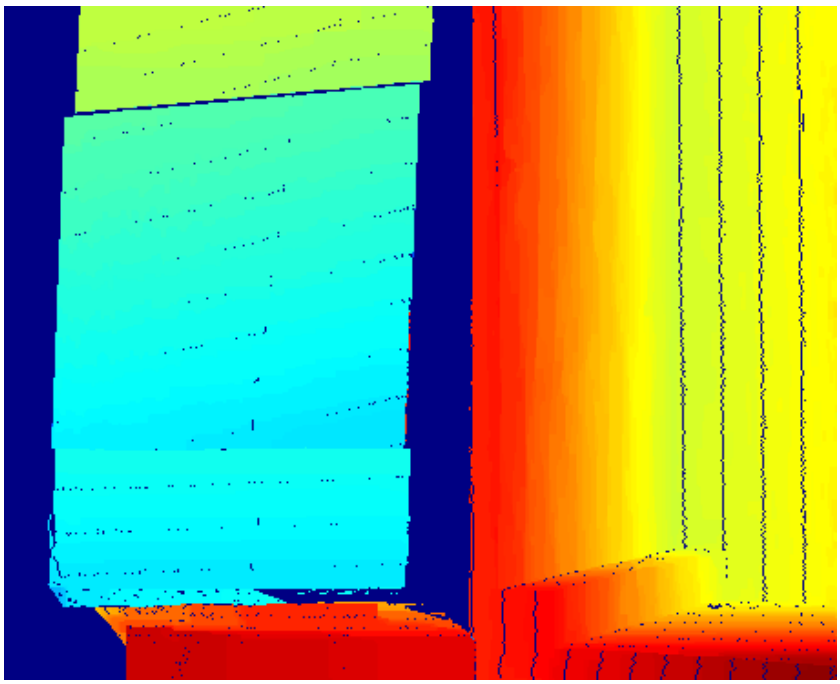


Figure 4.21: Disparity map for image “Wood1” computed using the proposed approach.

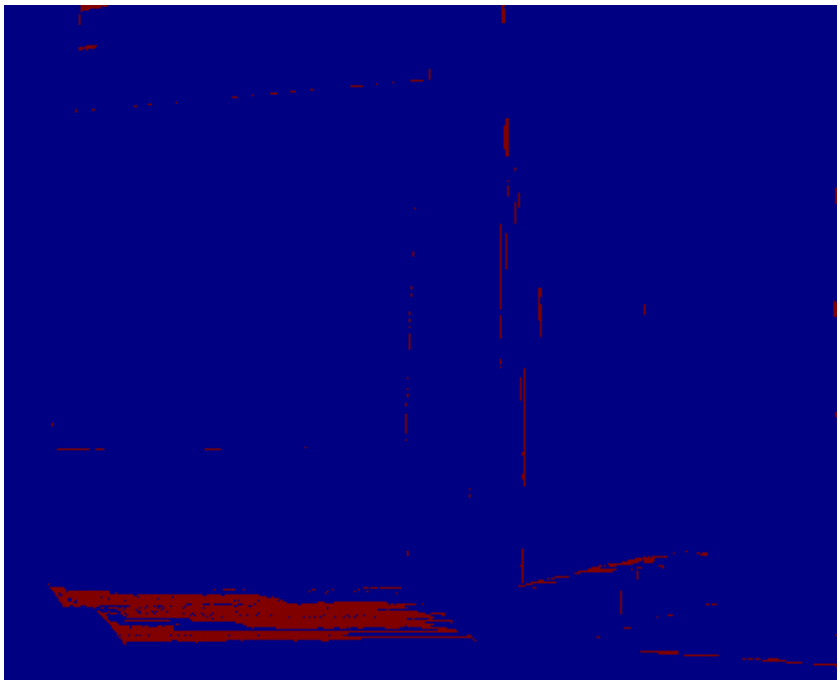


Figure 4.22: Error mask for image “Wood1” computed using the proposed approach.

4.9.1 Error oscillation

Fig.4.23 shows how the average disparity error (evaluated in non occluded areas) shows oscillations during the message passing iterations (using the Teddy image). This is not surprising due to the pixel-wise cost/local cues balancing issue described above, and floating point rounding errors. However, it is also noticeable that the oscillation is minimal and does not affect negatively the final results.

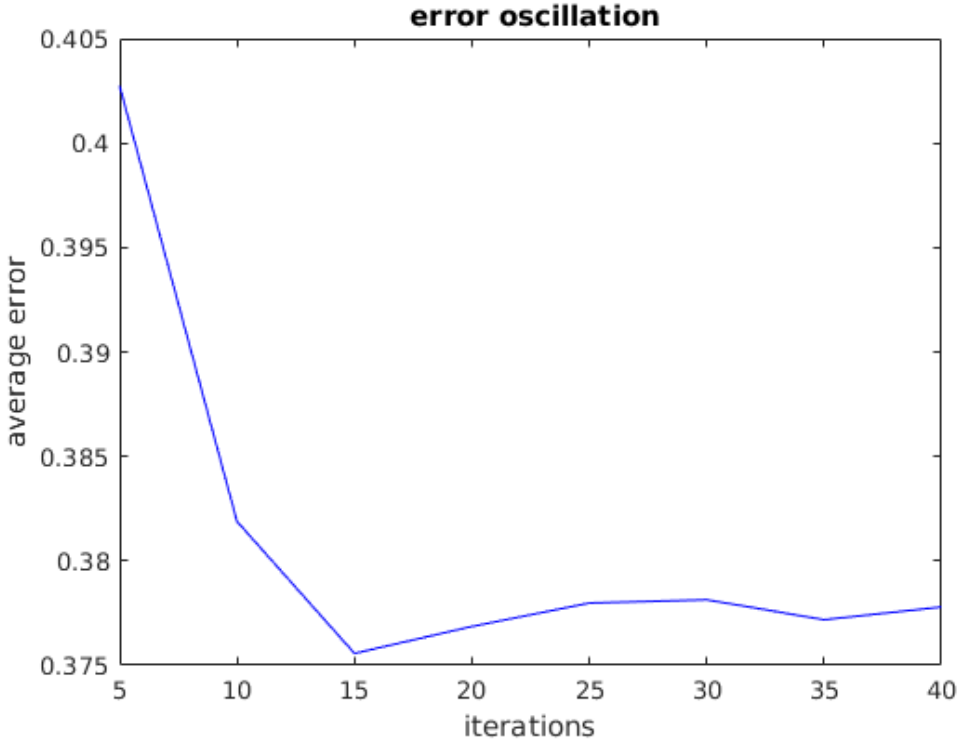
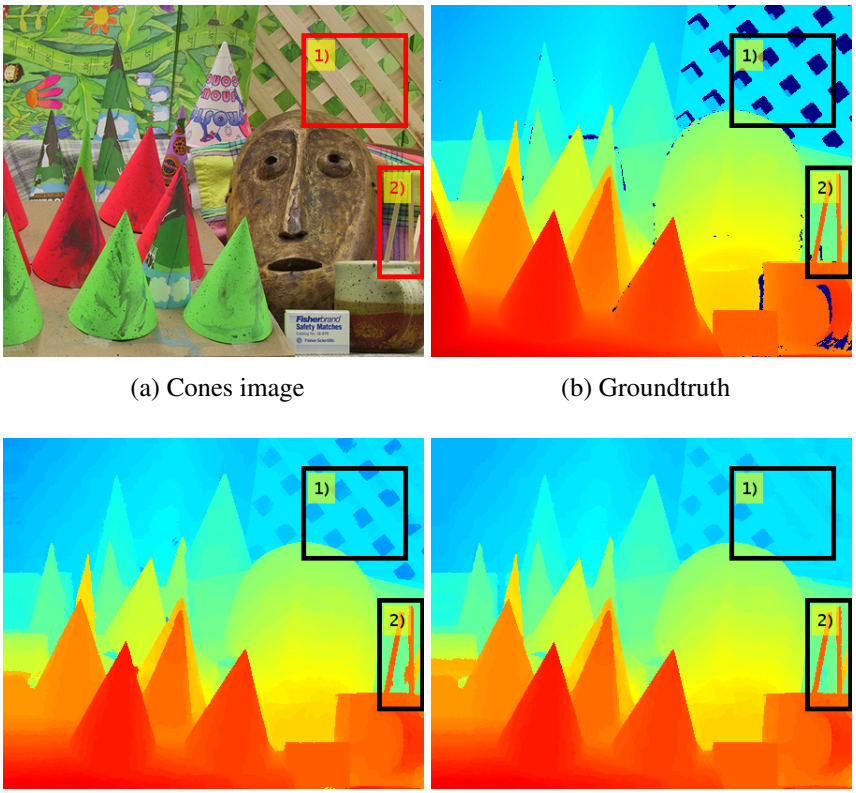


Figure 4.23: Error oscillation on Teddy image.

4.9.2 Groundtruth initialisation performance

Here we make a visual comparison of the "Cones" and the different results obtained when using a groundtruth disparity map to compute possible edge locations for the *CPL* model. Fig.4.24 shows two regions of evaluation 1) depth discontinuities close to occluded areas, and 2) depth discontinuities close to fine details. Fig.4.24a shows the left reference image, fig.4.24b is the groundtruth disparity map, fig.4.24c is the result obtained by computing the *CPL* edges using our algorithm, and fig.4.24d is the result obtained using the groundtruth as initialisation to compute the *CPL* edges. Using the groundtruth results in lower error on three different metrics (in non-occluded

areas) **%bad pixels** above 1 pixel 1.44 (initialised) vs. 1.63 (non-initialised), **average disparity error** 0.25 vs. 0.27, and **rmse** 0.77 vs. 0.97. This can be explained by the absence of texture (i.e. no noise in the *CPL* edges) in the groundtruth, which results in a smoother disparity map.



(a) Cones image (b) Groundtruth

(c) Non-Initialised (d) Initialised

Figure 4.24: Effect of using the groundtruth disparity map to compute *CPL* edges.

1) Using the groundtruth (fig.4.24d) has resulted in overfilling the holes in the background, which is likely caused due incorrect edge localisation. The result using our algorithm (fig.4.24c) with the intensity image has resulted in a plausible disparity value for such region. However, the groundtruth does not have any value in that area.
2) Using our algorithm (fig.4.24c) has resulted in overgrowth of the foreground (fine details), likely caused due to wrong estimation of the possible depth discontinuity. By contrast using the groundtruth edges in the same region (fig.4.24d) results in the correct disparity for both foreground and background.

It would be expected that the result from using the groundtruth to show an even lower error than observed. However, there are several factors that prevent us from obtaining perfect results. 1) Noise in the cost, which cannot be avoided due to image

noise, or lighting conditions. 2) 4-connected MRF, which does not model interactions with all pixels. 3) Fronto-parallel bias, which introduces error in all slanted or curved surfaces.

4.10 Summary

The pixel cost function described is tolerant to small radiometric differences due to the use of the census transform of the intensity image and pixel-wise cost using the gradient from intensity images, but an alternative pixel-wise cost could be explored. Another area of improvement is the optimisation, as the current algorithm gets stuck in a local minimum, which makes it sensitive to noisy images. Tab.4.5 and tab.4.6 highlight this problem, which is most noticeable pixels close to depth discontinuities where the percentage of erroneous pixels is 3-10 times higher than in non-occluded pixels. A better initialisation of the edge positions might help, which could also enable it to be used for disparity filtering since it's able to keep sharp edges and smooth surfaces. However, the current algorithm can only produce sub-pixel disparity accuracy up to half of a pixel, which may not be enough for certain application such as novel view rendering. A less obvious issue in our proposed algorithm is the weight assigned to the pixel-wise part assuming that the cost has low noise, which may not necessarily be the case with outdoor scenes, or may represent a problem when the search range is large. The use of eq.4.3 has the disadvantage of having a high computational cost $O(w)$ even when doing a two-pass approximation. It may be worth exploring alternatives to reduce the computational time. The evaluation time of eq.4.12 can also be reduced by taking advantage of the parallelism of computing vertical and horizontal messages independently.

Finally, the local cues described above have been successful at handling noisy textures, thus avoiding the creation of false edges in the disparity map. We have found that including information about the epipolar direction helps to handle slanted surfaces without the need of using adaptive slanted windows. Finally, our Combined Potts Linear edge model has helped to reduce overgrowth, and our combined adaptive window + pixel-wise cost function helped to reduce the fattening effect and allowed recovery of fine details. The combination of these innovations places our algorithm at the top of the integer and combined integer half of a pixel Middlebury V.2 lead tables, and as a top performer on lead table V.3. The initial parameters used were selected based on the visual quality of the depth discontinuities of the estimated disparity maps.

Chapter 5

Dense stereo matching using local cues with optimal per pixel plane assignment

In this chapter, we show how the classic problem of stereo matching using global optimisation can be significantly improved (compared to chapter 4) by using a cost function that combines both pixel-wise and block matching costs, modifying the smoothness term, including information about surface direction, and optimally assigning a plane per pixel. Our algorithm exploits both the underlying 3D structure and image entropy to generate an adaptive matching window.

The approach discussed in this chapter is an improvement of the algorithm "using local cues to improve dense stereo matching" (*LCU*) presented in the previous chapter. There are two reasons to improve the *LCU* algorithm:

- Need to estimate sub-pixel accurate disparity maps with more than half of pixel precision.
- Need to estimate disparity of curved and slanted surfaces.

The presented algorithm estimates real valued disparity maps by smartly exploring a 3D search space using a novel hypothesis generation approach that acts like a propagation scheduler. The proposed approach is a Dynamic Plane Inference (*DPI*) algorithm and it is among the top performing results when evaluated in the Middlebury, and KITTI 2015 benchmarks. The algorithm presented in this chapter is referred to as Local Planes with Uniqueness term (*LPU*).

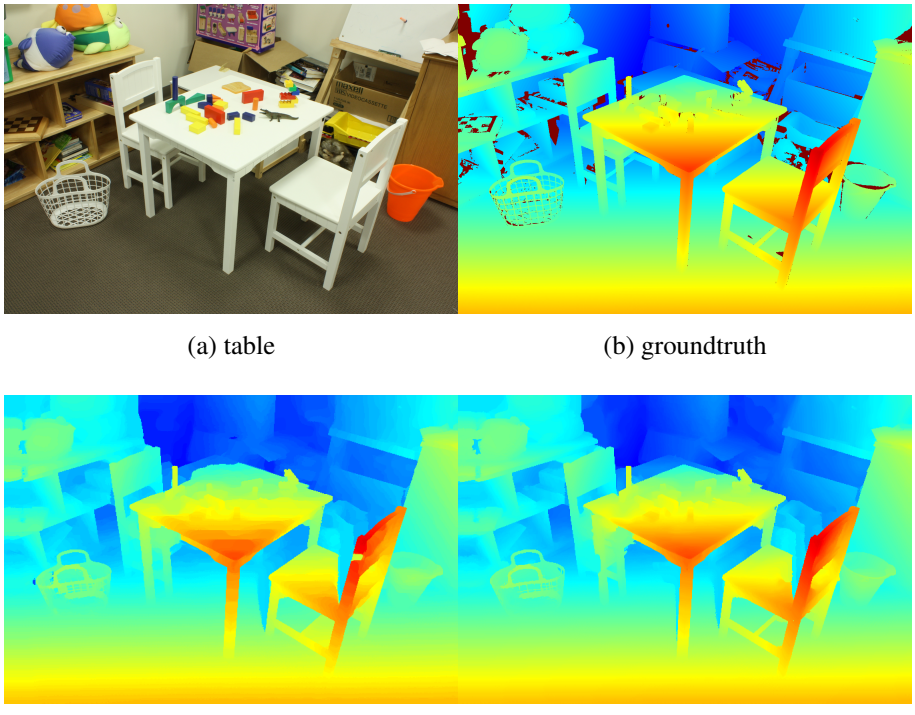
5.1 Differences/Similarities from LCU

The algorithm proposed in this chapter shares the following characteristics with LCU:

- The edge model uses the same criterion to locate possible edges.
- Local cues are computed in the same way.
- Inference is done using *TRW*-S.

However, the current algorithm differs significantly in the following areas:

- The cost function uses an adaptive slanted support window (similar to [9]), which takes into account surface orientation.
- The pixel-wise part and block matching (truncated Hamming distance of census transform) are aggregated together to estimate the pixel similarity cost.
- The search space is no longer discrete and limited to an integer valued disparity, but is now a 3D space and disparities are real valued.
- The smoothness term now uses the normal vectors in a similar way to [88].



(a) table (b) groundtruth

(c) LCU (d) new approach LPU

Figure 5.1: Result comparison for table image.

Fig.5.1 shows the comparison of the *LCU* algorithm and the new approach (*LPU*-Local Planes with Uniqueness). The *LCU* result has a noticeable staircase effect on the table, whereas *LPU* presents a smooth slanted surface. It is precisely because of the possibility of recovering slanted surfaces with sub-pixel accuracy that the *LPU* algorithm was developed.

5.2 Contributions

In order to estimate sub-pixel disparity most modern stereo matching algorithms make the assumption that a 3D scene can be represented locally as 3D planes. Our approach follows this same line of work, in particular the common structure of DPI algorithms is combined with the simplicity of the Fixed Plane Inference (FPI) algorithm.

It is relevant to briefly discuss the common issues in DPI and FPI algorithms which include the use of large adaptive windows (commonly using [112]) or segment-based functions [55] to compute the similarity/dissimilarity cost, which can result in a strong bias towards large planes. A possible solution to this issue is to change the window size, which requires either and assumption about the 3D structure or about the image content.

However, reducing the window size can result in poor performance in textureless areas and thus requires additional assumptions such as the uniqueness constraint or occlusion penalties. Another issue in DPI algorithms such as [9, 6] is that the hypothesis generation and propagation is done sequentially. Whereas the main issue with FPI algorithms is that their performance is entirely dependent on the initial disparity estimate.

To address some of the issues described the our approach makes the following contributions:

- Content aware adaptive window aggregation: Reduces error and loss of details.
- Use of a cost function that imposes a local hypothesis uniqueness constraint, unlike [43, 104, 73, 106, 85, 9, 6, 88]: Helps to handle textureless surfaces and does not require higher order interactions (unlike [96]) in a MRF.
- Use of a cost function that penalises disparity values outside a defined search range: Prevents invalid disparity values assignment.

- Use of a single global hypothesis per disparity plane: Eliminates the need to update multiple hypotheses, and the reduces memory requirements.
- Use of a hypothesis generator that acts as a propagation scheduler: Helps to search a 3D space.

The proposed approach for estimating a 3D plane labelling per pixel has the following components:

- Slanted windows to compute the data term (pixel similarity measure).
- Uniqueness and out of range terms.
- Adaptive search range.
- Smoothness term that adapts to image content.
- Hypothesis generation/update that acts as a propagation scheduler.

The novel content aware windows exploit both intensity and 3D structure (if available) to adapt the window size and similarity function. It is important to note that the proposed hypothesis generation provides an alternative to sequential algorithms (e.g. [9, 6]) and does not impose restrictions to the pairwise interactions unlike [88].

5.3 Proposed model

The algorithm presented in this chapter assumes that using 3D plane labelling is estimated using two images (left I_l and right I_r views of a scene) and consists of finding the correspondences for each pixel from image I_l to I_r by assigning a 3D plane that produces a real valued disparity.

The proposed algorithm is cast as an optimisation problem, which computes the disparity assignment that minimises eq.5.1, which is an updated version of eq.4.2.

$$E(D) = \arg\min_{D} \sum_{p}^{NumP} \{C_p(D_p) + U(D_p) + O(D_p) + \sum_{q \in N(p)} V_{pq}(D_p, D_q)\} \qquad (5.1)$$

where now there are four terms, three unary and one pairwise:

- $C_p(D_p)$ is a unary term that represents the pixel cost function.

- $U(D_p)$ is a unary term that penalises pixels with more that one match.
- $O(D_p)$ is a unary term that penalises pixels that have a disparity value that is outside a pre-defined search range.
- $V_{pq}(D_p, D_q)$ is a pairwise smoothness term.

These terms will have a slightly different formulation than in chapter 4. In eq.5.1 $E(D)$ is the cost of the disparity assignment (energy), D is a set of planes and D_p encodes the plane at pixel p, that gives the disparity of the pixel at p with respect to another image. $D_p(q)$ is the disparity estimated using plane D_p evaluated at pixel q. The plane D_p has two parameters: a 3D unit normal vector $\hat{n}_p = (n_p^x, n_p^y, n_p^z)$ and disparity d_p. The disparity of pixel $q = (x_q, y_q)$ using D_p is given by:

$$D_p(q) = a * x_q + b * y_q + c \tag{5.2}$$

where a, b, and c are real valued parameters, which allows one to obtain real valued disparities when evaluating the plane D_p.

5.4 Uniqueness and out of range terms

The core issue of stereo matching is to find image correspondences from one image to another. However, this has the implicit assumption that each pixel has a unique match, which in general is not necessarily true, especially in occluded areas and textureless regions are prone to have multiple matches, because of limited or non-existent image evidence, so this needs to be controlled. The solution commonly followed is penalising each pixel that has multiple matches (similar to [45]). Eq.5.3 is the uniqueness term used in eq.5.1.

$$U(D_p) = \begin{cases} \tau_{unique} & : L(Dp) \\ 0 & : \text{otherwise} \end{cases} \tag{5.3}$$

where $L(Dp)$ is true when a pixel at p is mapped to $d + D_p$ which itself has more than one match, τ_{unique} is a constant penalty. This is done per pixel and disparity hypothesis, which means it is a local uniqueness term (i.e. unary term), unlike [45] and [96] where it is represented as part of the MRF.

Fig.5.2 shows an example of uniqueness constraint violation; two pixels (red arrows) in the left image scanline map to a single pixel in the right image (red pixel).

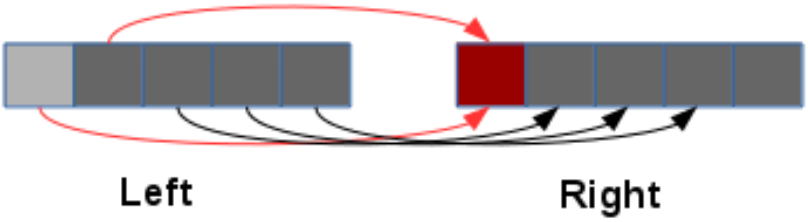
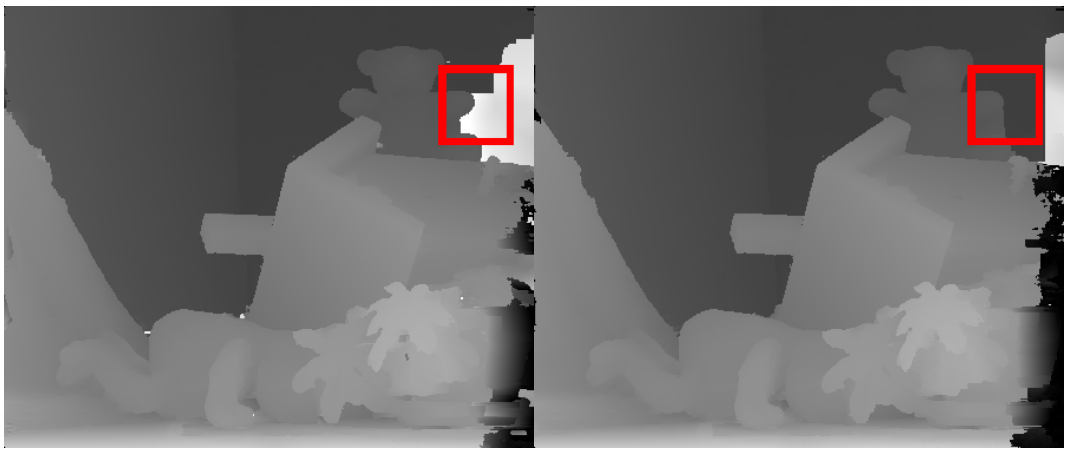


Figure 5.2: Uniqueness constraint violation.

This type of uniqueness penalty commonly uses higher order interactions (e.g. [96]) in a MRF. However, here the uniqueness term is assumed to be local. Although, a local penalty is simpler to compute, it requires to keep track of its current value after inference to update the current cost and prevent energy from oscillating if the current 3D labelling is to be re-used later.

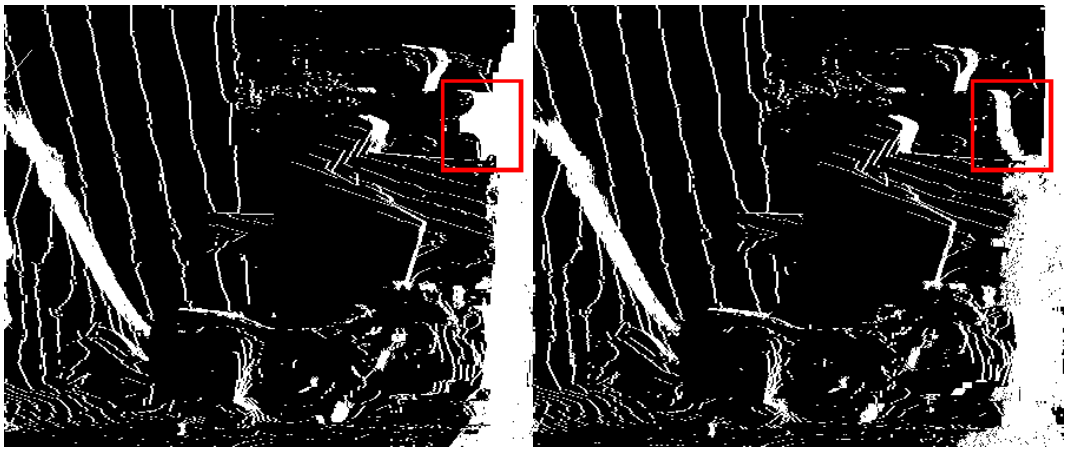
(a) Reference view (b) Groundtruth



(c) Uniqueness term disabled (d) Uniqueness term enabled

Figure 5.3: Reference image, groundtruth and estimated disparity maps.

To further illustrate the need for the uniqueness term fig.5.3 compares two disparity maps, and the effects on an almost textureless region (red boxes). When the uniqueness term is disabled the resulting disparity map has a large error (large white patch). By contrast enabling the uniqueness term eliminates the large error, and most importantly it shows that the local term does work. Fig.5.4 shows the non-unique pixel map, where a white pixel represents a pixel that has multiple matches (some of them might be occluded). Notice that the area around the top left (red boxes) has multiple matches when the uniqueness term is disabled, but less pixels have multiple matches when the term is enabled.



(a) Uniqueness term (b) Uniqueness term enabled

Figure 5.4: Uniqueness term visualisation. White pixels have more than one match, note the reduction of multiple matches in the red boxes.

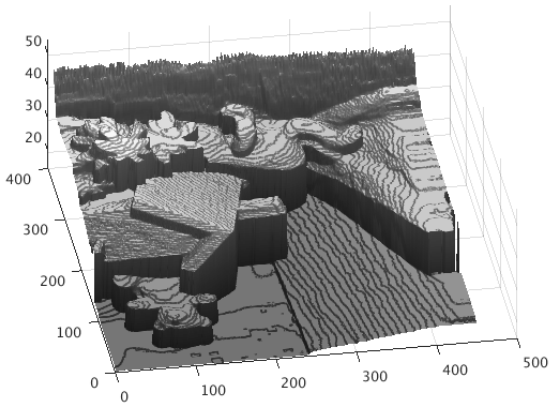


Figure 5.5: The 3D scene is only possible inside the disparity search range $[0, 50]$.

Since the proposed algorithm is a DPI algorithm each pixel has a plane D_p, which evaluated at a different pixel q in the image could result in a disparity value outside a pre-defined search range. Invalid disparity assignments can occur in occluded or textureless areas. In order to reduce the possibility of having an invalid disparity assignment eq.5.4 is used to penalise disparity values that lie outside a defined search range. Fig.5.5 shows a 3D scene which is only defined inside the disparity search range $[0,50]$, and thus any value that lies outside the range is penalised using eq.5.4, where *minD* and *maxD* are the minimum and maximum of the disparity search range, while σ_d scales the deviation allowed for values outside the search range.

$$O(D_p) = \begin{cases} 1 - exp(-|D_p - minD|/\sigma_d) & : D_p < minD \\ 1 - exp(-|D_p - maxD|/\sigma_d) & : D_p > maxD \\ 0 & : \text{otherwise} \end{cases} \tag{5.4}$$

5.5 Content aware adaptive windows

Most DPI algorithms use the adaptive window (*AW* in eq.5.6) from [112], which has the following three characteristics:

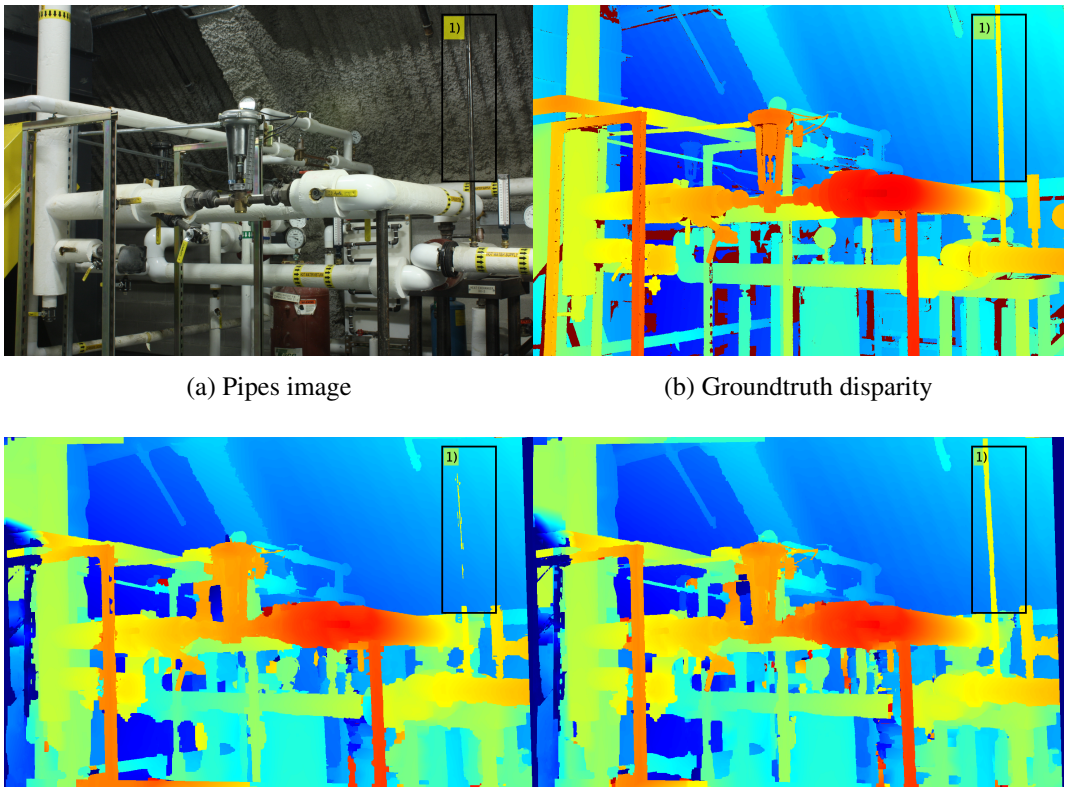
1. Bias towards large planes that may not be able to recover fine details, but good quality in textureless areas that are on planes.
2. Centre pixel bias for large windows resulting in incorrect labelling of a surface due wrong pixel adaptive weight computation.
3. Noisy results for small windows.

The centre pixel bias often results in loss of detail, which can be explained by two different reasons:

Adaptive weight ambiguity: Suppose that centre pixel at p in eq.5.6 has similar intensity as its neighbour at q although their 3D position may be different. Under this condition using an aggregation window can result in similar costs at different disparities, and due to the implicit plane bias some structure can be lost. For instance, fig.5.6 compares the standard aggregation algorithm with ours.

Smoothness term bias: A large weight for the smoothness weight makes the centre pixel bias even more pronounced. For instance, in fig.5.6 the vertical

pipe at the right has almost the same cost at different disparities when using the standard aggregation algorithm. This results in the smoothness term having a lower value if the same disparity as the background is kept, and thus resulting the wrong disparity assignment.



(a) Pipes image (b) Groundtruth disparity

(c) AW result (d) $AW + CW$ result

Figure 5.6: Raw result of Pipes image with c) AW (traditional approach) and d) $AW + CW$ (proposed approach) functions.

From these two problems, the adaptive weight ambiguity is the one that needs to be addressed as it reduces capability of the pixel cost to discriminate between hypotheses at different disparities. This issue can be reduced by using the windows from [119] that make the assumption that image intensity/colour is enough to compute the shape of an adaptive window that adapts better to local image changes. However, the adaptive windows from [119] need to estimate large local vertical and horizontal neighbourhoods (see chapter 2 for details) to give good results. If doing per pixel 3D plane labelling it requires intensive use of pixel interpolation which is expensive. Another problem is that cross aggregation gives poor results if the windows are poorly estimated or window size is not large, e.g. noisy or wrong results particularly in textureless areas.

The solution developed to reduce the centre pixel bias is to combine the adaptive windows from [112] and [119]. Combining these two aggregation algorithms is motivated by the following reasons:

- Keep the good performance of [112] in textureless areas.
- Use [119] where [112] has problems (e.g. when the centre pixel bias is present).
- Use the same window size for [119] and [112].

Combining these two aggregation algorithms pose two questions:

1. How to make the resulting values of two different aggregation algorithms similar?
2. How are these algorithms combined/balanced?

The range of resulting values from [112] (eq.5.6) and [119] is different, because the cross windows do not use adaptive weights. Not using adaptive weights is equivalent to giving all pixels the same weight, which is not necessarily helpful, especially if the vertical and horizontal neighbours are not perfectly estimated or do not coincide with the underlying 3D surface. To take into account this limitation from [119] adaptive weights and normalisation are introduced, and thus the cross aggregation algorithm is updated to become eq.5.7.

In order to combine/balance eq.5.6 and eq.5.7 it is necessary to find a criterion. As noted previously, using cross aggregation might result in poor performance in textureless areas depending on the window size. Our approach uses a fixed limit for the neighbourhood estimation to reduce the use of costly pixel interpolation. Thus, the performance of the cross aggregation will be poor in textureless areas. To overcome this we compute the texture measure L_p (eq.5.5) to balance the influence of eq.5.6 and eq.5.7, which results in eq.5.9.

$$L_p = \left\{ e^{-\frac{T_p}{\tau_w}} \,\middle|\, T_p = \frac{1}{Z} \sum_{q \in N(p)} e^{-\frac{|I_p - I_q|}{\sigma_r}} h(p) \right\} \tag{5.5}$$

$$AW_p(D_p) = \frac{1}{Z} \sum_{q \in N(p)} e^{-\frac{|I_p - I_q|}{\sigma_r}} c_q(D_p) \tag{5.6}$$

$$CW_p(D_p) = \frac{1}{Z_v} \sum_{q \in N_v(p)} e^{-\frac{|I_p - I_q|}{\sigma_r}} CW_q^h(D_p) \tag{5.7}$$

$$CW_q^h(D_p) = \frac{1}{Z_h} \sum_{s \in N_h(q)} e^{-\frac{|I_s - I_q|}{\sigma_r}} c_s(D_p) \tag{5.8}$$

$$C_p(D_p) = L_p \cdot AW_p(D_p) + (1 - L_p) \cdot CW_p(D_p) \tag{5.9}$$

where Z, Z_h, Z_v are normalisation constants such that the weights in the neighbourhood add up to one, $h(p)$ entropy filter, τ_w acts as a scale to regulate the influence of the measure T_p, $|I_p - I_q|$ is the $L1$ distance in *RGB* space with scaling factor σ_r, $N(p)$ is the neighbourhood around p ($n \times n$ window), $N_h(p)$ and $N_v(q)$ are the horizontal ($1 \times n$) and vertical ($n \times 1$) neighbourhoods around p and q, $c_s(D_p)$ is the raw pixel similarity cost (eq.5.10). Eq.5.5 is the adaptively filtered version of $h(p)$, which is done to clean the noisy measure, and better balance AW_p and CW_p close to edges.

The texture measure eq.5.5 is computed per pixel, and depends on the aggregated result of an entropy filter $h(p)$. An entropy filter is used, because a textureless region is constant (or almost constant) and therefore its entropy is either zero or close to zero, and this provided an easy way to identify such regions. The entropy filter is computed in a 5×5 window per pixel, which is used to prevent small high entropy regions from being lost by using a larger window filter. However, a small filter results in a noisy entropy measure that may not be correctly estimated around image intensity edges, which motivates the use of [112] as a joint bilateral filter that propagates the entropy measure and at the same time corrects it around intensity edges.

The aggregation algorithm described so far can use any type of pixel. The proposed approach uses a combined pixel similarity function that balances the benefits of a pixel-wise measure and a block matching cost. Our pixel cost is given by:

$$c_p(D_p) = \alpha c_p^1(D_p(p)) + c_p^2(D_p(p)) \tag{5.10}$$

$$c_p^1(D_p) = min(|\nabla I_p^1 - \nabla I_{p+D_p}^2|, \tau_{grad}) \tag{5.11}$$

$$c_p^2(D_p) = min(\chi(I^1, I^2, p, D_p), \tau_{cen}) \tag{5.12}$$

where:

- I^1 is the reference image, and I^2 is the target image ($left \rightarrow right$ or $right \rightarrow left$).
- $c_p^1(D_p)$ is the truncated absolute differences of gradients.
- $c_p^2(D_p)$ is the Truncated Hamming distance of the census transforms [115].

- χ computes the census transform at p and displacement D_p and Hamming distance.

- α balances the pixel-wise cost influence.

The gradient and census transform are used because they are known to be capable of handling small radiometric differences such as different light conditions in each image pair. These two measures ignore the colour information as they are computed using the grey level intensity, but colour is used during aggregation to compensate for this.

As noted previously performing aggregation using large windows may not be appropriate for all of the image (e.g. because of the centre pixel bias, and plane bias), and using small windows gives noisy results. The core idea in all aggregation algorithms is to add up pixels within the same window in the hope of cleaning up the initial raw cost. All pixels in the selected window are assumed to be from the same surface or are weighted according to colour similarity. The assumption of pixels being in the same surface is true for large smooth surfaces, but it quickly breaks for small surfaces inside large window sizes. This suggests that adapting the window size taking into account the underlying 3D surface would improve the aggregation results.

The proposed approach to adapting the window size (using the underlying 3D surface) is to choose from two possible window sizes based on a local disparity consistency measure W_p, which describes the behaviour of an initial disparity map gradient around a region. For the sake of simplicity a region is a superpixel segment obtained using SLIC [1]. The algorithm to estimate the window size consists of the following steps:

1. Measure the initial disparity map gradient at the pixels along the segment perimeter using eq.5.13.

2. Compute the segment difference with respect to neighbouring segments using eq.5.14. The median disparity of segments is used to compute the difference.

3. Propagate the previous disparity consistency measure per pixel using eq.5.15. This is done to reduce the effect of noise from the initial disparity map.

4. Select the window size at each pixel p is computed using $n = \Omega(p)$ in eq.5.16, where $\omega_1 < \omega_2$.

$$\hat{w_p} = \frac{1}{K_w|\hat{N}(p)|} \sum_{q \in \hat{N}(p)} \left[|\tilde{d_p} - \tilde{d_q}| \geq \frac{2}{5} K_w \right] \tag{5.13}$$

$$w_p = \begin{cases} \frac{1}{K_w|\hat{N}(p)|} \sum_{q \in \hat{N}(p)} min(|\tilde{d_p} - \tilde{d_q}|, K_w) & : \hat{w_p} > \tau'_w \\ 0 & : \text{otherwise} \end{cases} \tag{5.14}$$

$$W_p = \left\{ \frac{1}{Z} \sum_{q \in N(p)} e^{-\frac{|I_p - I_q|}{\sigma_r}} w_p \,\middle|\, Z = \sum_{q \in N(p)} e^{-\frac{|I_p - I_q|}{\sigma_r}} \right\} \tag{5.15}$$

$$\Omega(p) = \begin{cases} \omega_1 & : W_p > \tau_{nn} \text{ and } T_p > \tau_h \\ \omega_2 & : \text{otherwise} \end{cases} \tag{5.16}$$

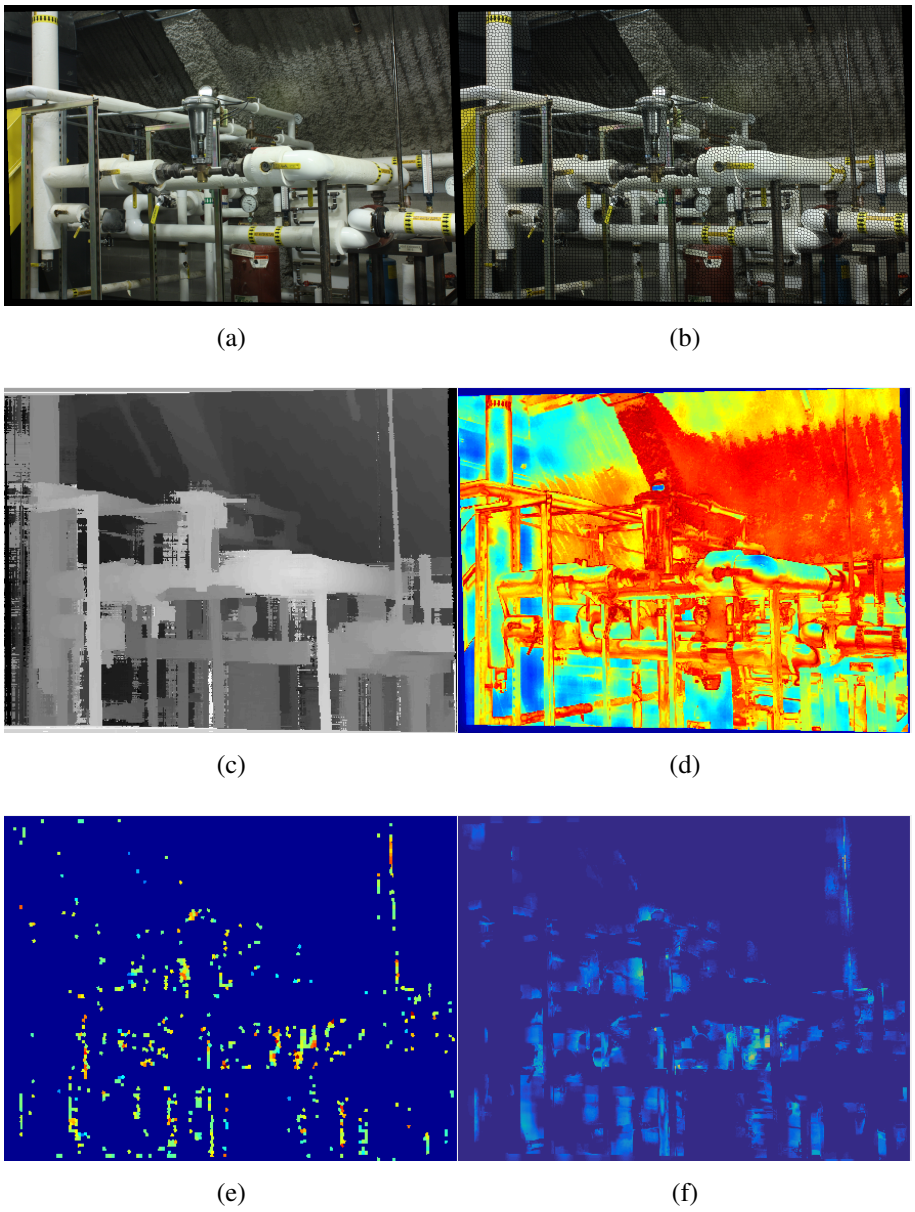
where, $\tilde{d_p}$ and $\tilde{d_q}$ are the median disparities (to compensate for noise in the initial estimate) of the segments where pixels p and q come from. $\hat{N}(p)$ is the perimeter of the segment where p comes from. Notice that eq.5.14 computes the same value for all pixels in a segment, whereas eq.5.15 does it independently per pixel by aggregating neighbouring values based on intensity similarity. T_p comes from eq.5.5. τ_{nn} is dynamically computed from W_p using the Otsu threshold algorithm. Eq.5.13 measures the number of pixels that can be considered an edge because their disparity difference is above a threshold.

The window size estimation algorithm described above makes two assumptions:

- An initial disparity map is available.
- Only two window sizes are possible.

It is worth mentioning that if no initial disparity map is available the proposed algorithm just behaves as any other aggregation algorithm using a fixed window size. Using only two windows sizes is meant to reduce the centre pixel and plane biases, which have to be large enough to avoid noisy results, but one of the them must be smaller to avoid losing small surfaces.

Fig.5.7 shows the intermediate stages used to compute the window size. T_p estimation (fig.5.7d, red high value, blue low value), which is the adaptively filtered version of the entropy filter. w_p is the disparity gradient measure per segment (fig.5.7e, red high value, blue low value). W_p is the disparity gradient measure per pixel (fig.5.7f, red high value, blue low value), which is the adaptively filtered version of w_p using eq.5.14with fig.5.7a as reference image.



(a) (b)

(c) (d)

(e) (f)

Figure 5.7: Window size estimation stages: (a) Reference image; (b) Reference image segments; (c) Initial disparity; (d) T_p; (e) w_p; (f) W_p.

5.6 Smoothness term

The assumption made by the proposed edge model is that possible depth discontinuities are allowed small disparity variations (i.e. edges are have a lower penalty), while areas with a similar intensity are allowed to have larger changes, which can result in a value for the smoothness term (i.e. encourages reduction of overgrown edges), by contrast in [6, 88] the maximum variation is constant.

$$V_{pq}(D_p, D_q) = \begin{cases} w_{pq}\lambda\omega(D_p, D_q, K2) & : |F_p - F_q| < \tau_{diff} \\ w_{pq}\lambda\omega(D_p, D_q, K1) & : \text{otherwise} \end{cases} \tag{5.17}$$

$$w_{pq} = \left\{ \frac{1}{Z_n} e^{-|I_{ms}(p) - I_{ms}(q)|} \middle| \; Z_n = \sum_{q \in N(p)} e^{-|I_{ms}(p) - I_{ms}(q)|} \right\} \tag{5.18}$$

where $K1 < K2$, I_{ms} is a grey level image after applying the quick shift [94, 93] segmentation to handle noise in the image, w_{pq} and λ are weights, F_p and F_q come from a local cue map F (eq.4.11 for LCU), and τ_{diff} is a threshold, which means the growth limit of the smoothness term adapts based on the local cue map F, and $\omega(D_p, D_q, K)$ is given by:

$$\omega(D_p, D_q, K) = min(|D_p(p) - D_q(p)| + |D_q(q) - D_p(q)|, K) \tag{5.19}$$

Unlike the simple L_1 difference from chapter 4 eq.5.19 can handle slanted and curved surfaces, but it has a higher computational complexity $O(n^2)$ as it cannot be computed using the distance transform.

5.7 Hypothesis generation and propagation

The basic idea of 3D plane labelling is that each pixel has associated a set of plane hypotheses from which a single optimal plane assignment (D) is chosen. We use TRW-S to find the optimal 3D plane labelling. The proposed approach relies on a hypothesis generation algorithm that gives each pixel p a new set of 3D plane hypothesis $GH_p = D_p \cup H_p \cup DS_p \cup V_p$, where:

- D_p is the current solution.
- H_p is generated from pixel coordinates using (r-sampling, sec.5.7.1).
- DS_p is generated from the pixels that belong to the same segment[1] (P-sampling, sec.5.7.2).
- V_p is generated from pixels coming from another view (V-sampling, sec.5.7.3).

Generating this set of potential disparity planes at each pixel allows:

[1] A segment will be a set of pixels that are grouped using some logical criterion e.g. spatial distance and colour similarity.

1. Pixels could keep their current plane assignment D_p.
2. Planes are propagated from close and distant pixels via r-sampling.
3. Disparity planes covering an entire segment are propagated to distant pixels via P-sampling.
4. Views propagate their planes via V-sampling.

Note that hypothesis generation is done from a single initial disparity plane assignment D for both left and right views, and the hypothesis generation is effectively acting as a propagation scheduler once inference is done.

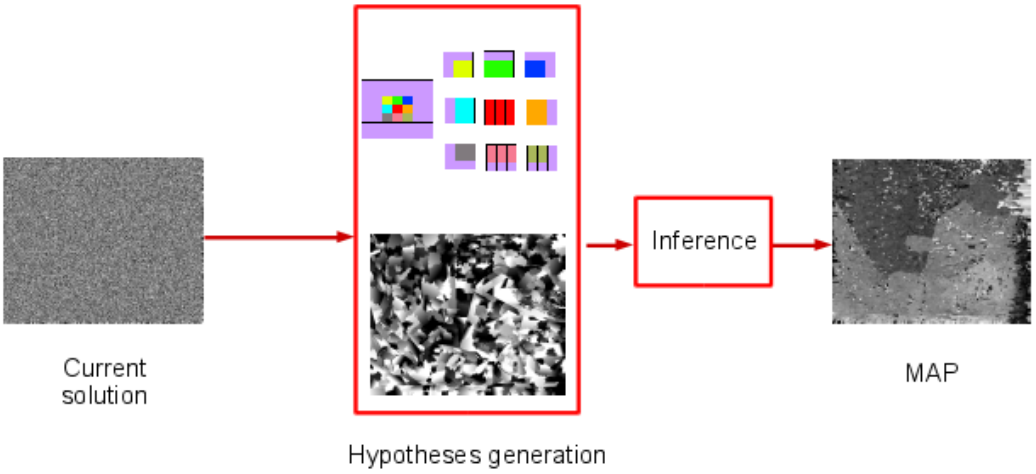


Figure 5.8: Proposed algorithm pipeline, see text for details.

Fig.5.8 shows an example of our algorithm pipeline, i.e. initialisation, hypotheses generation in the centre red box (pixel hypotheses upper part, segment hypotheses lower part), and inference. Our approach can work with either a random initialisation or a pre-computed disparity map, and thus it behaves like a combination of *DPI* and *FPI* algorithms.

5.7.1 r-sampling

The proposed r-sampling is an algorithm that generates hypotheses that cover a region while at the same time acts like a propagation scheduler. Our approach follows the two common assumptions made by *DPI* algorithms that propagate planes:

- Neighbouring pixels are likely to have the same plane [9, 6].

- Planes are shared within a certain area [88].

The purpose of r-sampling is to simulate a propagation scheduler like that from [9, 6], but without having to do it sequentially. Additionally, the generated hypotheses are shared in a region, similar to [88], but also include a few planes from distant regions. The r-sampling strategy works by transforming the coordinates (x,y) of the sampled pixel in the following way:

$$(X,Y) = (divs * \lfloor (x+i)/divs \rfloor + r + i, divs * \lfloor (y+j)/divs \rfloor + r + j) \tag{5.20}$$

where:

- $x,y \in \mathbb{Z}$, which are the current pixel coordinates.
- $X,Y \in \mathbb{Z}$, which are the transformed pixel coordinates.
- $(j,i) \in [-r,r] \times [-r,r]$ with $i,j \in \mathbb{Z}$ (i.e. the elements of the window).
- $divs = 2r+1$, which is the window size.
- $r \in \mathbb{N}$ and $r > 0$, is a parameter that controls the window sampling size width $(2r+1)$.

Using eq.5.20 each pixel p at (x,y) is r-sampled using all values (i,j), which results in the plane being sampled from pixel q at (X,Y):

$$\hat{D}_p^{ij} = D_q^{XY} \tag{5.21}$$

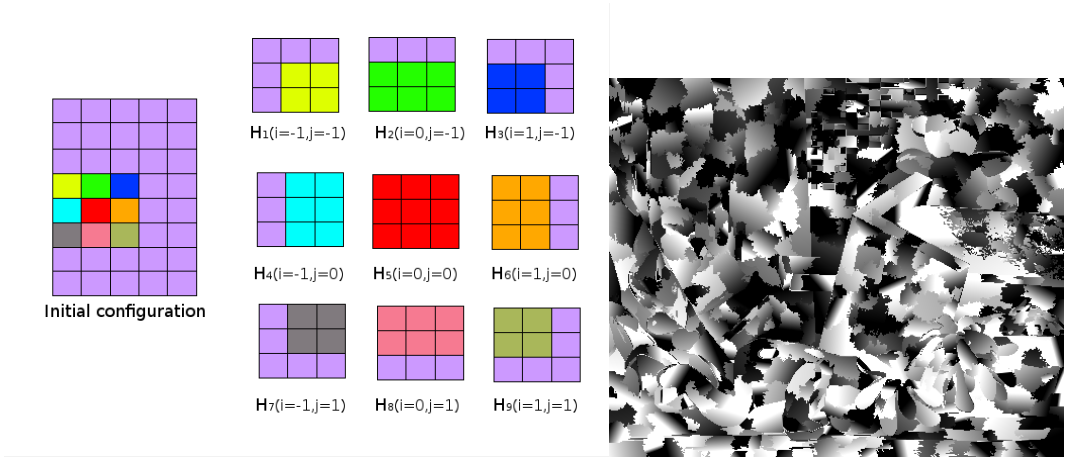
This sampling is repeated for each $(i,j) \in [-r,r] \times [-r,r]$ generated by a window centred at p, which means each pixel has a set of hypotheses:

$$H_p = \bigcup_{i=-r}^{r} \bigcup_{j=-r}^{r} \hat{D}_p^{ij} \tag{5.22}$$

This sampling strategy allows one to include the planes from all pixel neighbours in the $divs \times divs$ window centred at p and also neighbours that come from a different window of $divs \times divs$. Fig.5.9 shows how a 3×3 window (i.e $r = 1$) would generate nine hypotheses, where each coloured box is a different plane in the initial configuration shown fig.5.9. For illustration purposes we will centre our attention around the "red plane" and its eight neighbours. $H_1 ... H_n$ represent the hypotheses generated by

using i and j according to eq.5.22. In particular fig.5.9 demonstrates how each of the non-purple planes are transformed into a hypothesis. Also note that the planes from the purple pixels are used as well, although they are not necessarily the same because they come from neighbouring sampling windows. This is meant to allow the propagation of distant neighbours. This hypothesis generation strategy is similar to that of [88], but ours does not impose a sub-modular requirement to the pairwise interaction in the generated hypothesis.

(a)



(b) (c)

Figure 5.9: Hypothesis generation: (a) Teddy reference image; (b) r-sampling (see sec.5.7.1); (c) $Ds_p = D\phi(p)$ (see sec.5.7.2).

Finally, the plane hypotheses generated by r-sampling at pixel p are given to several neighbouring pixels (see fig.5.9), which reduces the chances of the inference algorithm getting stuck at a local minimum, because of the following reasons:

1. A hypothesis plane extending over a region when mapped to the other view creates less violations of the local uniqueness term, as each pixel in the planar

region is mapped a single location!

2. A hypothesis plane extending over a region results in a lower smoothness cost (i.e. neighbours have the same plane), which prevents the unary term from having too much influence.

3. A lower smoothness cost allows each pixel to have a random disparity plane initialisation, and prevents the smoothness term from having too much influence.

5.7.2 *P*-sampling

The proposed 3D labelling algorithm exploits the assumption commonly used in stereo matching that pixels with similar intensity have a locally similar disparity value or disparity plane. In particular, intensity/colour pixels are grouped as super-pixels or segments using SLIC [1], and then initialised with a plane. A disparity plane hypothesis Ds_p is assigned to a segment s_p by randomly selecting a plane from pixels that belong to the same segment:

$$Ds_p = D\phi(p) \tag{5.23}$$

where Ds_p denotes a plane at pixel p that was assigned by function $D\phi(p)$ which randomly selects a plane such that it belongs to the same segment as pixel p. In this way only one hypothesis per pixel is generated, although there may be other planes in the same segment. This assumes that one of planes selected may be correct without the need to fit one plane to the entire segment. Although each Ds_p covers a region of several pixels, it does not exploit neighbouring segment planes that may provide a better hypothesis. For this reason additional hypotheses are generated from the segments that surround s_p using eq.5.24 as shown in fig.5.10.

$$DS_p = \left\{ Ds_p \cup (\bigcup_{i=1}^{P} Ds_{qi}) \,\middle|\, \Theta(s_{q1}, s_p) < \ldots < \Theta(s_{qP}, s_p) \right\} \tag{5.24}$$

where $s_{qi} \in N(s_p)$ and $\Theta(s_{qi}, s_p)$ is the function that evaluates the polar angle with respect to the 2D centroid of s_p. The segments are ordered because they have irregular shapes, and therefore some criterion is needed to have a similar concept of neighbourhood as in the case of regular pixels, which becomes a propagation scheduler for segments. The process to sample the neighbouring segments is broken down in the following steps:

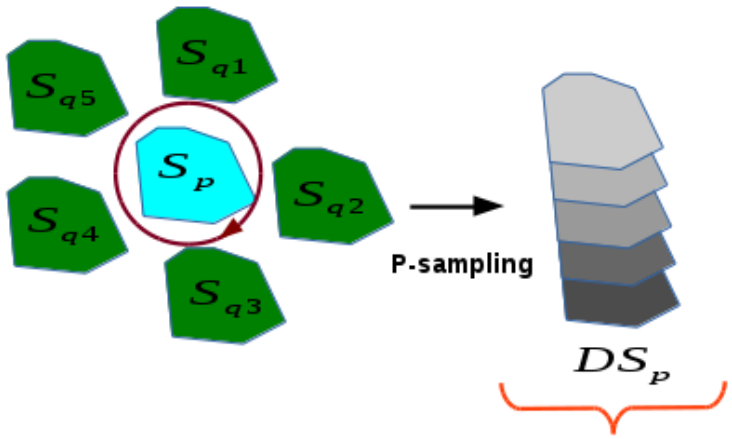


Figure 5.10: P-sampling example. The neighbouring segments of S_p are ordered clockwise, and the first P are selected (DS_p).

1. Compute the centroid of each s_p.

2. Sort all the neighbouring s_q centroids their polar angle with respect to s_p.

3. Select P segments from the ordered centroids with respect to their polar angle (ascending order).

The hypothesis generation algorithm described above has the following characteristics:

1. Large regions can be labelled using the segment disparity plane during the inference stage.

2. The plane hypothesis and later labelling inference is done per pixel, which helps to overcome segmentation errors.

3. Distant planes can be propagated (depending on the segment size) by using neighbouring segments to generate hypothesis, assuming that segments are large enough.

The P-sampling described in this section is only a hypothesis generation algorithm and must not be confused with a 3D labelling algorithm.

5.7.3 V-sampling

The hypotheses H_p and DS_p so far are generated from only one view. Since left and right sets of disparity planes are computed it is possible that one of them contains planes which could be used to generate a better set of hypothesis; for instance one of the solutions could have correct disparity in a textureless area, while the other does not. An additional set of hypotheses is computed by mapping (using the known camera

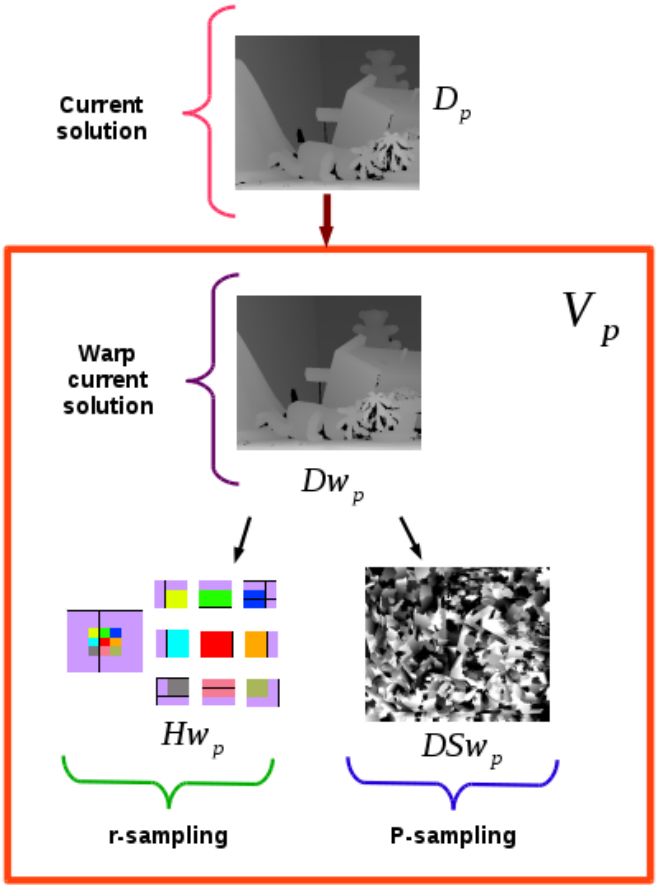


Figure 5.11: V-sampling diagram.

calibration) the planes D_p from the other view to the current image, then r-sampling and P-sampling is performed to obtain the new set of hypotheses as shown in fig.5.11. This process (shown in fig.5.11) will be referred to as V-sampling and can be expressed as:

$$V_p = Dw_p \cup Hw_p \cup DSw_p \tag{5.25}$$

where Dw_p (warp current solution in fig.5.11) is obtained by mapping the current solution D_p of the other view, Hw_p (r-sampling in fig.5.11) and DSw_p (P-sampling in

fig.5.11) are obtained from Dw_p to generate the new set of hypothesis. Note that this doubles the number of hypotheses. For instance: assume two views are used, then for $r = 2$ and $P = 6$ the number of hypotheses generated would be 32 for one view, but after doing V-sampling the number of hypotheses would be 64. Note that [9] follows a similar approach (i.e. mapping the planes from one view to the other), but only generates one hypothesis.

5.8 Refinement

The algorithms from the previous section generate hypotheses and inference (see fig.5.8) is used to find a unique 3D disparity plane assignment for each pixel (i.e D_p). However, the estimated solution is the result of propagating the planes from an initial disparity assignment (random or pre-computed), which may still have incorrect assignments. In order to improve the current solution it is refined by generating a new set of hypotheses $Gref_p = D_p \cup Href_p \cup DSref_p$ with perturbations[9] added to the plane parameters (normal vector $\hat{n}_p$ and disparity d_p), and then doing inference once again.

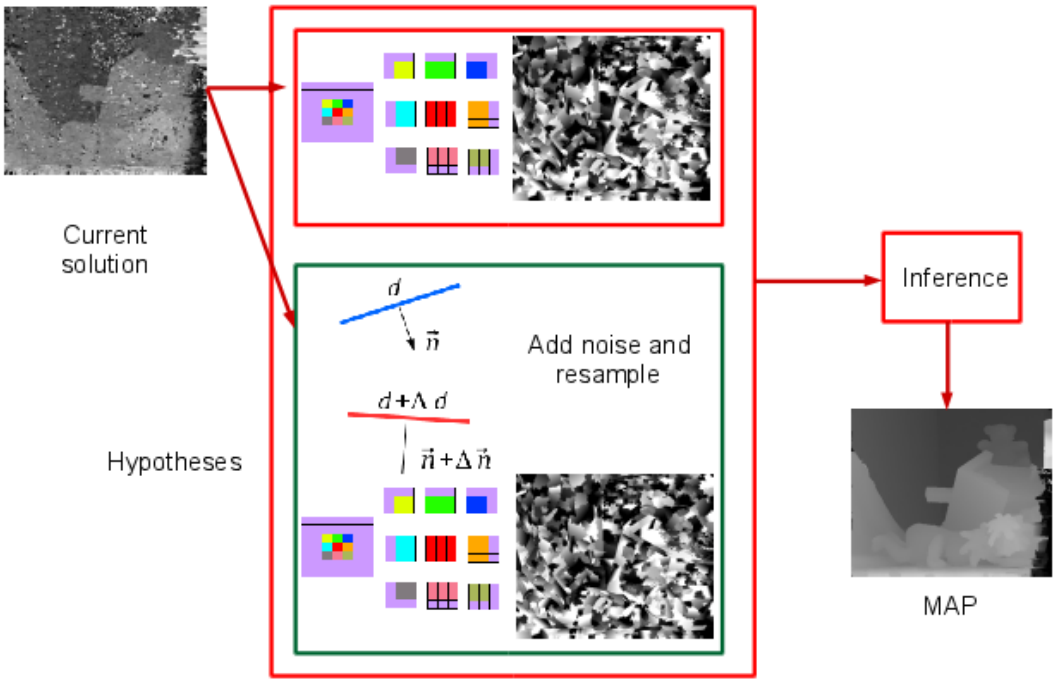


Figure 5.12: Proposed refinement algorithm pipeline. Centre upper red box: r-sampling and P-sampling of the current solution; Centre lower greed box: r-sampling and P-sampling of the current solution with perturbations added (see sec.5.8.1 and sec.5.8.2).

Fig.5.12 shows how the refinement algorithm works, which can be summarised in the following way:

1. Given the current disparity map D_{curr}, $\Delta n_1 = \kappa$, $\Delta d_1 = Sr \times \kappa$ as input (see sec.5.8.1 and sec.5.8.2).
2. For $t = 1$ to s steps do:
   (a) compute pixel hypothesis re-sampling ($Href_p$), see sec.5.8.1.
   (b) compute segment hypothesis re-sampling ($DSref_p$), see sec.5.8.2.
   (c) compute $Gref_p = D_p \cup Href_p \cup DSref_p$ and do inference to obtain a new solution $\hat{D}_{curr}$.
   (d) set $D_{curr} = \hat{D}_{curr}$, $\Delta n_t = \Delta n_{t-1}/2$ and $\Delta d_t = \Delta d_{t-1}/2$.

This means that for each pixel in the disparity map three situations can happen:

1. it gets a new plane propagated from a neighbour.
2. it gets a plane that was refined and propagated from a neighbour.
3. its plane stays the same.

The algorithm described above performs propagation simultaneously for all pixels during the hypothesis refinement process, which happens because of the way hypotheses are generated. Also note that V-sampling is not used. To the best of our knowledge no other algorithm performs simultaneous propagation and refinement, which could have the potential to reduce the space needed to store hypotheses. The refinement process doubles the number of hypotheses. For instance: if r-sampling ($r = 2$) and P-sampling($P = 6$) generate 32 hypotheses, then refinement would add 32 additional hypotheses (with perturbations added).

5.8.1 Pixel hypothesis re-sampling

In order to overcome local minima of the current disparity plane assignment, we obtain a refined version of r-sampling. This is accomplished by adding to each pixel a random perturbation selected from $[-\Delta n_t, \Delta n_t]$ (uniform distribution) and $[-\Delta d_t, \Delta d_t]$ (uniform distribution) which are initially set to $\Delta n_1 = \kappa$ (the new normal vector has to be normalised) and $\Delta d_1 = Sr \times \kappa$, ($Sr$ search range in disparity units), with $\kappa = 0.2$ (note that

this is similar to the simulated annealing temperature update) as recommended in [6]. This re-sampling process is repeated several times. After each step Δn_t and Δd_t are updated by setting them to $\Delta n_t = \Delta n_{t-1}/2$ and $\Delta d_t = \Delta d_{t-1}/2$. This process generates a new set of hypotheses $Href_p = H_p \cup Hn_p$ where Hn_p is generated using r-sampling from D_p after perturbations have been added.

The pixel hypothesis re-sampling acts like particle filtering by generating new hypotheses around the current solution, and adding a few extra hypothesis from distant places. Adding a random perturbation directly to each pixel could result in a larger smoothness cost, which could potentially make the algorithm get stuck at a local minima, but this is avoided because r-sampling decomposes a single hypothesis into multiple hypotheses covering a single area, which might lead to a lower smoothness cost.

5.8.2 Segment hypothesis re-sampling

In the same way that each pixel is re-sampled, refined versions of the disparity plane image segments are obtained by creating variants of the current set of disparity planes per image segment. First, for each segment in the current image random disparity planes (DSr_p) are generated using P-sampling and then noise is added. This generates a new set of hypotheses $DSu_p = \Psi(DSr_p \cup DSrn_p)$, where DSr_p was generated via P-sampling, $DSrn_p$ is the perturbed version of DSr_p, and Ψ is a function that takes each pair of planes DSr_p^i and $DSrn_p^i$ and returns the one whose cost locally minimises eq.5.1, with respect to the current solution. This leaves P plane hypotheses per segment.

Another set of hypotheses DSU_p (jumping planes) is then generated (for each segment S_p of the current solution), in which each segment SU_p has a number from $[1...n]$. These numbers are used as multipliers to control the random interval $[-\Delta d_t, \Delta d_t]$, such that $\Delta^i d_t = i \times \Delta d_t$. Thus each element of DSU_p is the $\Delta^i d_t$ perturbed version of a plane generated by $D\phi(p)$ to allow large jumps of disparity that can change an entire segment. The large jumps in disparity are needed, not only to improve the current solution, but also to mimic at a local scale the behaviour of fronto-parallel disparity estimation that explores all disparities. Finally, the planes with added perturbations and jumping planes are combined to produce a new set of hypotheses $DSref_p = DSu_p \cup DSU_p$, which is referred to as the per pixel segment re-sampled hypotheses.

5.9 Overall disparity estimation process

The disparity estimation process can now be summarised as follows:

1. Initialise solution $Dleft$ and $Dright$ to either a random plane per pixel or initialise from a pre-computed disparity map.

2. For $t = 1$ to s steps do:

   (a) Generate hypotheses $GHleft_p$ and $GHright_p$, see sec.5.7.

   (b) Do inference and compute new solutions $\hat{D}left$ and $\hat{D}right$.

   (c) Refine the new solutions $\hat{D}left$ and $\hat{D}right$, see sec.5.8.

   (d) set $Dleft = \hat{D}left$ and $Dright = \hat{D}right$.

Random initialisation is done by selecting a random uniform disparity. The normal $\hat{n}$ is selected from the uniform distribution of a half unit sphere. Δd and $\Delta\hat{n}$ are selected in the same way.

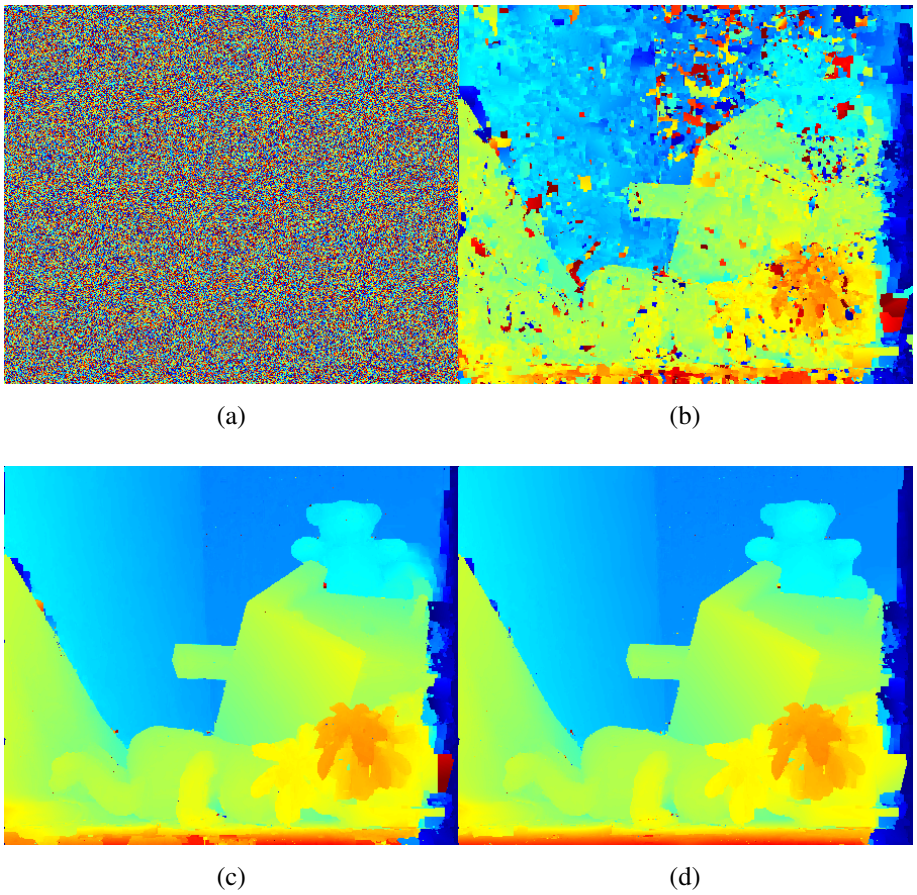
Fig.5.13 shows intermediate stages for the teddy image, using the proposed algorithm with random initialisation. It can be observed that our approach is effectively propagating the planes and also minimising eq.5.1 as shown in fig.5.14. Fig.5.13 also shows that after the first propagation the outline of the underlying 3D surface can already be seen.

The proposed algorithm can also be initialised with a pre-computed disparity map (i.e. using any other algorithm) used to adapt the aggregation window size. The pre-computed solution is also used to compute the disparity plane normal vectors, which are computed at five scales averaging, and creating a plane per pixel by estimating the parameters of eq.5.2 using the normal vectors and current disparity. Finally, this algorithm is extended to multi-scale:

1. Pre-compute disparity maps $Dleft_{pre}$ and $Dright_{pre}$ (e.g. using *SGM* or random initialisation).

2. Estimate window size from $Dleft_{pre}$ and $Dright_{pre}$ (if disparity map was pre-computed).

3. Compute initial solution $Dleft$ and $Dright$ by downscaling $Dleft_{pre}$ and $Dright_{pre}$ to one quarter size and solving.



(a) (b)

(c) (d)

Figure 5.13: Proposed algorithm with random initialisation ($\tau_{grad} = 5/255$, $\tau_{cen} = 5/25$, and no multi-scale), intermediate and final results: (a) Initial random configuration; (b) First propagation, $E = 123,257.02$; (c) First Refinement, $E = 87,134.61$; (d) Final Result, $E = 82,334.98$.

4. Compute updated solution at half-size by initialising with current $Dleft$ and $Dright$.

5. Compute updated solution at full-size by combining $Dleft_{pre}$ and $Dright_{pre}$ with current $Dleft$ and $Dright$. i.e compute $GHpre_p \cup GH_p$ for left and right, then do inference to update current solutions, and update $maxD$.

6. Compute updated solution at full-size by initialising with current $Dleft$ and $Dright$.

7. Compute occlusion detection and post-processing.

It is worth noting that a scaling factor of 1.5 was also tested, but it resulted in a higher computational time, and little improvement. To detect spurious disparities

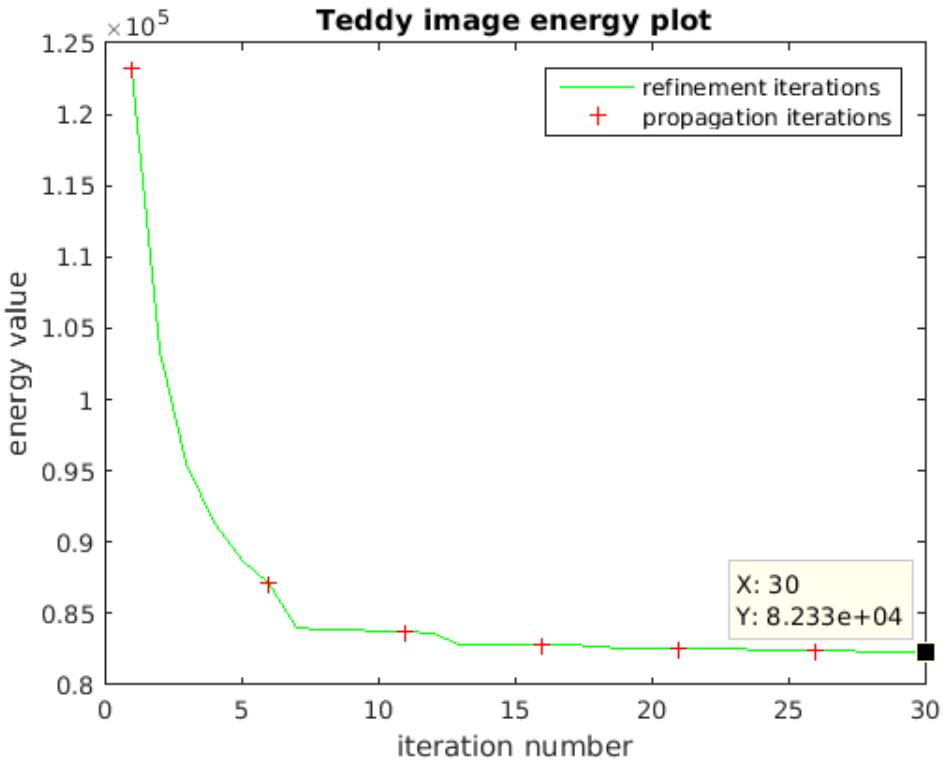


Figure 5.14: Energy plot for Teddy image (random initialisation, and no multi-scale).

the normalised disparity histogram of each disparity present in both the left and right disparity maps is computed only in the initial search range $[minD, maxD]$ (i.e. the valid disparities). An n-point (n is 18% of the initial disparity search range) Parzen window is applied to smooth the normalised histogram. Finally, a search finds the lowest ($min\hat{D}$) and largest ($max\hat{D}$) disparities that are above a threshold τ_{maxD} and $\tau_{minD} = \frac{2}{3}\tau_{maxD}$. It is possible that $min\hat{D}$ is too large and $max\hat{D}$ too small, so they are adjusted $min\hat{D} = max(minD, min\hat{D} - \Delta minD)$, and $max\hat{D} = min(maxD, max\hat{D} + \Delta maxD)$. The estimated bounds on the disparities are used to eliminate (in post-processing) or penalise (using eq.5.4) unrealistic disparities. $max\hat{D}$ is used during the inference process, and $min\hat{D}$ during the post-processing stage.

The post-processing eliminates spurious disparities, which is done by computing occluded, and mismatched areas as in [97]. Then masks containing non-occluded areas, and depth edges[2] are eroded once and marked as occluded to reduce the fattening effect. Then $minD$ is updated (as described above), and disparities lower than $minD$ are marked as occluded. The occluded areas are then filled in using simple background interpolation and filtered by applying a weighted median filter. The non-occluded areas

[2] detected using Canny edge detector.

are NOT post-processed.

5.10 Pre-computing the disparity map initialisation

As noted in sec.5.9 our algorithm can use a precomputed disparity map as initialisation. We use SGM to compute the initial disparity map. The SGM implementation used to minimise eq.5.1 is due to [97] (only the optimisation part is used), which was modified to use less memory due to hardware constraints. The similarity function C_p used to pre-compute the disparity map initialisation makes no use of the content aware adaptive windows to keep fine details, but instead uses eq.5.26. The smoothness term used (the same in [97]) is defined by eq.5.30, which is used to align to depth and intensity discontinuities.

$$C_p(D_p) = \alpha c_p^1(D_p(p)) + AW_p(D_p) \tag{5.26}$$

$$AW_p(D_p) = \left\{ \frac{1}{Z} \sum_{q \in N(p)} e^{-\frac{|I_p - I_q|}{\sigma_r}} c_q^2(D_p) \,\middle|\, Z = \sum_{q \in N(p)} e^{-\frac{|I_p - I_q|}{\sigma_r}} \right\} \tag{5.27}$$

$$c_p^1(D_p) = min(|\nabla I_p^1 - \nabla I_{p+D_p}^2|, \tau_{grad}) \tag{5.28}$$

$$c_q^2(D_p) = min(\chi(I^1, I^2, q, D_q), \tau_{cen}) \tag{5.29}$$

$$V_{pq}(D_p, D_q) = P1 \cdot \{|D_p - D_q| = 1\} + P2 \cdot \{|D_p - D_q| > 1\} \tag{5.30}$$

where $P1$ and $P2$ are set as follows (parameters given bellow):

$$(P1, P2) = \begin{cases} (sgm_{P1}, sgm_{P2}) & : |I_p^1 - I_q^1| < sgm_D \text{ and } |I_{p+D_p}^2 - I_{q+D_p}^2| < sgm_D \\ (\frac{sgm_{P1}}{sgm_{Q2}}, \frac{sgm_{P2}}{sgm_{Q2}}) & : |I_p^1 - I_q^1| \geq sgm_D \text{ and } |I_{p+D_p}^2 - I_{q+D_p}^2| \geq sgm_D \\ (\frac{sgm_{P1}}{sgm_{Q1}}, \frac{sgm_{P2}}{sgm_{Q1}}) & : \text{otherwise} \end{cases} \tag{5.31}$$

In eq.5.26, I^1 is the reference image, and I^2 is the target image, $c_p^1(D_p)$ is the truncated absolute differences (using the Birchfield dissimilarity measure) of gradients, $c_p^2(D_p)$ is the Truncated Hamming distance of the census transform [115]), χ computes the census transform at p and displacement D_p and Hamming distance, α balances the pixel-wise cost influence. Note that $D_p(p)$ and $D_p(q)$ are the integer disparities at pixels p and q. Note that $c_p^1(D_p)$ is not aggregated to avoid losing small details. In eq.5.30, $P1$ and $P2$ are used to enforce a Potts like edge model.

The parameters used in all experiments are set as follows: $\alpha = 30$, $\tau_{grad} = 5/255$, $\tau_{cen} = 5/25$, $sgm_{P1} = 1.32$, $sgm_{P2} = 24.25$, $sgm_{Q1} = 3$, $sgm_{Q2} = 2$, $sgm_D = 0.08$. These parameters are kept constant for all experiments using Middlebury, KITTI 2015, and KITTI 2012 data sets.

5.11 Proposed approach evaluation

The approach presented in this chapter was validated in three stages. 1) The proposed content aware adaptive windows are compared to the traditional approach [112]. 2) We evaluate the tolerance to noise and performance of $r-$sampling. 3) Then more extensive experiments are performed (see sec.5.12) to validate our approach on the three commonly used data sets: The new Middlebury (15 images) data set [81], KITTI 2015[66] and 2012[24] (up to 200 images).

All experimental results were carried out using the multi-scale approach with pre-computed initialisation. The experiments are performed using the following parameters for all scales: aggregation window sizes of $\omega_1 = 41 \times 41$, $\omega_2 = 25 \times 25$, $\tau_{grad} = 3/255$, $\tau_{cen} = 9/25$, $\sigma_r = 10/255$, $\sigma_d = 0.5$, $\tau_w = 2.5$, $\tau_{diff} = 0.07$, $\tau_{unique} = 0.01$, $\alpha = 30$, $K_1 = 1$, $K_2 = 6$, $r = 2$, $P = 6$, $\tau_h = 2$, $\tau'_w = 0.5$, $K_w = 8$, see appendix A for parameter exploration. For quarter and half size images $\lambda = 0.09$, and $\lambda = 0.18$ for full size images. The disparity estimation is iterated 4 times at quarter size, 1 time at half-size and 4 times at full size. The refinement iterations are set to 5 times at each scale. These parameters were obtained by using the Middlebury training data, and every fifth image from KITTI 2015 and 2012 training data.The only parameters that are set differently are $\Delta minD$ (10 for Middlebury and 5 for KITTI), $\Delta maxD$ (60 for Middlebury and 20 for KITTI) and τ_{maxD} (0.0028 for Middlebury and 0.0003 for KITTI). The value $\Delta maxD$ is set differently due to the number of disparities present in the KITTI (256) vs. Middlebury (400) data sets. Additional experiments can be found in appendix B and appendix C.

To test the performance of the adaptive window aggregation algorithm three competing functions are compared (tab.5.1): AW is the traditional approach (eq.5.6) with the enabled out of range term , but no uniqueness term. $AW + CW$ is the adaptive window (eq.5.9) with the enabled out of range term, but no uniqueness term. $AW + CW + U$ is the proposed adaptive window with the out of range and local uniqueness terms (eq.5.9), which clearly show improvement over the popular adaptive windows[112] commonly used in DPI algorithms. To further show the improvement of our content

Function	%bad noc	avg. error	rms
AW	13.62	2.00	7.62
$AW + CW$	13.58	1.98	7.67
$AW + CW + U$	**12.95**	**1.82**	**7.28**

Table 5.1: Comparative table of the proposed content aware function vs adaptive windows. Evaluation done with the Middlebury training data set at half size in non occluded areas before post-processing.

aware adaptive windows three error measures and their average are given: percentage of bad pixels larger than one disparity (tab.5.2), average disparity error (tab.5.3), and rms of estimated disparity (tab.5.4). These evaluations are done using the Middlebury data set training images at half resolution.

Image	AW	$AW + CW$	$AW + CW + U$
Adirondack	4.18	4.23	**3.77**
ArtL	7.70	7.82	**7.55**
Jadeplant	14.19	12.41	**12.03**
Motorcycle	**6.56**	6.89	6.78
MotorcycleE	7.29	7.40	**7.18**
Piano	14.44	14.42	**14.20**
PianoL	27.38	28.39	**27.02**
Pipes	9.64	9.03	**8.74**
Playroom	17.94	17.88	**16.61**
Playtable	11.81	12.01	**10.42**
PlaytableP	7.81	8.48	**6.74**
Recycle	12.39	11.96	**11.25**
Shelves	36.87	36.90	**36.31**
Teddy	4.00	4.07	**3.83**
Vintage	22.08	21.85	**21.75**
Average	13.62	13.58	**12.95**

Table 5.2: Comparative table: % of bad pixels > 1.

Image	AW	$AW+CW$	$AW+CW+U$
Adirondack	0.51	0.52	**0.48**
ArtL	1.28	1.31	**1.18**
Jadeplant	5.37	5.33	**4.05**
Motorcycle	**0.91**	0.97	1.02
MotorcycleE	**0.93**	0.96	1.01
Piano	1.45	1.47	**1.44**
PianoL	6.88	6.81	**6.29**
Pipes	2.12	1.84	**1.78**
Playroom	1.65	1.70	**1.45**
Playtable	0.91	0.96	**0.88**
PlaytableP	0.64	0.71	**0.59**
Recycle	0.95	0.94	**0.91**
Shelves	**3.74**	3.85	3.88
Teddy	**0.37**	0.39	0.38
Vintage	2.25	**2.01**	2.03
Average	2.00	1.98	**1.82**

Table 5.3: Comparative table: average disparity error.

Image	AW	$AW+CW$	$AW+CW+U$
Adirondack	3.40	3.53	**3.36**
ArtL	5.40	5.62	**5.02**
Jadeplant	24.72	25.11	**20.14**
Motorcycle	**5.48**	5.62	5.88
MotorcycleE	**5.32**	5.46	5.81
Piano	5.70	5.93	**5.66**
PianoL	19.25	18.80	**18.74**
Pipes	9.80	9.16	**8.67**
Playroom	6.70	6.86	**6.45**
Playtable	**3.97**	4.26	4.10
PlaytableP	3.13	3.38	**3.00**
Recycle	4.19	**4.17**	4.19
Shelves	**8.64**	8.99	9.21
Teddy	1.11	1.37	1.28
Vintage	**7.57**	6.85	7.75
Average	7.62	7.67	**7.28**

Table 5.4: Comparative table: rmse of estimated disparity.

To evaluate the effectiveness of r-sampling we replace it with the pixel shared hypotheses LSL from [88]. In tab.5.5 our sampling strategy ($ns0$) in general matches the performance of LSL, and even gives improved performance for foreground objects $D1-fg$, which is caused by the larger number of hypotheses 25 for r-sampling vs. 16 for LSL due to different hypothesis generation strategies. The performance of our approach in tab.5.5 shows that our approach $ns0$ is robust to several levels of uniform real valued noise added to the initial disparity map D, with $ns0$ no noise, and added disparity noise $[-1,1]$ ($ns1$, $D_p(p)+\Delta ns1$), $[-2,2]$ ($ns2$, $D_p(p)+\Delta ns2$), $[-3,3]$ ($ns3$, $D_p(p)+\Delta ns3$). The evaluation of r-sampling and proposed approach (all stages enabled) tolerance to noise in initial disparity map is done using every fifth image from KITTI 2015 training data.

Algorithm	%bad D1-bg	%bad D1-fg	%bad D1-all
$ns0$	2.96	**9.26**	3.86
LSL	**2.90**	9.55	3.86
$ns1$	2.94	9.49	3.89
$ns2$	2.96	9.39	3.88
$ns3$	2.94	9.85	3.94

Table 5.5: Evaluation of r-sampling and proposed approach (with all stages enabled), tolerance to noise in initial disparity map, and comparison to LSL.

5.12 Proposed approach vs. state of the art

We validate our approach using the three commonly used stereo matching data benchmarks: The new Middlebury (15 images) data set [81], KITTI 2015[66] and 2012[24] (up to 200 images). The main competitors to our algorithm (using the Middlebury and KITTI data sets) are $MCNCC$[97] and MDP[53], since they were evaluated using the same data sets (see tab.5.6 and tab.5.7). To compare to the state of the art, from all data sets competitors were selected by choosing the best performing convolutional neural network algorithms ([97, 28, 117]) and the best performing algorithms not using convolutional neural networks ([96, 66, 53]). For the KITTI 2015 and 2012 results, algorithms must appear in both data sets evaluation tables. The proposed algorithm is among top performers in the Middlebury (tab.5.6) and KITTI 2015/2012 (tab.5.7 and

tab.5.8). In tab.5.6 the disparity map is evaluated only in non-occluded areas, integer and sub-pixel scores are computed using an error pixel threshold of 2.0 (rank 9^{th} out of 44) and 0.5 (rank 8^{th} out of 44) respectively. Only non-anonymous entries compared as they have a publication and can be analysed.

Algorithm	$\%bad > 2.0$	$\%bad > 0.5$	avg. error	rms
Our[3]	10.50^{9th}	43.2^{8th}	3.17	15.6
PMSC	6.87^{1st}	39.1^{1st}	2.27	12.9
MeshE[4]	7.29^{2nd}	40.1^{2nd}	2.50	15.4
APAP	7.46^{3rd}	50.9^{12th}	3.89	21.1
MCNCC	8.29^{5th}	40.7^{4th}	3.82	21.3
MDP	12.6^{10th}	61.8^{27th}	5.28	23.1
Mesh	13.4^{11th}	51.2^{13th}	4.63	20.1

Table 5.6: Comparative table of results (on non-occluded pixels) on the new Middlebury data set. Only non-anonymous entries are used for comparison: PMSC[54], Mesh and MeshE[117], APAP[75], MCNCC[97], MDP[53].

Algorithm	%bad D1-bg	%bad D1-fg	%bad D1-all
Our^{8th}	3.55	12.30	5.01
Disp.v2^{1st}	3.00	5.56	3.43
MCNCC^{3rd}	2.89	8.88	3.89
PRSM^{5th}	3.02	10.52	4.27
MDP^{12th}	4.19	11.25	5.36
OSF^{15th}	4.54	12.03	5.79

Table 5.7: Comparative table of results (all pixels evaluated) on the KITTI 2015 data set (200 images). Only non-anonymous entries are shown: Disp.v2[28], PRSM[96], OSF[66].

In the KITTI benchmark, our algorithm ranks 8^{th} (out of 38), and 14^{th} (out of 83) for KITTI 2015 and 2012 respectively. The evaluation on KITTI 2012 proved

[3] Submitted to both KITTI and Middlebury benchmarks as LPU.

[4] Mesh[117] but using the cost from [97], but not published.

Algorithm	%bad noc	%bad occ	avg. noc	avg. occ
Our^{14th}	3.22	4.27	0.80	1.00
Disp. $v2^{2nd}$	2.37	3.09	0.70	0.80
$MCNCC^{4th}$	2.43	3.63	0.70	0.90
$PRSM^{8th}$	2.78	3.00	0.70	0.70
OSF^{15th}	3.28	4.07	0.80	0.90

Table 5.8: Comparative table of results (all pixels evaluated) on the KITTI 2012[24] data set (194 images). Only non-anonymous entries are shown.

more challenging (tab.5.8) mostly because of the presence of reflective regions, and colour image misalignment (the intensity images are coloured, but not properly aligned to the ground truth depth image). The top performing algorithms (for KITTI 2015 and 2012) currently achieve high performance by: exploiting scene specific content to solve ambiguities (e.g. cars in Disp.v2), training specifically for the data set (e.g. MCNCC), or using multiple image pairs to estimate disparity (e.g. PRSM, OSF). By contrast our algorithm achieves top performing results across multiple data sets by: using only one image pair, solving at different scales, not using scene specific features (e.g. cars), and generating a set of hypothesis from a single initial hypothesis.

5.13 Similarity cost evaluation and intermediate results

In sec.5.9 we described the algorithm steps used to compute the 3D disparity plane assignment using a pre-computed initialisation. This section shows the results of the following stages (All images shown are computed using fig.5.15):

1. Pre-compute disparity maps $Dleft_{pre}$ and $Dright_{pre}$ (e.g. using SGM, fig.5.16).
2. Compute updated solution (fig.5.18) at full-size by combining $Dleft_{pre}$ and $Dright_{pre}$ with current $Dleft$ and $Dright$ (fig.5.17). i.e compute $GHpre_p \cup GH_p$ for left and right, then do inference to update current solutions, and update $maxD$.
3. Compute updated solution (fig.5.19) at full-size by initialising with current $Dleft$ and $Dright$.

All disparity maps shown in this section are the raw result with NO post-processing.

Figure 5.15: Image number 126 from KITTI 2012 data set.

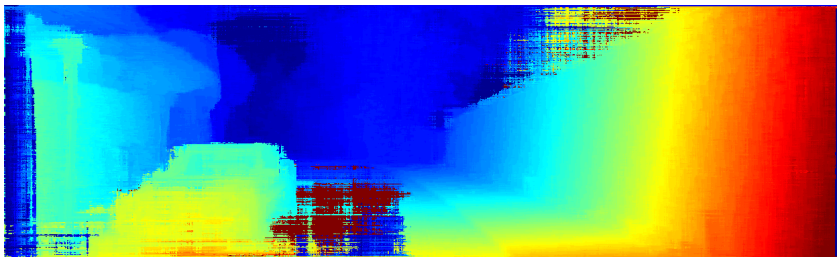


Figure 5.16: Disparity $Dleft_{pre}$ obtained using SGM see sec5.10.

In fig.5.16 it can be noticed the large error in estimation (large red patch in the middle), which is caused by the bright and almost untextured road. However, fig.5.17 shows that solving at a lower scale (and obtaining planes per pixel) has corrected the large error from initialisation.

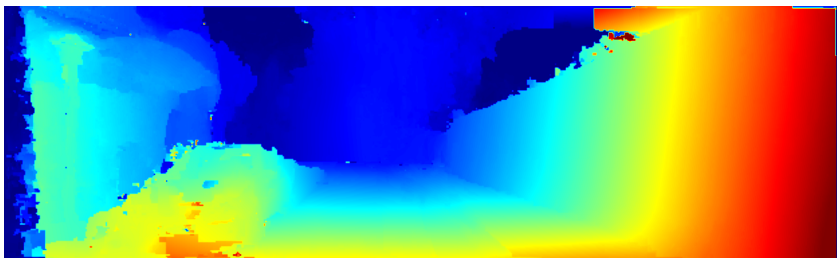


Figure 5.17: Computed disparity map $Dleft$ from lower scales.

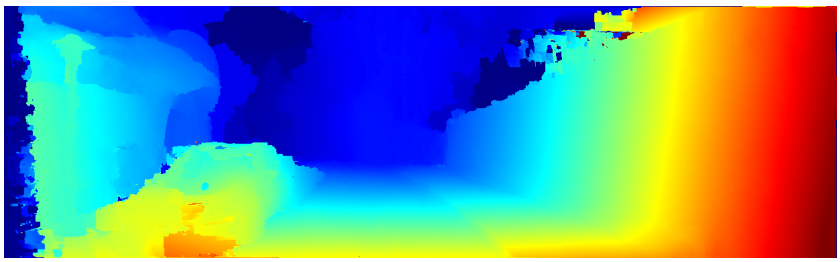


Figure 5.18: Combined $Dleft$ and $Dleft_{pre}$.

Fig.5.18 is the result of combining the initial disparity map (fig.5.16) with a solu-

tion from a lower scale (fig.5.17). The combination of initialisation and lower scales keeps the correct disparities values for the road from the lower scale. Finally, fig.5.19 shows the result of combining left and right disparity maps.

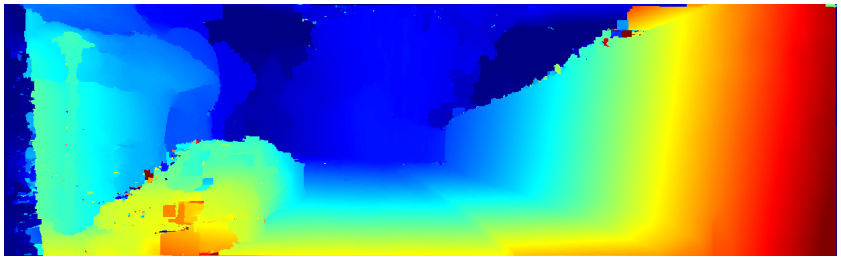
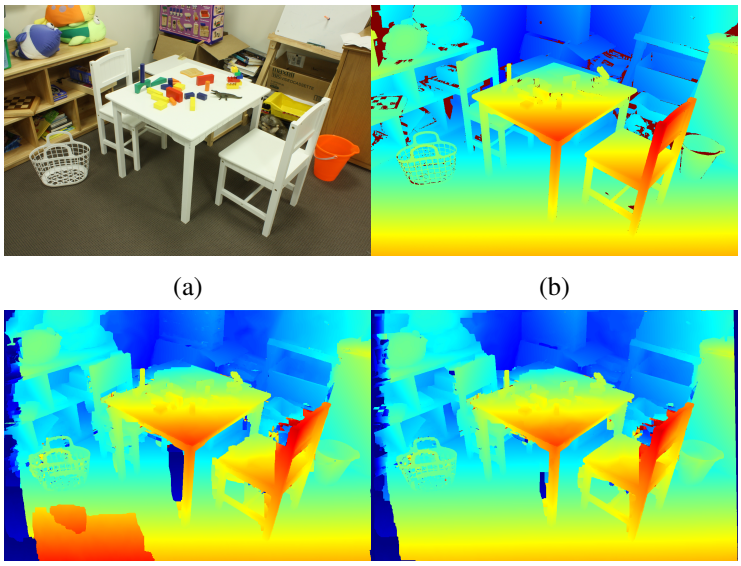


Figure 5.19: Final solution computed after pre-computed initialisation, multi-scale combination and refinements.

5.14 Handling rectification errors



(a) (b)

(c) (d)

Figure 5.20: Handling rectification errors: (a) Reference image “Playtable”, (b) Groundtruth, (c) Raw result without error correction, (d) Raw result with rectification error correction.

So far it is assumed that no rectification errors are present in the image. In order to handle image rectification errors, key points are computed per image using ASIFT[70], then the average vertical disparity is computed. Finally, if the average vertical disparity

is larger than τ_{vert} pixels a fundamental matrix is estimated, and images are rectified using Loop-Zhang's algorithm [57]. Remapping the estimated disparity to the original image is trivial once the rectifying homographies are known.

Fig.5.20 clearly shows that the disparity map without error correction has a larger error in the lower left corner (large bright patch), while the disparity map using error correction has more plausible disparity values in the same area. This procedure to reduce rectification errors is done on both Middlebury, KITTI 2015, and KITTI 2012 data sets, using $\tau_{vert} = 0.3$ pixels.

5.15 Critical analysis discussion

The results presented so far have shown that the proposed algorithm delivers some of the best performing results across different data sets. However, our approach still faces the following issues:

- **Fixed window sizes**: The proposed algorithm to adapt the window size according to the underlying 3D surface assumes that only two fixed window sizes are possible. However, if there are fine details (e.g. thin fences) that require a smaller window size, our approach is likely to be unable to recover such details.

- **Uniqueness term**: The proposed uniqueness term has the disadvantage of being local. This means that there is no hypothesis interaction when computing it, and therefore the uniqueness term is prone to make the energy function gets stuck in a local minimum. Another issue is that using a local term requires the algorithm to keep track of its current value, which is not necessarily correct.

- **Asymmetric cost function**: The proposed cost function is only computed from one image to the other and this results in a strong bias towards the reference image, e.g. the cost in the right image may be better, but since it is not included in the left image cost it may compute a bad matching cost. It would be desirable to use a function that reduces this problem.

- **Planar assumption**: The core idea of the proposed algorithm is to use 3D plane labelling. However, this may not necessarily be the case for round surfaces or very thin surfaces. It would be better to use the spherical conformal model, per pixel as it can model planes, spheres, lines, and points.

- **Large textureless areas**: The proposed algorithm gives reasonable results in textureless areas by using 1) the local uniqueness term, 2) multi-scale disparity estimation, 3) Adaptive function based on texture information. However, it would be better if our algorithm was able to explicitly label such areas, as this would result in less noisy disparity maps and possibly faster hypothesis propagation.

- **Intensive use of interpolation**: The proposed cost function is computed per pixel, and requires intensive use of interpolation making it slow. A new type of cost function is needed so that it can be computed over a large area, and avoids introducing either centre pixel or segment bias.

5.15.1 Detailed analysis of KITTI 2015 test image 0

Here we make a visual comparison of our results with the competitors of tab.5.7. Fig.5.21 shows the left reference image "0" with overlaid boxes to evaluate three types of regions 1) low contrast regions, 2) fine details, 3) edges. Our result is shown in fig.5.22 and is compared to Disp. v2 (fig.5.24), MCNCC (fig.5.26), PRSM (fig.5.28), and OSF (fig.5.30). The test images do not have groundtruth.

Figure 5.21: Kitti image "0" (left image).

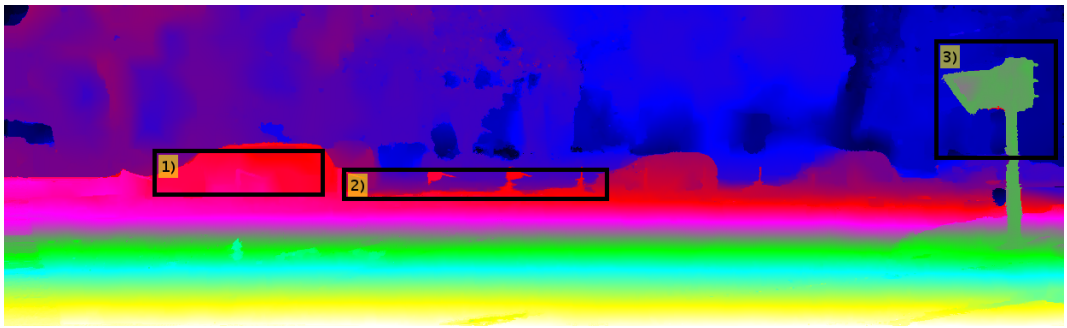


Figure 5.22: Kitti image disparity image "0" (our result).

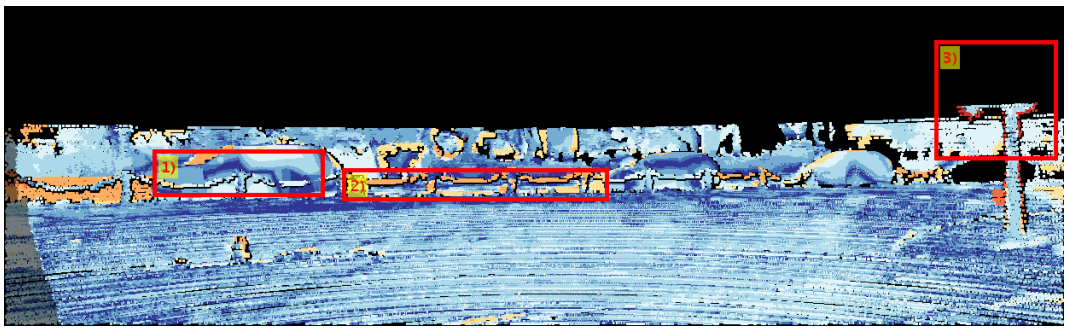


Figure 5.23: Kitti image error map “0” (our result).

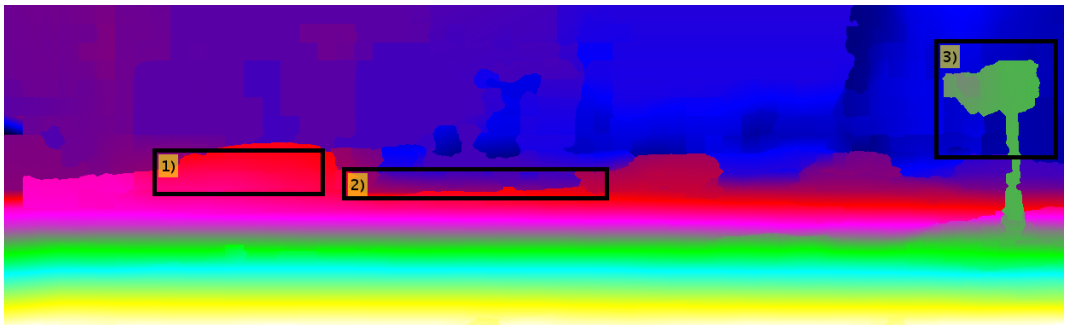


Figure 5.24: Kitti image disparity image “0” (Disp. v2 result).

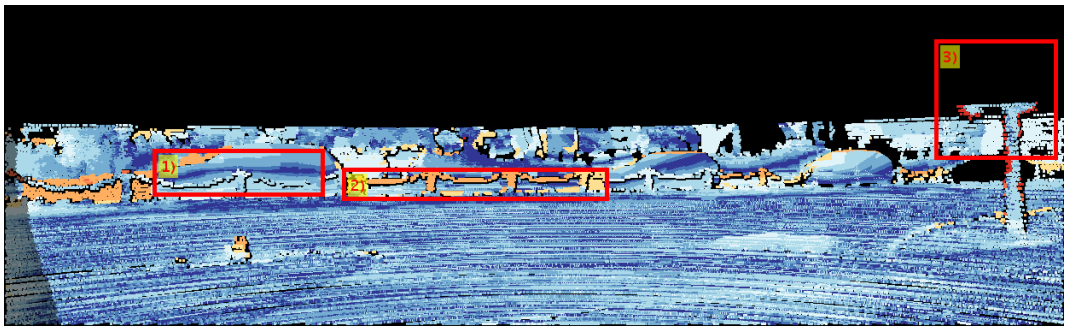


Figure 5.25: Kitti image error map “0” (Disp. v2 result).

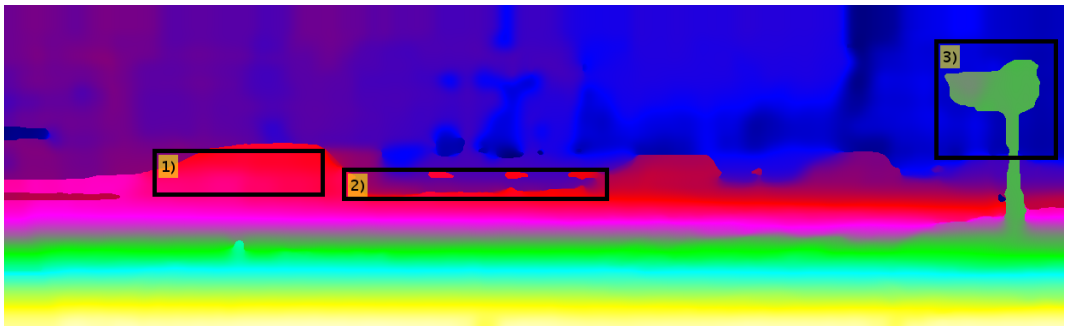


Figure 5.26: Kitti image disparity image “0” (MCNCC result).

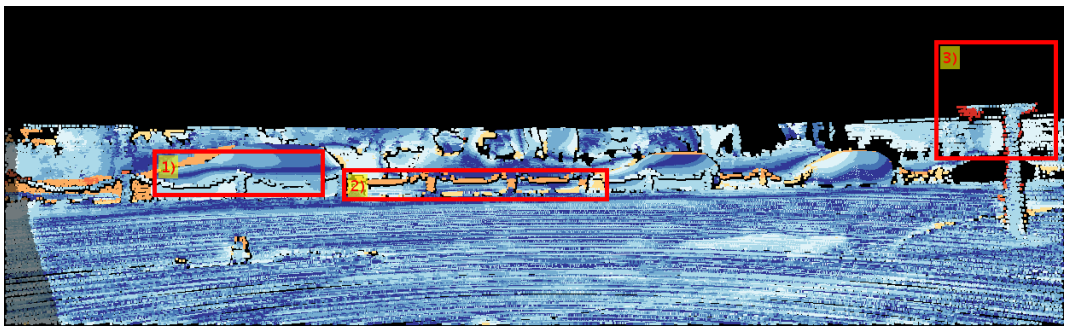


Figure 5.27: Kitti image error map “0” (MCNCC result).

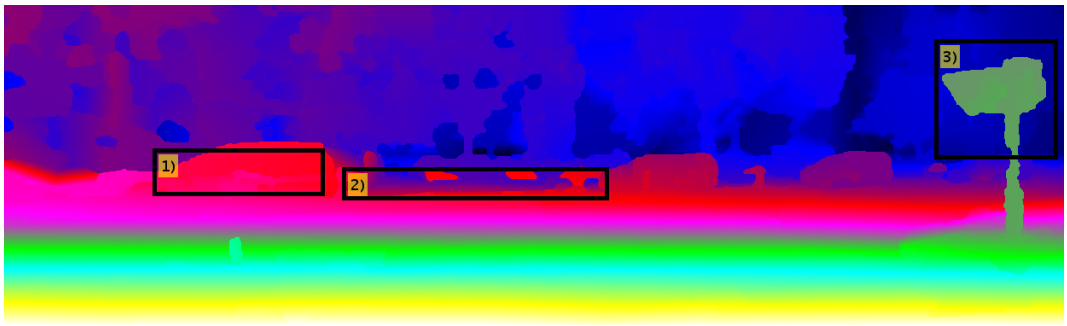


Figure 5.28: Kitti image disparity image “0” (PRSM result).

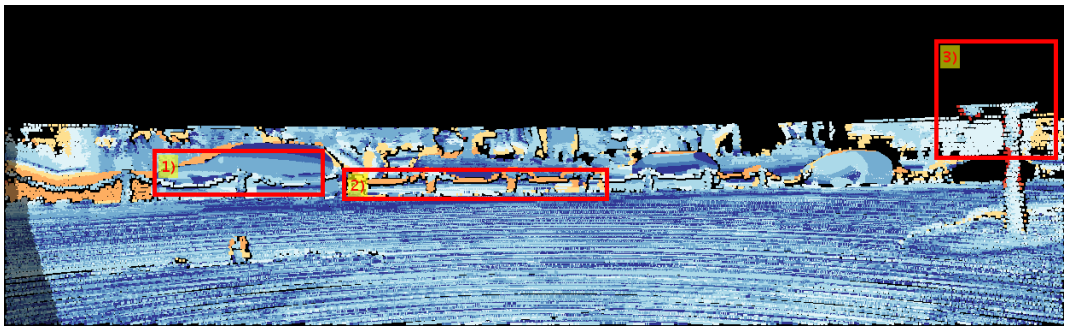


Figure 5.29: Kitti image error map “0” (PRSM result).

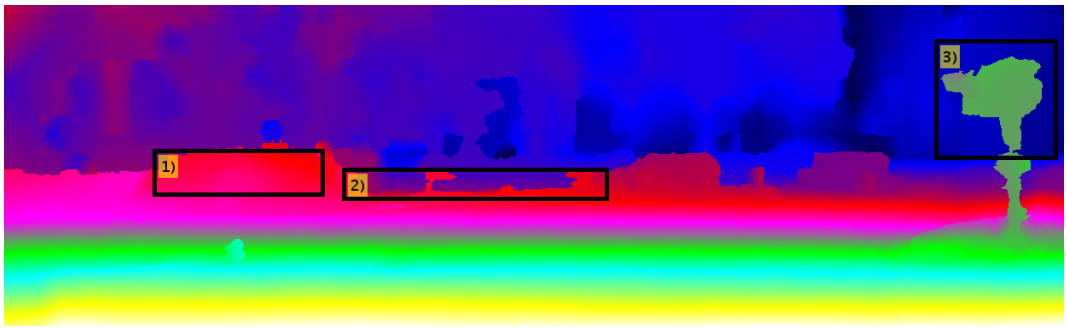


Figure 5.30: Kitti image disparity image “0” (OSF result).

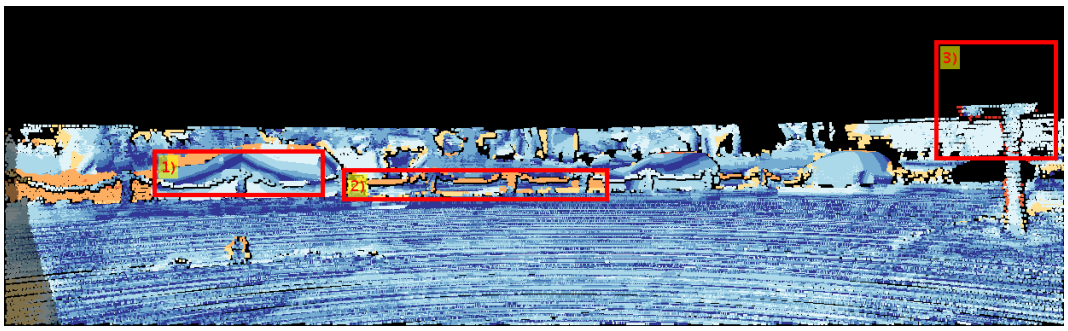


Figure 5.31: Kitti image error map “0” (OSF result).

1) In the low contrast region our algorithm shows higher error (fig.5.23), whereas Disp. v2 (fig.5.25), MCNCC (fig.5.27), PRSM (fig.5.29), and OSF (fig.5.31) have a lower error, which can be explained by the use of disparity plane segments, scene specific priors and use of convolutional neural networks.
2) In the fine details regions all algorithms our (fig.5.23), MCNCC (fig.5.27), PRSM (fig.5.29), and OSF (fig.5.31) were barely able to recover the thin fence (i.e. have large error), whereas Disp. v2 (fig.5.25) was unable to recover any detail in the same area. This case illustrates the problem of using fixed window sizes and the centre pixel bias.
3) In the edge evaluation region our algorithm (fig.5.22) recovers the triangle shaped traffic signal and the traffic light fine details, while the results of Disp. v2 (fig.5.24), MCNCC (fig.5.26), PRSM (fig.5.28), and OSF (fig.5.30) look more like a blob which is barely recognisable. This is explained by the inclusion of the pixel-wise cost, and adaptive windows that helps keep these fine details.

5.15.2 Detailed analysis of KITTI 2015 test image 2

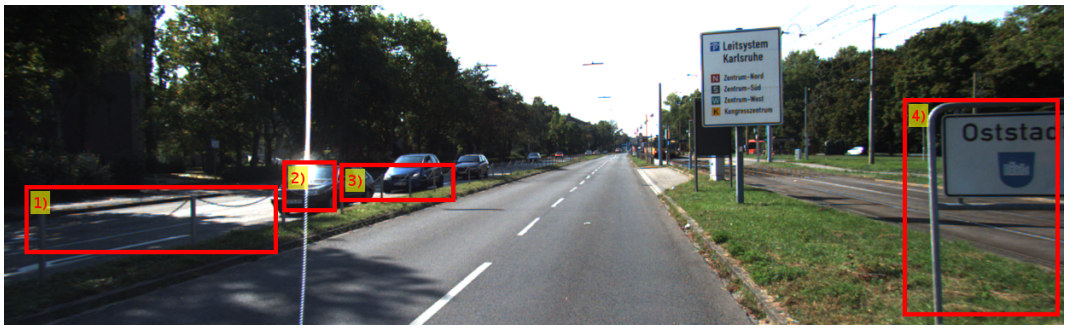


Figure 5.32: Kitti image “2” (left image).

The following images show a comparison of our results with the competitors of tab.5.7. Fig.5.32 shows the left reference image “2” with overlaid boxes to evaluate

four types of regions 1) fine details (partial success), 2) bad lighting condition (saturated pixel), 3) fine details (failure case), 4) disparity overgrow. Our result is shown in fig.5.33 and is compared to Disp. v2 (fig.5.35), MCNCC (fig.5.37), PRSM (fig.5.39), and OSF (fig.5.41).

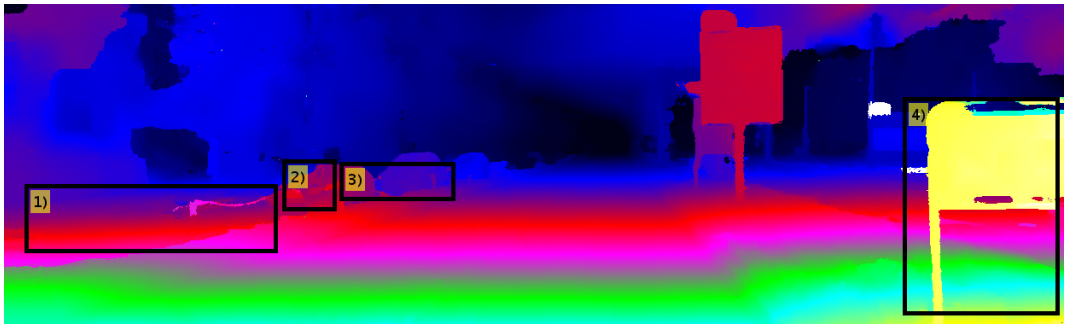


Figure 5.33: Kitti image disparity image “2” (our result).

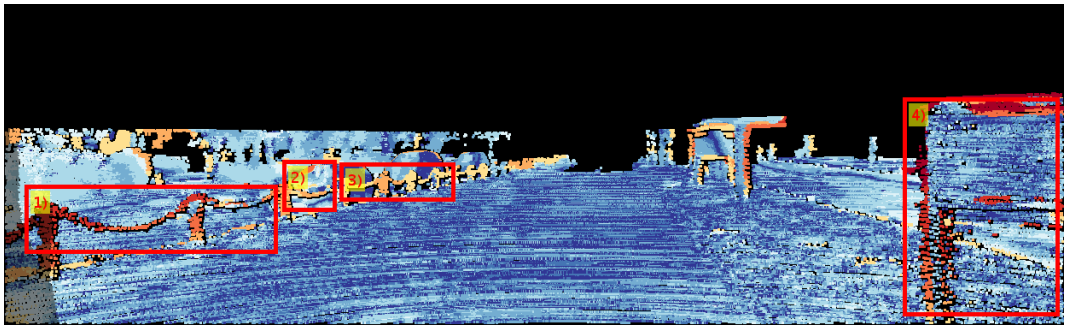


Figure 5.34: Kitti image error map “2” (our result).

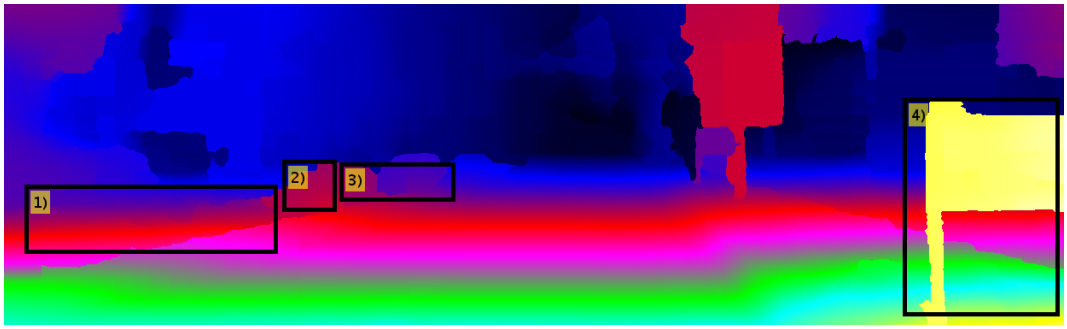


Figure 5.35: Kitti image disparity image “2” (Disp. v2 result).

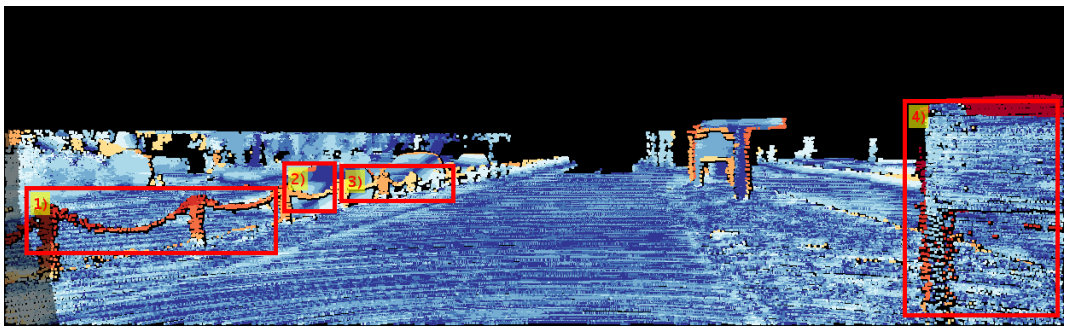


Figure 5.36: Kitti image error map “2” (Disp. v2 result).

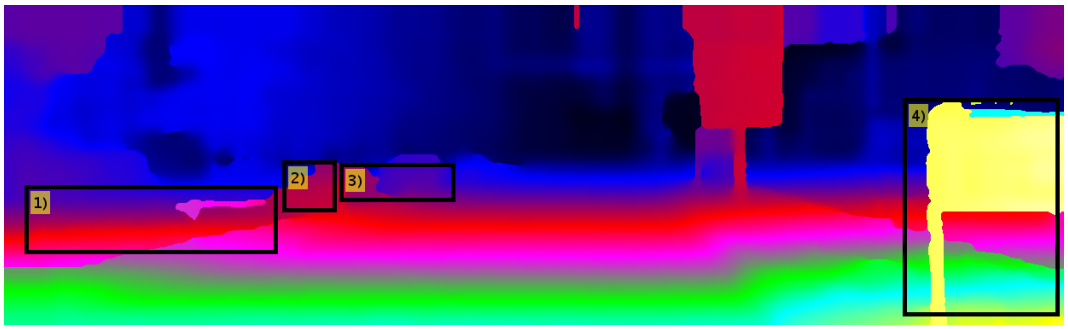


Figure 5.37: Kitti image disparity image “2” (MCNCC result).

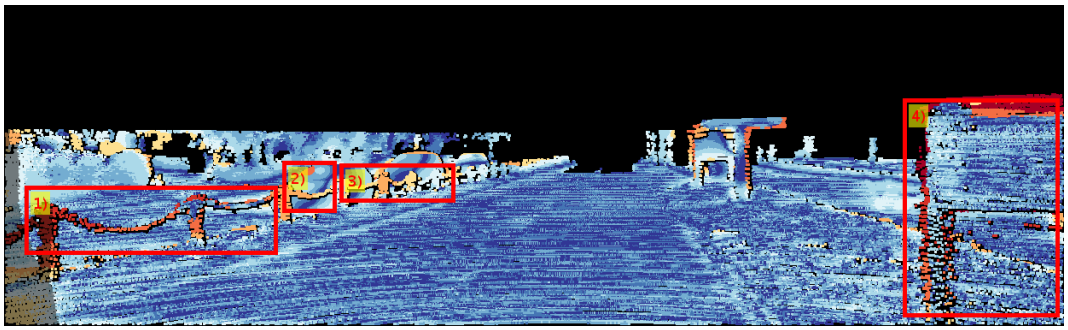


Figure 5.38: Kitti image error map “2” (MCNCC result).

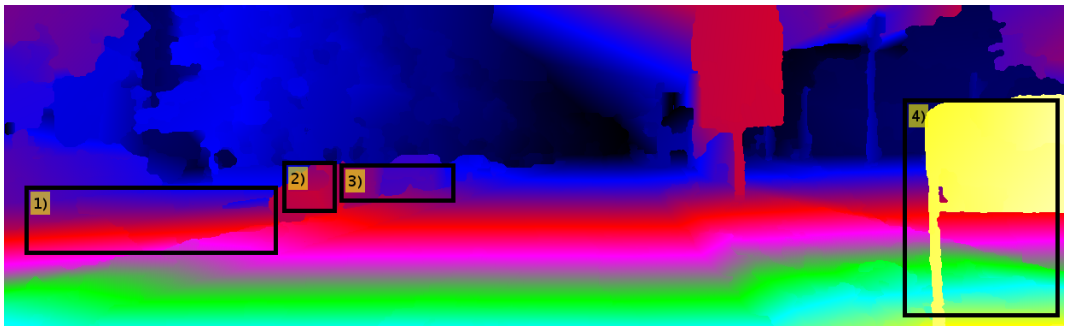


Figure 5.39: Kitti image disparity image “2” (PRSM result).

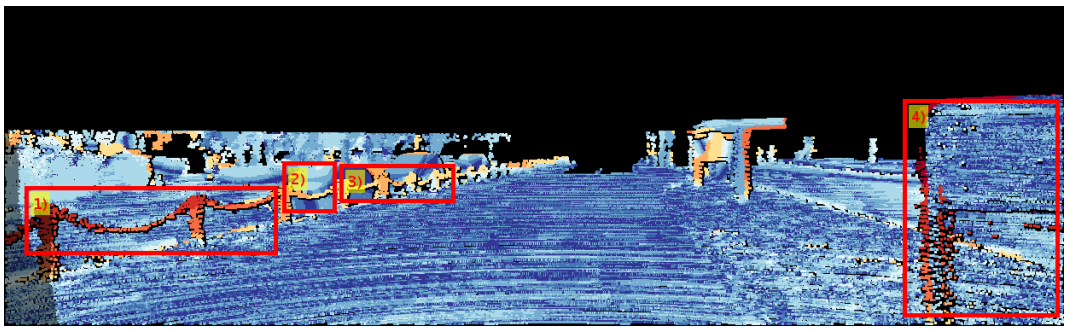


Figure 5.40: Kitti image error map “2” (PRSM result).

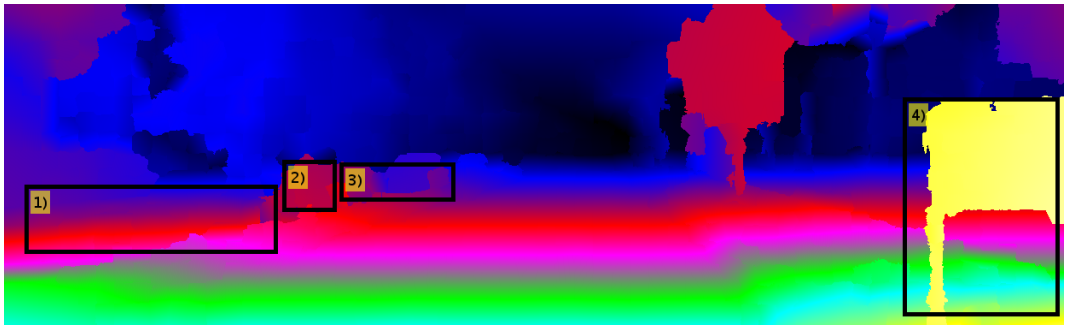


Figure 5.41: Kitti image disparity image “2” (OSF result).

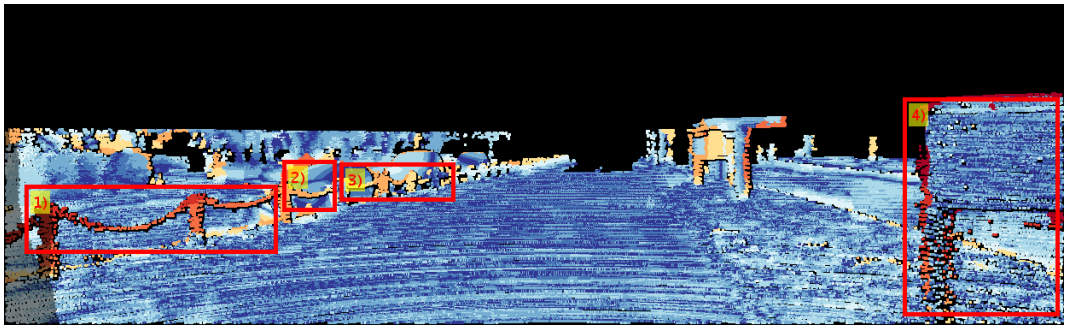


Figure 5.42: Kitti image error map “2” (OSF result).

1) In the fine details region (partial success) our algorithm (fig.5.33) and MCNCC (fig.5.37) were able to recover half of the fence, whereas all other algorithms Disp. v2 (fig.5.35), PRSM (fig.5.39), and OSF (fig.5.41) were unable to recover any detail in the same area. This case illustrates the advantage of using a pixel-wise cost with our adaptive windows.

2) In the bad lighting condition region all algorithms our (fig.5.33), Disp. v2 (fig.5.35), MCNCC (fig.5.37), PRSM (fig.5.39), and OSF (fig.5.41) partially succeeded in recovering the car despite sun flare.

3) In the fine details region all algorithms (fig.5.34), Disp. v2 (fig.5.36), MCNCC

(fig.5.38), PRSM (fig.5.40), and OSF (fig.5.42) failed to recover the thin fence. This case illustrates the problem of using fixed window sizes and the centre pixel bias.

4) In the disparity overgrowth region it can be seen that our algorithm (fig.5.33) and PRSM (fig.5.39) recover the traffic sign with better defined details, while the results of Disp. v2 (fig.5.35), MCNCC (fig.5.37), and OSF (fig.5.41) are overgrown. Additionally, the corresponding error images show that the ground truth has an error as the whole traffic sign has almost the same disparity, while the reference image fig.5.32 shows that there should be holes where the road is seen and therefore should have a different disparity.

5.15.3 Detailed analysis of KITTI 2015 test image 18

The following images show a comparison of our results with the competitors of tab.5.7. Fig.5.43 shows the left reference image "18" with overlaid boxes to evaluate five types of regions 1) almost textureless , 2) reflective, 3) fine details, 4) planar bias, 5) disparity overgrown and pixel centre bias. Our result is shown in fig.5.44 and is compared to Disp. v2 (fig.5.46), MCNCC (fig.5.48), PRSM (fig.5.50), and OSF (fig.5.52).

Figure 5.43: Kitti image "18" (left image).

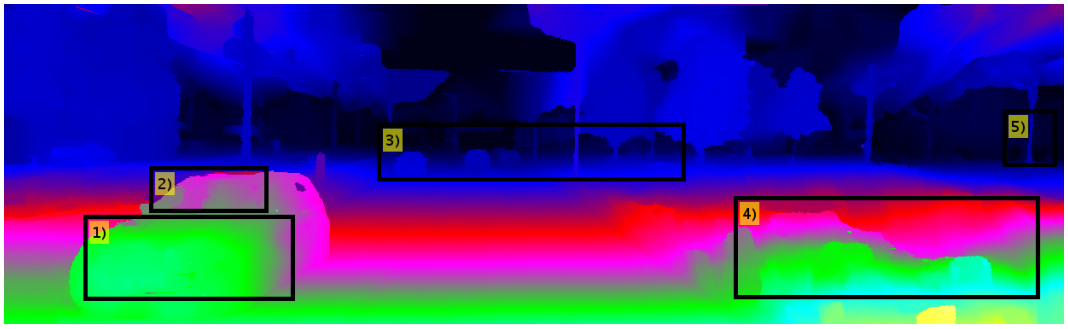


Figure 5.44: Kitti image disparity image "18" (our result).

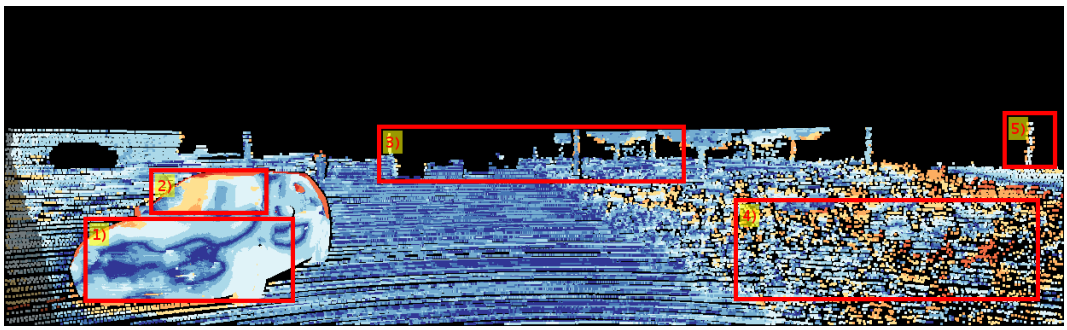


Figure 5.45: Kitti image error map “18” (our result).

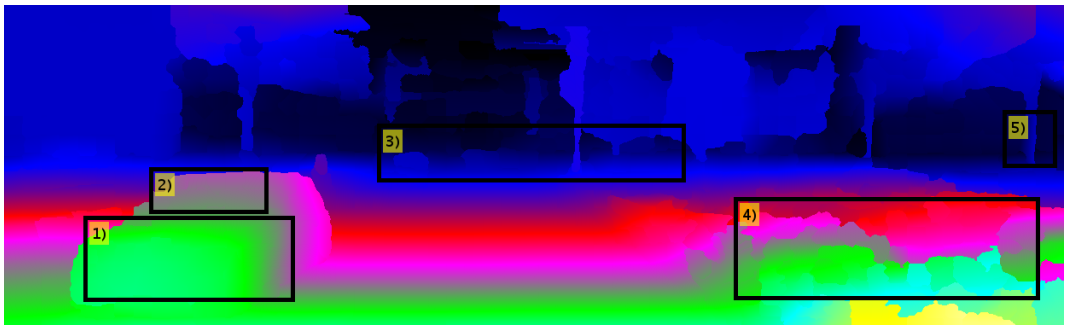


Figure 5.46: Kitti image disparity image “18” (Disp. v2 result).

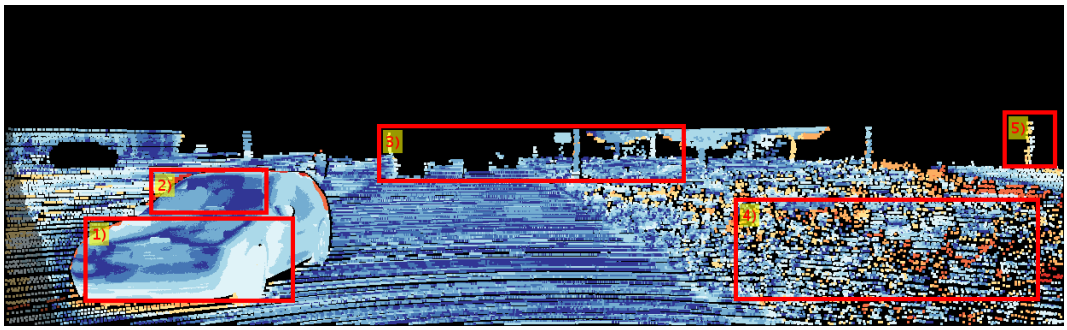


Figure 5.47: Kitti image error map “18” (Disp. v2 result).

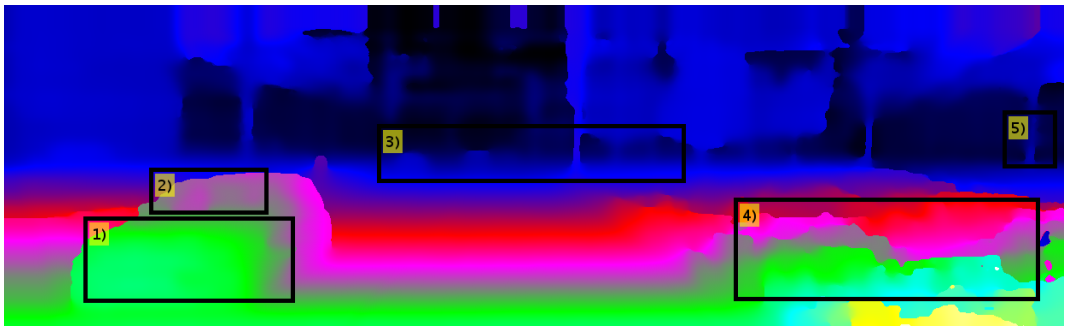


Figure 5.48: Kitti image disparity image “18” (MCNCC result).

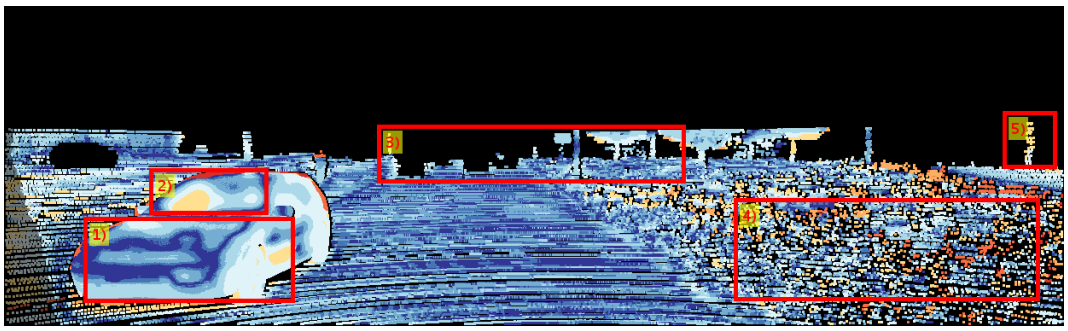


Figure 5.49: Kitti image error map “18” (MCNCC result).

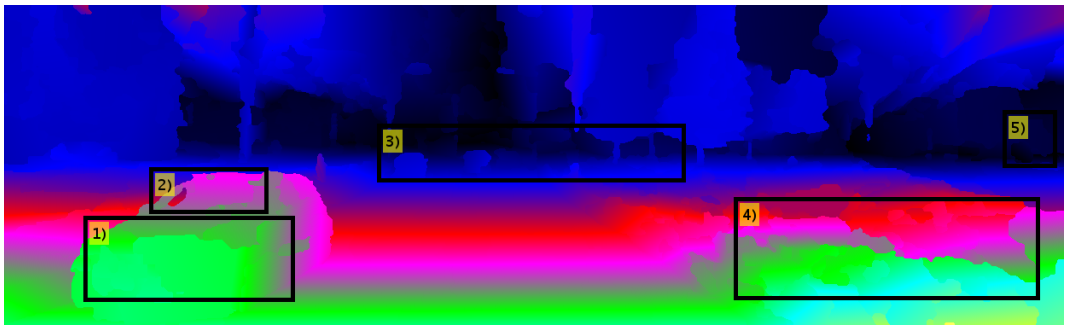


Figure 5.50: Kitti image disparity image “18” (PRSM result).

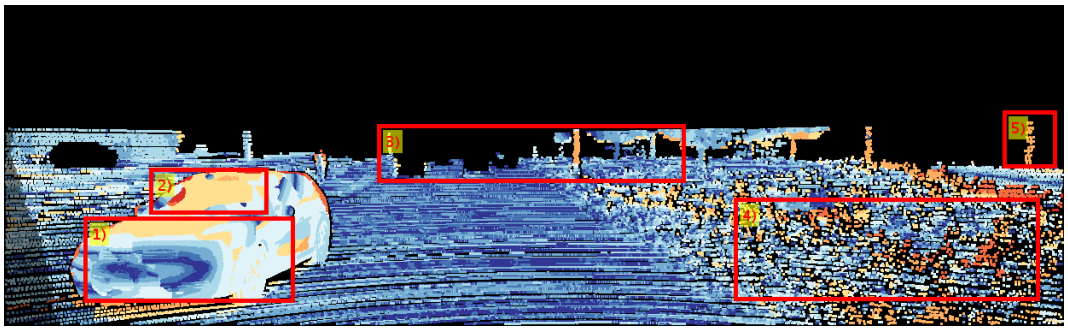


Figure 5.51: Kitti image error map “18” (PRSM result).

Figure 5.52: Kitti image disparity image “18” (OSF result).

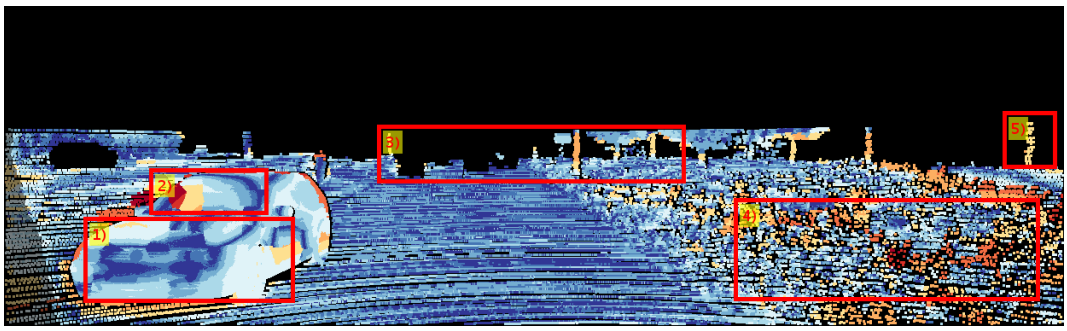


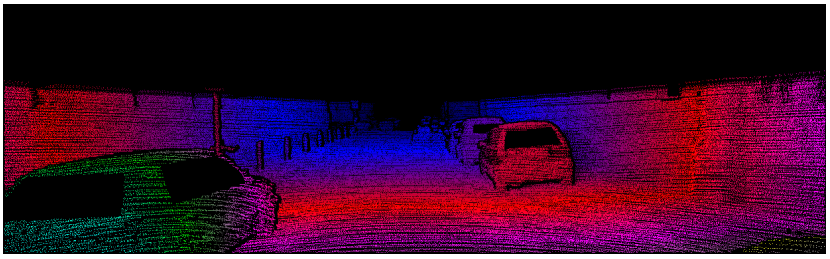
Figure 5.53: Kitti image error map “18” (OSF result).

1) In the almost textureless region (partial success) our algorithm (fig.5.44), Disp. v2 (fig.5.46), MCNCC (fig.5.48), and OSF (fig.5.52) are able to recover the car, whereas PRSM (fig.5.50) has problems handling the textureless area. For our algorithm the performance is explained by the use of planes and the uniqueness term.
2) In the reflective region our algorithm (fig.5.44), MCNCC (fig.5.48), PRSM (fig.5.50), and OSF (fig.5.52) partially succeeded in recovering the car windscreen, whereas Disp. v2 (fig.5.46) estimates the correct values due to the use of car model priors.
3) In the fine details regions only our algorithm (fig.5.45) recovers all trees and cars present, while Disp. v2 (fig.5.47), MCNCC (fig.5.49), PRSM (fig.5.51) achieve partial success as trees are almost lost and cars are barely recognisable. OSF (fig.5.53) failed to recover the trees and cars. This is explained by the inclusion of the pixel-wise cost, and adaptive windows that helps keep these fine details.
4) The bushes in fig.5.43 are replaced by large planes, which results in large error for our algorithm (fig.5.45), Disp. v2 (fig.5.47), MCNCC (fig.5.49), PRSM (fig.5.51), and OSF (fig.5.53).
5) The disparity overgrown and pixel centre bias region shows that our algorithm (fig.5.44) correctly estimates the disparity of the pole without significant overgrowth. Disp. v2 (fig.5.46), MCNCC (fig.5.48) achieve partial success, but the area is either overgrown or blurred. PRSM (fig.5.50), and OSF (fig.5.52) fail to recover the tree due to their strong planar segment bias.

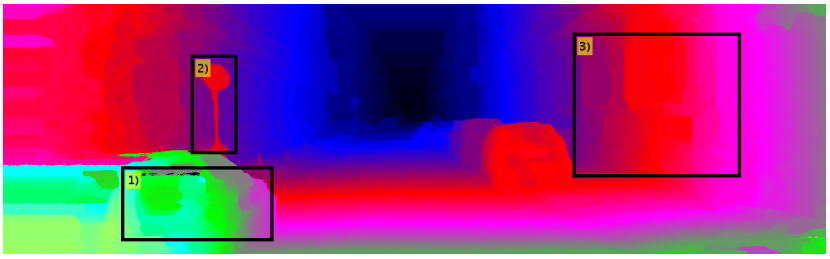
5.15.4 Detailed analysis of kitti 2012 image 15 *LCU* vs. *LPU*

Here we make a visual comparison of our results for the *LCU* algorithm from chapter 4, and the our new algorithm *LPU*. The are three comparison regions 1) reflective, 2) fine details, and 3) saturated, see fig.5.54.

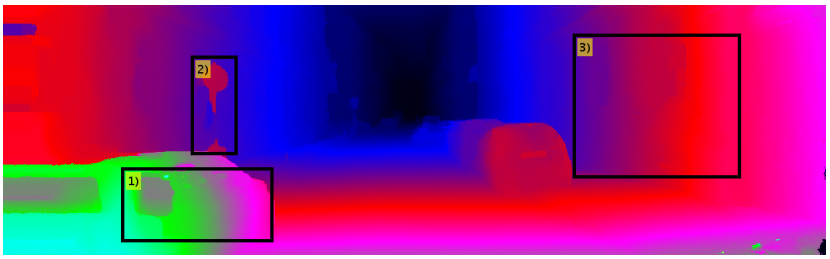
(a) KITTI image 15



(b) Groundtruth



(c) LCU



(d) LPU

Figure 5.54: LCU vs. LPU for KITTI 2012 training image 15.

1) In the reflective region region the new *LPU* algorithm performs better than *LCU* due to the use of planes and the uniqueness term, while the pixel-wise used in *LCU* becomes ambiguous in this region.

2) In the fine details regions the *LCU* algorithm correctly recovered the pole of the traffic sign, while our new algorithm *LPU* was unable to recover the thin pole. This case illustrates the problem of using fixed window sizes and the centre pixel bias. By contrast *LCU* does not suffer from this issue due to the use of a pixel-wise cost function

with no aggregation.
3) In saturated and textureless region our algorithm *LPU* gives better results than *LCU* due to the use of the uniqueness term, and the use of slanted planes. The *LCU* algorithm has a strong fronto-parallel bias in this region.

5.15.5 Detailed analysis of performance using groundtruth initialisation

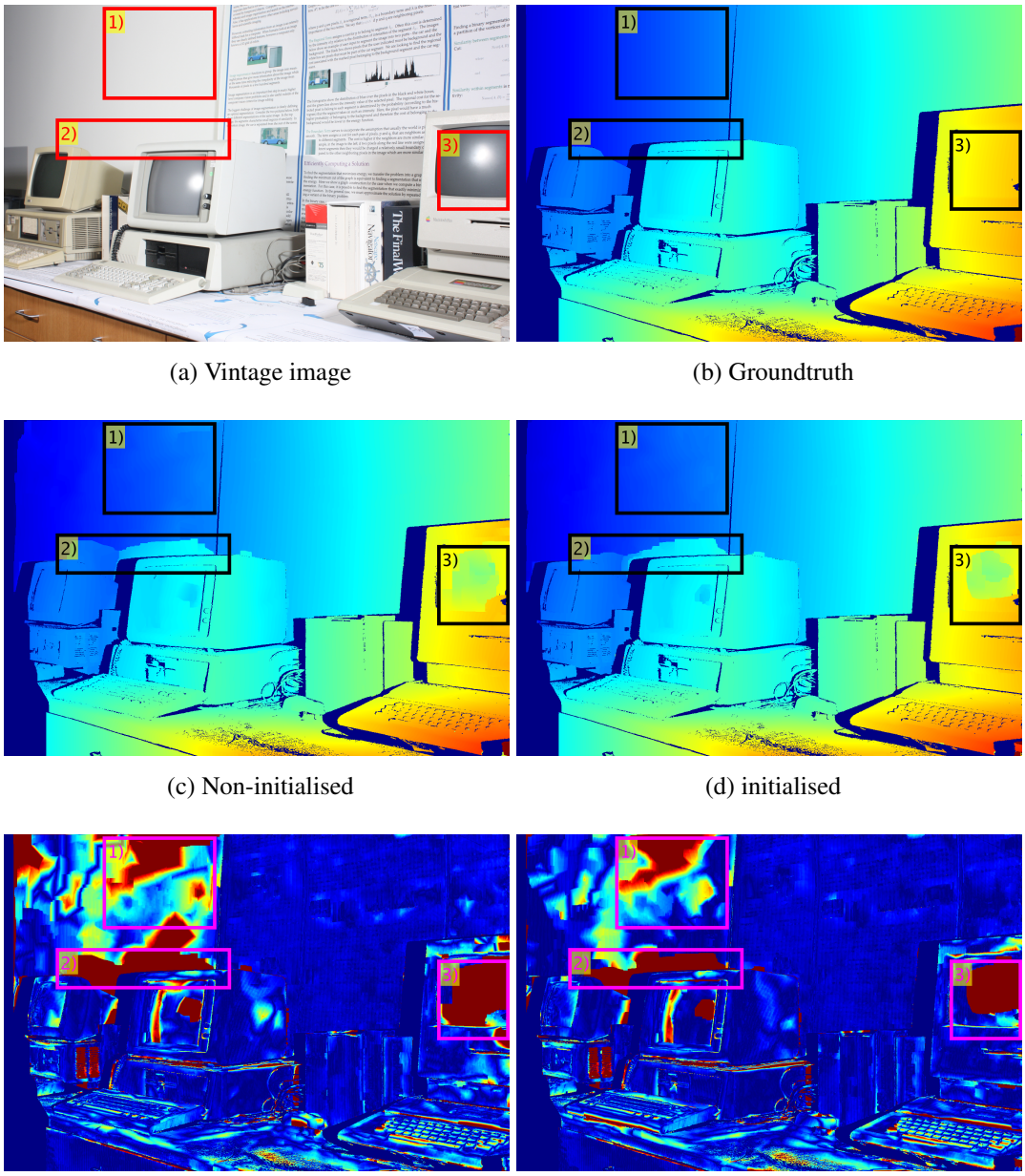
As described previously our algorithm can use either a precomputed disparity map or random initialisation. Taking advantage of this property we have initialised our algorithm using groundtruth disparity maps, and evaluated what are the effects of this on the algorithm performance.

Function	%bad noc	avg. error	rms
GTI	10.57	1.46	6.60
LPU	12.95	1.82	7.28

Table 5.9: Comparative table proposed algorithm initialised with a precomputed disparity map (*LPU*) vs. groundtruth (*GTI*).

Tab.5.9 shows that using the groundtruth (*GTI*) as initialisation results in less error for all metrics in non-occluded areas without post-processing. However, it would be expected that the error should be even lower (i.e. zero). This discrepancy can be explained by several factors 1) similarity cost noise and 2) smoothness model. In other words our model gives good results, but it is only an approximation (e.g. assumed connectivity of surfaces, perfect pixel similarity, and planar surfaces) of the process that produced the groundtruth.

Fig.5.55 shows the effect of initialisation on three different regions 1) slanted textureless, 2) depth discontinuity close to textureless surface, and 3) reflective surface. Fig.5.55a is the left reference image, fig.5.55b is the groundtruth disparity map (without occluded areas), fig.5.55c is the resulting disparity map using our algorithm (without occluded areas), and fig.5.55d is the result using the groundtruth disparity map as initialisation (without occluded areas). Fig.5.55e and fig.5.55f show the error maps (from 0 to 3 pixel of error) for non-initialised and initialised results respectively.
1) In the slanted textureless region the groundtruth initialised result (fig.5.55f) has



(a) Vintage image (b) Groundtruth

(c) Non-initialised (d) initialised

(e) Non-initialised error (f) initialised error

Figure 5.55: Effect of using the groundtruth disparity map as initialisation.

a lower error, but not constant as it would be expected. This is caused due to the ambiguity of the similarity cost in the textureless area. By contrast the non-initialised (fig.5.55e) result has a higher error as the precomputed disparity map is already noisy.

2) In the slanted textureless region the groundtruth initialised result (fig.5.55f) has a lower error, but disparity overgrowth is noticeable in fig.5.55d. This is caused by the ambiguity of the similarity cost in the textureless area, and issues detecting the possible depth discontinuity used in the smoothness term. By contrast the non-initialised

(fig.5.55e) result has a higher error due cost ambiguities in the same region, which favours disparity overgrowth in fig.5.55c.
3) In the reflective surface region the groundtruth both initialised (fig.5.55f) and non-initialised (fig.5.55e) results have have a large error. This explained due to the noisy and ambiguous cost that are characteristic of reflective regions.

5.16 Summary

The proposed approach delivers some of the best performing results (on KITTI and Middlebury leader boards) even though only one global hypothesis is used (eliminating the need to update multiple hypotheses), and no convolutional neural network (e.g. [117, 97, 28]) or prior 3D models (e.g. cars) are used. The use of r-sampling and P-sampling are novel, simple and effective ways of simulating a propagation scheduler, which can be executed in parallel, unlike those from [9, 6, 117]. The initial parameters used were selected based on the visual quality of the depth discontinuities of the estimated disparity maps.

The presented adaptive window aggregation algorithm is capable of reducing the centre pixel bias and fixed window size issues commonly found when using the traditional adaptive windows in DPI algorithms. This was achieved by using the underlying 3D scene and image entropy as additional local cues to improve the stereo matching function. Finally, our algorithm works as a combination of both DPI and FPI algorithms when a precomputed disparity map exists, and when no disparity map is available it behaves just like any other DPI algorithm.

Chapter 6

Plane labelling trinocular stereo matching with baseline recovery

In this chapter we present an algorithm which recovers the rigid transformation that describes the displacement of a binocular stereo rig in a scene, and use this to include a third image to perform dense trinocular stereo matching and reduce some of the ambiguities inherent in binocular stereo. The core idea of the proposed algorithm is the assumption that the binocular baseline is projected to the third view, and thus can be used to constrain the estimate of the stereo rig transformation. Our approach shows improved performance over the binocular stereo algorithm presented in chapter 5, and the accuracy of the recovered motion allows us to compute optical flow from a single disparity map. These claims are validated with the KITTI 2012 data set. The algorithm presented in this chapter is referred to as trinocular baseline recovery (TBR).

6.1 Proposed trinocular algorithm

The problem of 3D plane labelling stereo matching using three images, two binocular and a third with some known displacement, can be described as finding the correspondences for each pixel from image I_l to I_r and I_u by assigning the 3D disparity plane that encodes the position of the corresponding 3D point X. Using a 1D disparity implies that (I_l, I_r) are rectified, and a known projective transformation P (camera) maps X_i to pixel position x_i^u in I_u. Finding the optimal 3D disparity plane labelling D is modelled as an optimisation problem where the objective is to minimise eq.6.1.

$$E(D) = \arg\min_{D} \sum_{p}^{NumP} \{C_p(D_p) + \sum_{q \in N(p)} V_{pq}(D_p, D_q)\} \tag{6.1}$$

This equation is the same model as eq.5.1 but will be enhanced to take into account a third image. $E(D)$ is the cost of the disparity assignment (energy), D is a set of planes and D_p encodes the plane that gives the disparity of the pixel at p with respect to another rectified image. $D_p(q)$ is the disparity estimated using plane D_p evaluated at pixel q. $NumP$ is the number of pixels in the image. $N(p)$ is a neighbourhood around p, and q is a neighbour of p. V_{pq} (*smoothness* term) is a function that evaluates how well the disparity at position p fits its neighbours (eq.5.19). The plane D_p has two parameters: a 3D unit normal vector $\hat{n}_p = (\hat{n}_p^x, \hat{n}_p^y, \hat{n}_p^z)$ and disparity d_p with $p = (x_p, y_p)$. The disparity of pixel $q = (x_q, y_q)$ using D_p is given by:

$$D_p(q) = a * x_q + b * y_q + c \tag{6.2}$$

where $a = -\hat{n}_p^x / \hat{n}_p^z$, $b = -\hat{n}_p^y / \hat{n}_p^z$ and $c = (\hat{n}_p^x * x_q + \hat{n}_p^y * y_q + \hat{n}_p^z * d_p) / \hat{n}_p^z$ just as previously described in chapter 5. The main difference when compared to the *LPU* algorithm (eq.5.9) is that C_p is now a function that measures the similarity/dissimilarity of three pixels, e.g. $I_l(p)$ is compared to $I_r(p + D_p(p)))$ and $I_u^l(\phi(P \cdot X))$ with $\phi(x) = (x_1/x_3, x_2/x_3)$. By adding the third matching pixel we increase the likelihood that pixels are correctly matched and thus disparities are correctly estimated.

The trinocular algorithm developed in this chapter works under several assumptions. 1) There is no prior extrinsic calibration that describes the stereo rig motion, as it can move freely. 2) The binocular baseline T_r (fig.6.1) is projected to I_u^l (after the stereo rig has been displaced), and thus can be used to constrain the recovery of the transformation P from the stereo rig, i.e. the third image is used to recover the baseline. 3) The proposed approach uses rectified binocular stereo pairs (i.e with fronto parallel cameras) with a known intrinsic matrix K and baseline known T_r. 4) The coordinate system origin is C_l, and C_{lu} is the new coordinate origin after the stereo rig has displaced. From this we will be able to estimate the transformation parameters R, T_{lu}, and T_{ru}, which then enables the use of eq.6.1. Fig.6.1 shows how the baseline endpoints (C_l, C_r) are projected to (e_{lu}, e_{ru}) by $[R|T_{lu}]$ and $[R|T_{ru}]$. The dotted green line connecting (e_{lu}, e_{ru}) shows how the points along T_r are projected by $P = K[R|T_{lu}]$ into I_u^l.

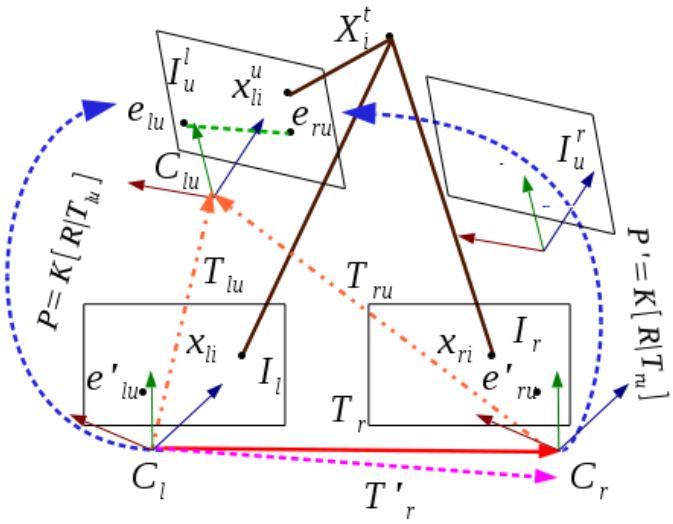


Figure 6.1: baseline T_r vs. recovered baseline T_r': Under perfect conditions the estimated epipoles e'_{lu} and e'_{ru} are the projections of the camera centre C_{lu}. These epipoles can be used to recover T_r exactly. However, in the presence of noise (realistic case when estimating the epipoles) T_r' is recovered. Knowing that T_r exists allows one to constrain an optimization problem to compute transformations that minimise the discrepancy of T_r' and T_r.

6.2 Related work

The idea of using one or more images has been previously explored to compute joint optical flow and disparity in [66, 96, 106], where the reference image (e.g. left image) is segmented and each segment is assumed to be a moving plane. All these algorithms work using four similar steps: 1) Compute an initial estimate of disparity and optical flow, 2) do plane fitting to generate plane hypotheses per image segment, 3) estimate transformation per segment, and 4) do per segment plane inference. This type of approach is known as scene flow estimation. Using disparity planes to estimate sub-pixel stereo disparity has been previously used in [43, 104, 9, 6, 73, 32, 88, 85, 106]. These algorithms can be classified in two categories: fixed plane inference (FPI) and dynamic plane inference (DPI). FPI algorithms usually work by making an initial disparity estimate and then extracting a set of plane hypotheses, which are then used to compute the 3D plane labelling. DPI algorithms use one or more plane hypotheses per pixel, and then propagate the planes with the "best" scores (depending on the cost function) to neighbouring pixels/regions assuming that neighbours/regions may have the same plane. The initial plane labelling is refined in a separate stage, i.e. planes are dynami-

cally updated. DPI algorithms have become the state of the art (e.g. [9, 6, 32, 88]). We use the *LPU* algorithm from chapter 5 to generate hypotheses and plane labelling.

In order to estimate the transformation from I_l/I_r to I_u the most common approach is to compute keypoints and recover the camera position as in [30], and then do bundle adjustment to refine the obtained solution (e.g. [2, 92]). These algorithms are designed to work with multiview uncalibrated stereo, and the bundle adjustment process estimates both optimised camera positions and 3D points. Another option is to compute the trifocal tensor using either matching points [30] or lines [120, 91] to recover the missing camera position like in the configuration described in fig.6.1, or use the trifocal tensor to do point/line transfer, which has the inconvenience of being unreliable at points that are close to the epipolar plane. Using either the multi-view approach or trifocal tensor is excessively complex especially when there are two calibrated cameras and only one extra camera's position needs to be computed.

6.2.1 Contributions

As noted previous approaches to recover the transformation P either rely on optical flow and disparity estimates, or using camera estimation algorithms that do not take into account the particular case of a calibrated stereo rig displacing in space. Our type of scenario is commonly found in vehicles moving either forward or backwards (e.g. [66]). We develop a camera recovery algorithm that exploits existing calibration to constrain and estimate a transformation $P = K[R|T_{lu}]$ that maps a 3D point X_i recovered from images I_l/I_r to a point X_i' consistent with the projected point x_i^u in I_u, and in doing so we also develop a pixel cost similarity that seamlessly uses a third image to reduce some of the pixel matching ambiguities inherent to the standard binocular stereo matching pixel cost. Our contributions can be summarised as follows:

- Algorithm to recover a rigid transformation $[R|T_{lu}]$ constrained by the baseline T_r of the calibrated stereo rig.
- Pixel similarity function that integrates three views in a DPI algorithm (we use [37]).

6.3 Baseline recovery

The case of a calibrated stereo rig moving as in fig.6.1 has the characteristic that the camera centre C_{lu} is projected as the epipoles e'_{lu}/e'_{ru} (see fig.6.1) in I_l/I_r, and the

distance between e'_{lu} and e'_{ru} is related to the baseline size. Furthermore, if two fundamental matrices F_{lu} and F_{ru} are available then $[R|T_{lu}]$ and $[R|T_{ru}]$ are extracted, and it is trivial to compute the baseline $T'_r = T_{lu} - T_{ru}$, but most importantly it is possible to measure the error of the recovered baseline. To illustrate this situation consider the following case of fronto-parallel cameras:

$$K \cdot [I|C_l] \cdot C_{lu} - K \cdot [I|C_r] \cdot C_{lu} = e'_{lu} - e'_{ru} \tag{6.3}$$

Eq.6.3 assumes that both cameras in the stereo rig are are fronto parallel with the same intrinsic parameters, and to further simplify the situation let the camera C_l be at the origin in world coordinates and thus $C_r = T_r$, $C_{lu} = T_{lu}$. Eq.6.3 then simplifies to:

$$\begin{aligned} K(\cancel{T_{lu}} - \cancel{T_{lu}}^{\mathbf{0}} - T_r) &= e'_{lu} - e'_{ru} \\ -K \cdot T_r &= e'_{lu} - e'_{ru} \end{aligned} \tag{6.4}$$

All 3D points projected in the image using the intrinsic matrix K are equal up to a scale factor [30] and thus from eq.6.4 the following relation is derived:

$$\|T_r\| = S\|K^{-1}(e'_{lu} - e'_{ru})\| \tag{6.5}$$

When calibration is available and T_r is known there are only three unknowns: S, e'_{lu} and e'_{ru}. The epipoles e'_{lu} and e'_{ru} are extracted from fundamental matrices F_{lu} and F_{ru}, which can be computed from keypoints that are consistent across three views (I_l,I_r and I^l_u). Therefore it is trivial to compute the scale factor S and estimate a baseline T'_r (fig.6.1) using eq.6.4, and R can be extracted from F_{lu}. Note T'_r and T_r should be same, however due to noise in the points used to estimate F_{lu}, F_{ru}, T'_r is an approximation of T_r. Since the stereo rig is calibrated, we know T_r and thus if there is a discrepancy with the recovered baseline T'_r, therefore refining the initial estimates R, T_{lu}, and T_{ru} should help to make T'_r closer to the known T_r.

6.3.1 Finding consistent transformations

The process of estimating the baseline T'_r (see fig.6.2) can be stated as finding an updated $[\hat{R}|\hat{T}_{lu}]$ and $[\hat{R}|\hat{T}_{ru}]$ such that they can be used to approximate the baseline T_r. This allows us to obtain fundamental matrices $\hat{F}_{lu}$ and $\hat{F}_{ru}$ with minimal Sampson error when evaluated, such that T_{lu}, T_{ru} translate points to the same depth (because we have

fronto-parallel cameras) and give minimum reprojection error of a point X_i to the third view. This is expressed as the following optimisation problem:

$$\arg\min_{\hat{R},\hat{T}_{lu},\hat{T}_{ru}} \sum_{i=0}^{n} [ds(x_{li},x^u_{li},F_{lu}) + ds(x_{ri},x^u_{li},F_{ru}) + \|x^u_{li} - \phi(PX_i))\|] + \|T_r - T'_r\| \quad (6.6)$$

where X^t_i is a 3D point at time t, x_{li},x_{ri},x^u_{li} are the projections of X_i in I_l,I_r,I^l_u with $P = K[R|T_{lu}]$. $\hat{F}_{lu} = K^{-1'}[\hat{T}_{lu}]_\times \hat{R}K^{-1}$ and $\hat{F}_{ru} = K^{-1'}[\hat{T}_{ru}]_\times \hat{R}K^{-1}$ are fundamental matrices consistent with the recovered baseline, $ds(x,x',F)$ is the Sampson error. Eq.6.6 is parametrised by incremental changes such that $\hat{R} = R_{\Delta\theta_x\Delta\theta_y\Delta\theta_z}R$, $\hat{T}_{lu} = T_{lu} + (\Delta T^x_{lu},\Delta T^y_{lu},\Delta T^z_u)$ and $\hat{T}_{ru} = T_{ru} + (\Delta T^x_{ru},\Delta T^y_{ru},\Delta T^z_u)$ where R , $T_{lu} = \beta K^{-1}e'_{lu}$, and $T_{ru} = \beta K^{-1}e'_{ru}$ are the initial estimates with $\beta = K_{11}\|T_r\|/\|e'_{lu} - e'_{ru}\|$ assuming a single focal length. Note that both $\hat{T}_{lu},\hat{T}_{ru}$ share ΔT^z_u, which gives a total of 8 parameters to optimise, three for rotation and five for translation. To ensure that initial (T_{lu} and T_{ru}) move points to the same depth their z component is set to the same initial value, selecting either of T^z_{lu},T^z_{ru}. Eq.6.6 is minimised using the Levenberg-Marquardt algorithm to estimate $R_{\Delta\theta_x\Delta\theta_y\Delta\theta_z}$, $(\Delta T^x_{lu},\Delta T^y_{lu})$, $(\Delta T^x_{ru},\Delta T^y_{ru})$, and ΔT^z_u to update transformations and make them consistent with the three views and the stereo rig baseline T_r, i.e. recover the baseline. Finally, a second solution $\hat{R}',\hat{T}'_{lu},\hat{T}'_{ru}$ is computed by minimising again eq.6.6 using the previously estimated $R_{\Delta\theta_x\Delta\theta_y\Delta\theta_z}$ with $(\Delta T^x_{lu} = 0,\Delta T^y_{lu} = 0),(\Delta T^x_{ru} = 0,\Delta T^y_{ru} = 0),\Delta T^z_u = 0$ as initial estimates, and keeping the best solution. This is done to compensate for noisy initial estimates of T_{lu} and T_{ru}.

To further expand the explanation of baseline recovery, fig.6.2 shows how matching keypoints (x_{li},x_{ri},x^l_{ui}) are used to compute initial fundamental matrices (F_{lu},F_{ru}) (green and brown arrows in fig.6.2) later used to extract (R,T_{lu},T_{ru}) (blue arrows in fig.6.2), and finally compute $(\hat{R},\hat{T}_{lu},\hat{T}_{ru})$. Notice that if T_{lu}/T_{ru} are used as initial estimates then the line connecting e'_{lu} and e'_{ru} should result in a parallel line to the x axis when T_{lu} and T_{ru} are exact, but under realistic conditions and noise T'_r is an initial estimate of T_r. Because of this situation our algorithm uses Levenberg-Marquardt to find updated versions $\hat{T}_{lu}$ and $\hat{T}_{ru}$ (red arrows in fig.6.2). Finally, it is worth mentioning that the rotation/noise in the initial camera pose may cause swapped positions of T_{lu} and T_{ru} due to epipoles e'_{lu} and e'_{ru} locations, resulting in an inverted T'_r estimate. This may be consistent with the three views but it is incorrect. To handle this situation a second optimisation is done using the previously computed $R_{\Delta\theta_x\Delta\theta_y\Delta\theta_z}$ as initialisation to estimate updated $(\hat{T}'_{lu},\hat{T}'_{ru})$ (magenta arrows in fig.6.2). The new solution is compared to first estimate and the best solution kept (orange arrows in fig.6.2). We found this approach to be able to handle the inverted T'_r problem.

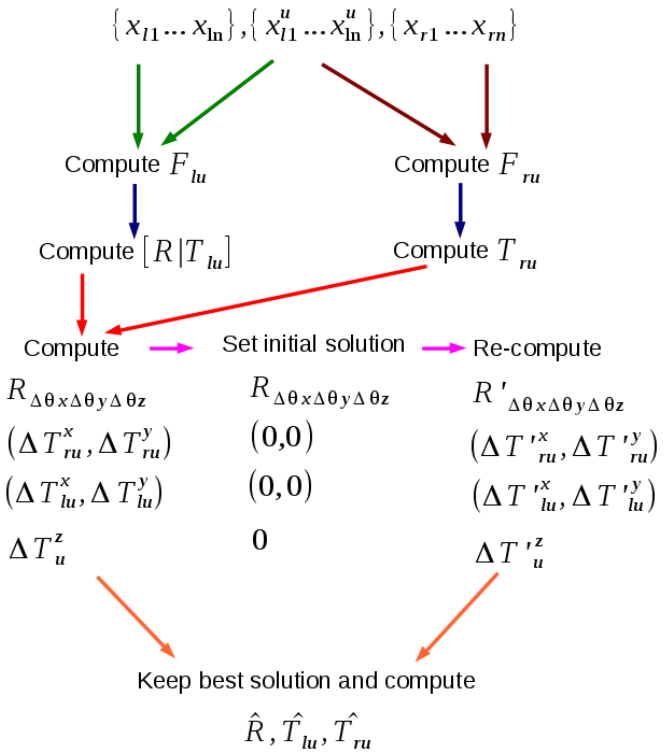


Figure 6.2: Baseline recovery algorithm

6.3.2 Computing initial estimates

The initial transformations (R, T_{lu}, and T_{ru}) and 3D points (X_i) are estimated by performing the following steps:

(1) Compute matching ASIFT [70] key points $(x_{li}, x_{ri}, x^u_{li})$ for views I_l, I_r and I^l_u.

(2) Compute the 3D points X^t_i from key points x_{li}, x_{ri}.

(3) Compute F_{lu} from x_{li}, x^u_{li}, and F_{ru} from x_{ri}, x^u_{li} using the normalised 8-point algorithm [30].

(4) Compute R from F_{lu} using the algorithm described in [30], and (T_{lu}, T_{ru}) as in sec.6.3.1.

In the case there are moving points it is necessary to remove them before recovering the baseline. This is accomplished by computing $v_i = x_{li} - x^u_{li}$ and cluster them using k-means. Then remove the smallest cluster and discard the second cluster if the ratio of the number of elements is lower than 5% of the largest cluster.

6.4 Trinocular pixel matching cost

The raw pixel matching cost (eq.5.10) described in chapter 5 is enhanced to include a third image. The new non-aggregated pixel similarity function is given by:

$$c_p(D_p) = \alpha c_p^1(D_p(p)) + c_p^2(D_p(p)) \tag{6.7}$$

$$\begin{aligned} c_p^1(D_p) = {} & \alpha_t \cdot min(|\nabla I_l(p) - \nabla I_r(p + D_p(p))|, \tau_{grad}^b) \\ & + (1-\alpha_t) \cdot min(|\nabla I_l(p) - \nabla I_u^l(\phi(PX))|, \tau_{grad}^t) \end{aligned} \tag{6.8}$$

$$\begin{aligned} c_p^2(D_p) = {} & \alpha_t \cdot min(\chi(I_l, I_r, p, D_p), \tau_{cen}^b) \\ & + (1-\alpha_t) \cdot min(\chi(I_l, I_u^l, p, D_p), \tau_{cen}^t) \end{aligned} \tag{6.9}$$

where I_l is the reference image, I_r and I_u^l are the target images. $c_p^1(D_p)$ is the truncated absolute differences of gradients using using ($\tau_{grad}^b = 3/255, \tau_{grad}^t = 1.33\tau_{grad}^b$). $c_p^2(D_p)$ is the truncated Hamming distance of the census transform using ($\tau_{cen}^b = 9, \tau_{cen}^t = 1.2\tau_{cen}^b$), χ computes the census transform and Hamming distance at pixel p with disparity plane D_p , α balances the pixel-wise cost influence. The trinocular cost influence is balanced with $\alpha_t = 0.80$ to prevent points in the image I_u^l from having too much influence in case they have changed position, i.e. reduce outliers. For parameter exploration see appendix D.

6.5 Converting a disparity map to optical flow

The baseline recovery algorithm described above is easily extended to convert a disparity map into optical flow by mapping each stereo correspondence in either I_l or I_r to I_u^l. The conversion process is done using the following steps:

(1) Use the baseline recovery algorithm to compute $[\hat{R}|\hat{T}_{lu}]$.

(2) Use the estimated disparity at point x_{li} to obtain 3D point X_i using existing calibration.

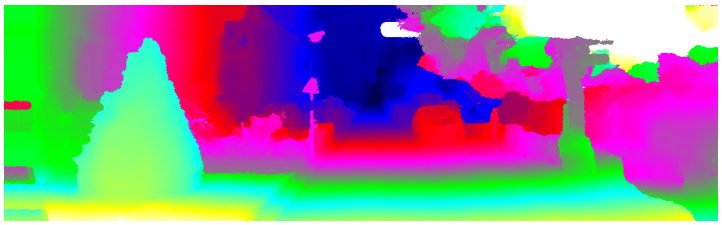
(3) Project 3D point X_i to x_{ui}^l.

(4) Compute $\vec{f}_{lu} = x_{ui}^l - x_{li}$. The vector $\vec{f}_{lu}$ is the optical flow.

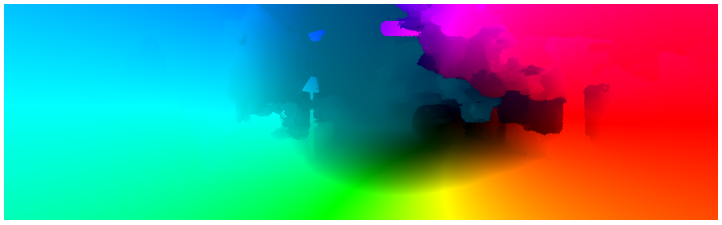
This conversion only works for static scenes such as those in KITTI 2012. Notice that no actual matching (x_{ui}^l, x_{li}) is computed. If a dynamic scene is presented this

optical flow may not be correct. Fig.6.3 shows an example of our resulting disparity map and its mapping to optical flow using the recovered motion. The colour map used for display is the same as in the KITTI 2012 benchmark. More detailed information can be found in the KITTI 2012 benchmark websites under the table entry *TBR*.

(a) KITTI image 1



(b) Disparity map



(c) Optical flow

Figure 6.3: Result for KITTI 2012 test image 1.

6.6 Experimental results

The baseline recovery algorithm is evaluated using the KITTI 2012 data set. The groundtruth depth maps are projected (see sec.6.5) to a third image displaced in time and then optical flow is computed using the recovered motion from our algorithm and compared with the groundtruth (tab.6.1). The proposed stereo matching approach was evaluated using the KITTI 2012 stereo disparity and optical flow benchmarks, and compared with a binocular algorithm (tab.6.2). Our algorithm is also compared to the state of the art competitors (the best performing convolutional neural network algorithms and the best performing algorithms not using convolutional neural networks,

same as in sec.5.12). Most of the algorithms compared appear in both data set evaluation tables. Our approach is among the top performers on the KITTI 2012 optical flow (tab.6.3) and stereo (tab.6.4) benchmarks (submitted as *TBR*). The parameters used are the same as those in chapter 5 for the *LPU* algorithm.

Algorithm	avg. init.	avg. ref.	time secs.
Our	6.47	0.52	0.23
6*PT*	9.16	0.53	152.91

Table 6.1: Baseline recovery accuracy. avg. init.: average error of initial solution; avg. ref.: average error of refined solution; time secs: average compute time.

Tab.6.1 compares our approach to recover the camera motion with the 6 points algorithm 6*PT* to recover 3 cameras [30]. The evaluation uses 40 images from KITTI 2012 and measures the average pixel displacement error of all pixels in the optical flow evaluated computed using our approach (using every 5th image from KITTI 2012). We report the error of the initial camera motion estimate on all images (*avg. init.*) and error after refinement (*avg. ref.*) using our approach. Our algorithm has lower error on initialisation and it is further reduced after refinement, whereas 6*PT* has a larger error on initialisation (even after using RANSAC) also it is slower as it optimises 24 vs. 8 parameters using our approach. The 6*PT* algorithm was refined using our approach.

Algorithm	%bad noc	%bad occ	avg. noc	avg. occ
Our	3.07	4.13	0.69	0.86
bino	3.22	4.12	0.72	0.86

Table 6.2: Trinocular vs. Binocular evaluation on KITTI 2012 training data set (40 images used). %bad noc: average percentage of wrong pixel in non-occluded areas; %bad occ: average percentage of wrong pixel in all pixels including occluded areas; avg. noc: average disparity error in non-occluded areas; avg. occ: average disparity error in all pixels including non-occluded areas.

Tab.6.2 shows that using the baseline recovery and the proposed trinocular cost gives better results in non-occluded areas while giving approximately the same performance in occluded areas. Tab.6.2 also shows that the baseline recovery algorithm

works as intended in images with no moving objects (see fig.6.4). In the KITTI benchmark, our algorithm ranks 11^{th} (out of 85), and 15^{th} (out of 89) for KITTI 2012 optical flow and stereo respectively. The evaluation on KITTI 2012 proved challenging as coloured intensity images that are not properly aligned to the ground truth shape image, causing problems for the aggregation algorithm. The top performing competitors achieve better performance by: using scene specific content to eliminate ambiguities (e.g. cars in Disp.v2), training specifically for the data sets (e.g. MCNCC, SDF), or using 2-3 image pairs to estimate disparity (e.g. PRSM, OSF). By contrast the proposed algorithm achieves top performing results in multiple data set by: using only the left, right and $t+1$ left images, using baseline recovery, not using scene specific features (e.g. cars), and not computing optical flow directly but instead mapping disparities using the recovered motion.

Algorithm	%bad noc	%bad occ	avg. noc	avg. occ
Our^{11th}	4.24	7.50	0.9	1.5
PRSM^{1st}	2.46	4.23	0.7	1.0
OSF^{5th}	3.47	6.34	1.0	1.5
SDF^{9th}	3.80	7.69	1.0	2.3

Table 6.3: Optical flow evaluation on test data (194 images). Non-anonymous entries are used for comparison: PRSM[96], OSF[66], SDF[3].

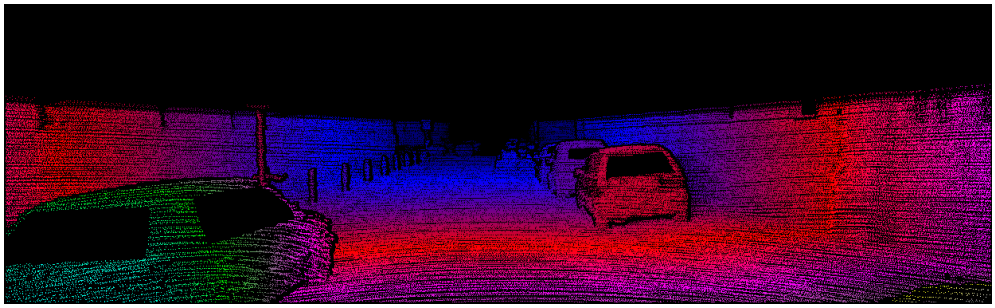
Algorithm	%bad noc	%bad occ	avg. noc	avg. occ
Our^{15th}	3.09	4.29	0.70	0.90
Disp. v2^{4th}	2.37	3.09	0.70	0.80
MCNCC^{5th}	2.43	3.63	0.70	0.90
PRSM^{10th}	2.78	3.00	0.70	0.70
OSF^{19th}	3.28	4.07	0.80	0.90

Table 6.4: Disparity evaluation on test data (194 images). Non-anonymous entries are used for comparison: Disp. v2[28], MCNCC[97].

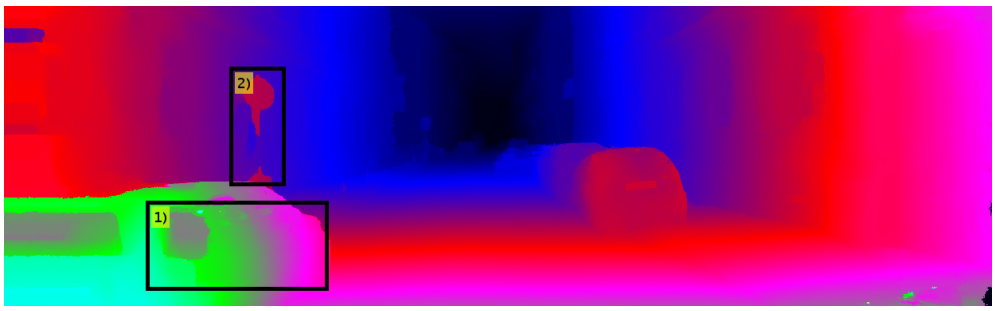
Fig.6.4 shows the comparison of the binocular algorithm (*LPU*, see chapter 5) and the proposed trinocular approach (*TBR*). The are two comparison regions 1) reflective

, and 2) fine details. In both cases the trinocular approach obtains shows improvement by keeping the traffic pole, and reducing noise in the reflective region.

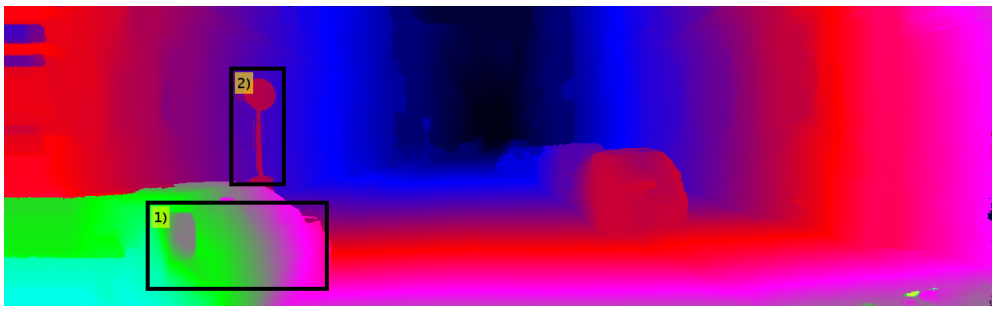
(a) KITTI image 15



(b) Groundtruth



(c) LPU

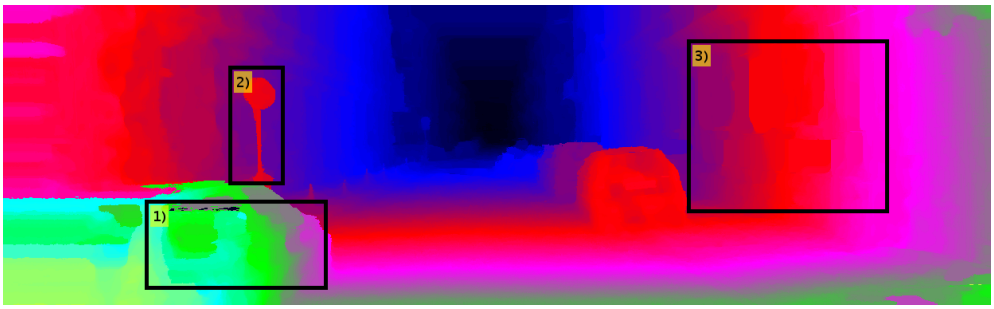


(d) TBR

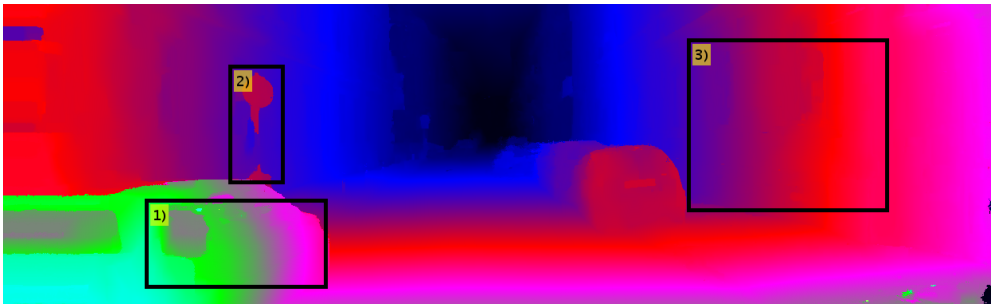
Figure 6.4: LPU vs. TBR for KITTI 2012 training image 15.

Finally, Fig.6.5 shows the comparison of the binocular algorithms *LCU* (see chap-

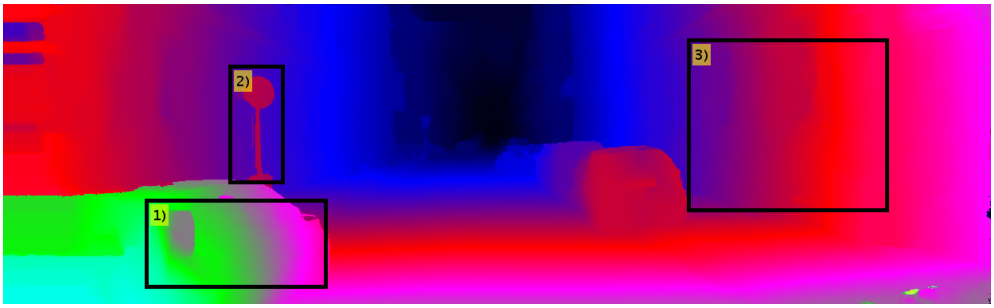
(a) KITTI image 15



(b) LCU



(c) LPU



(d) TBR

Figure 6.5: LCU vs. LPU vs. TBR for KITTI 2012 training image 15.

ter 4), *LPU* (see chapter 5) and the proposed trinocular approach (*TBR*). The are three comparison regions 1) reflective, 2) fine details, and 3) saturated. For the regions 1) and 3) *LPU* and *TBR* give better results in particular in the saturated region which

is almost textureless, this is due to the use of the uniqueness term. While in the fine details region 1) TBR and LCU recover the thin pole.

6.7 Critical analysis discussion

The results presented so far have shown that the proposed algorithm delivers some of the best performing results across different data sets. However, our approach still faces the following issues:

- **Fixed camera motion**: The proposed algorithm recovers the dominant motion of a third camera. However, it is assumed that the motion is rigid, and other possible cameras that explain dynamic scenes are not recovered. Under this circumstance using the recovered camera to compute the trinocular cost will work for static objects (e.g. the background), but is likely to introduce noise in the cost of moving objects (e.g. cars) leading to inaccurate disparity estimates.

- **Binocular uniqueness term**: The proposed algorithm only uses a binocular uniqueness penalty, which does not take into account the additional third view. Although not immediately obvious this is likely to lead to disparity estimates that are not unique in the third view.

- **Cost scale issues**: The proposed cost function is computed using the recovered motion, which implies that there are changes in scale in the third view depending on the translation and disparity being evaluated. This may result in a noisy cost that can lead to poor results if not weighted and truncated. A mechanism to compute a scale penalty would need to be implanted to address this issue.

- **Fronto-parallel restriction**: The baseline recovery algorithm only works for rectified fronto-parallel cameras, which restricts its applications to stereo matching only.

6.7.1 Detailed analysis of KITTI 2012 test image 1

Here we make a visual comparison of our results with the competitors of tab.6.4. Fig.6.6 shows the left reference image "1" with overlaid boxes to evaluate three types of regions 1) planar bias, 2) fine details, 3) reflective. Our result is shown in fig.6.7 and is compared to Disp. v2 (fig.6.9), MCNCC (fig.6.11), PRSM (fig.6.13), and OSF (fig.6.15).

Figure 6.6: Kitti image “1” (left image).

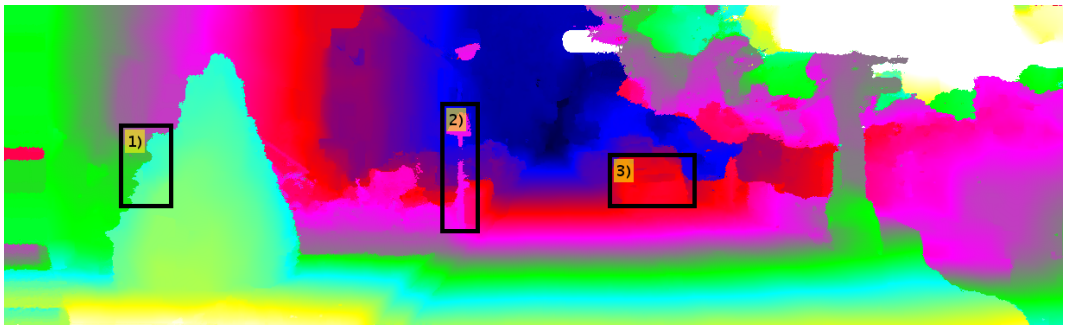


Figure 6.7: Kitti image disparity image “1” (our result).

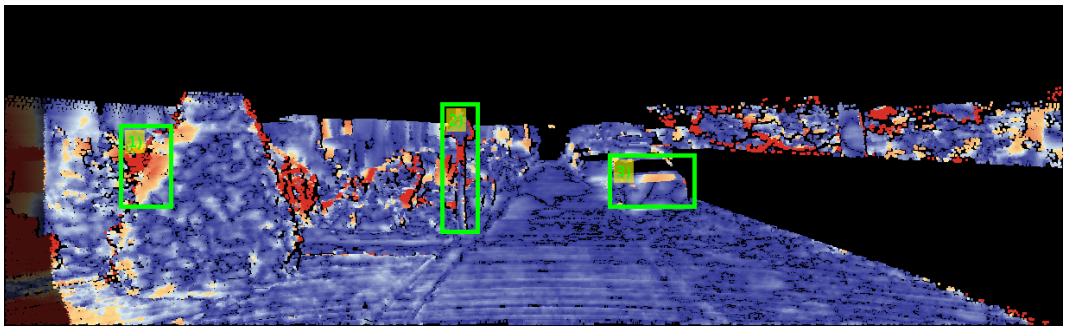


Figure 6.8: Kitti image error map “1” (our result).

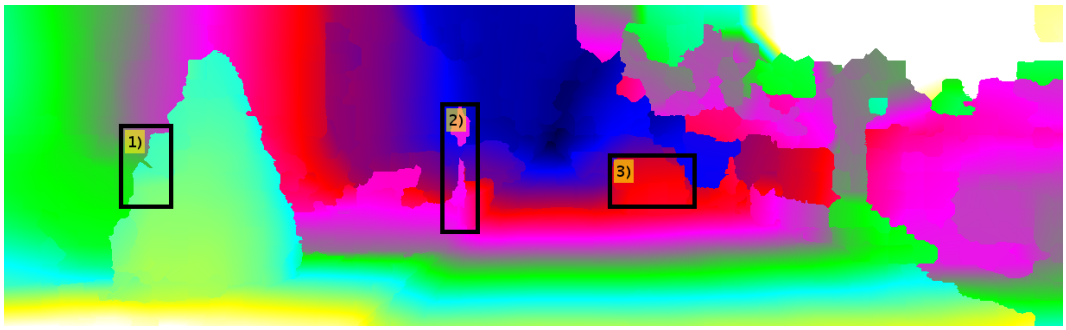


Figure 6.9: Kitti image disparity image “1” (Disp. v2 result).

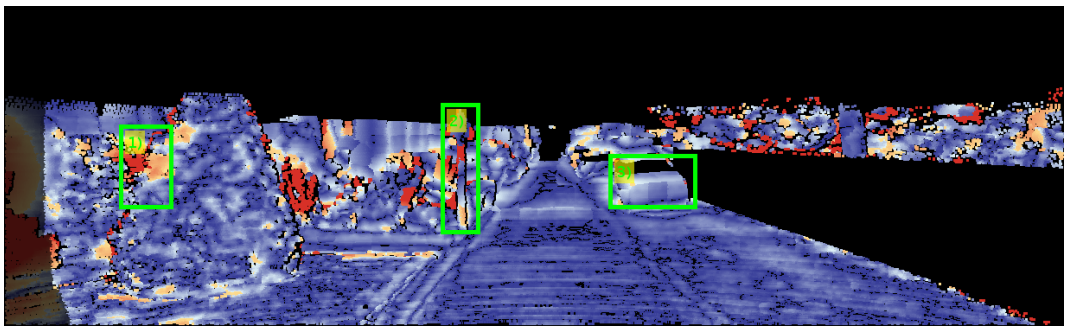


Figure 6.10: Kitti image error map “1” (Disp. v2 result).

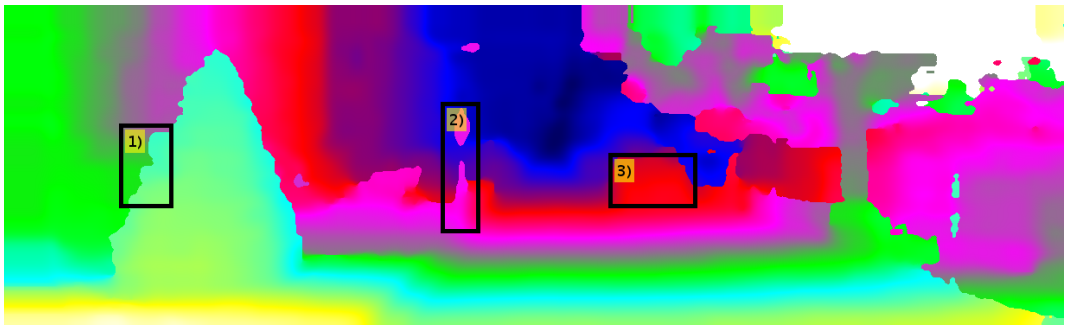


Figure 6.11: Kitti image disparity image “1” (MCNCC result).

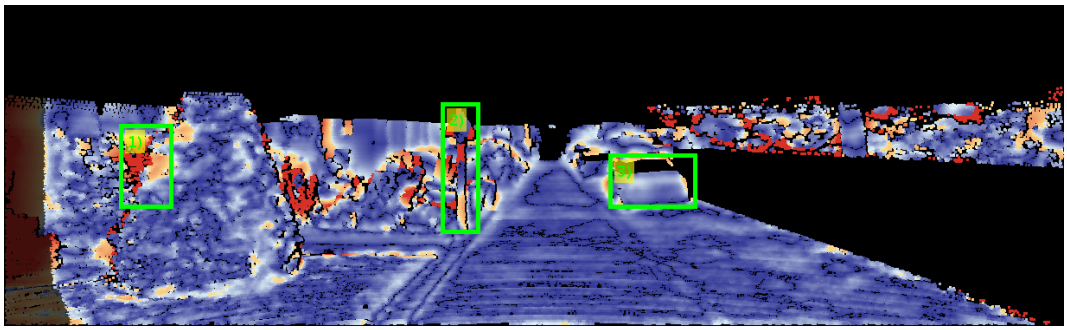


Figure 6.12: Kitti image error map “1” (MCNCC result).

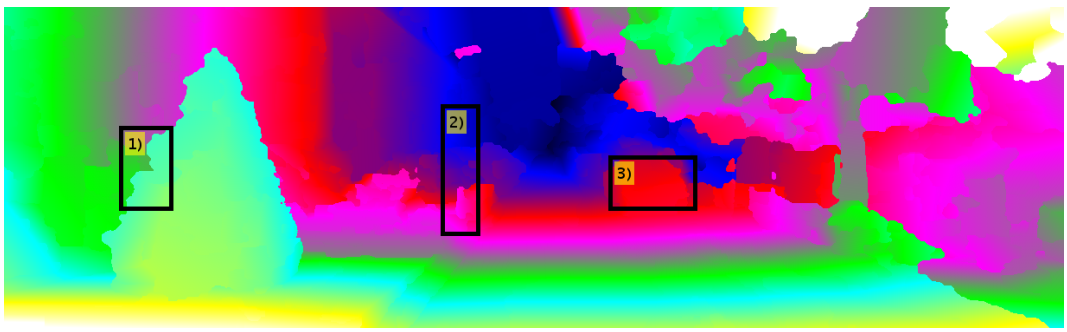


Figure 6.13: Kitti image disparity image “1” (PRSM result).

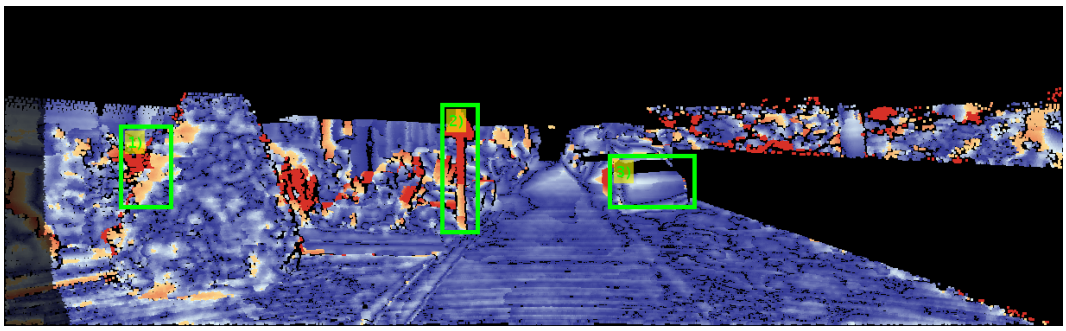


Figure 6.14: Kitti image error map “1” (PRSM result).

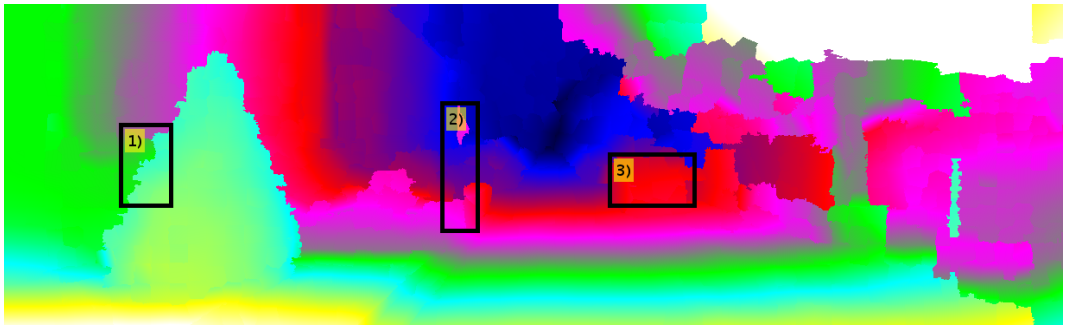


Figure 6.15: Kitti image disparity image “1” (OSF result).

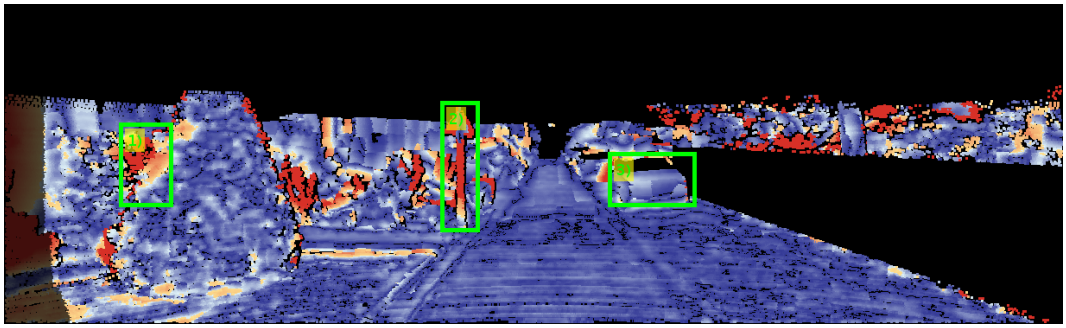


Figure 6.16: Kitti image error map “1” (OSF result).

1) In the planar bias region all algorithms our (fig.6.8), Disp. v2 (fig.6.10), MCNCC (fig.6.12), PRSM (fig.6.14), and OSF (fig.6.16) have a large error on the pine tree. This is explained by a strong planar bias, and the disparity overgrowth to the background.

2) In the fine details region our algorithm (fig.6.7) recovers the triangle shaped traffic signal, and partially recovers the pole. Disp. v2 (fig.6.9) and MCNCC (fig.6.12) recover the same regions but they are not very well defined. PRSM (fig.6.13), and OSF (fig.6.15) fail to recover this region.

3) In the reflective region our algorithm (fig.6.8) shows a higher error, due to the use of a per-pixel disparity plane. Whereas the algorithms Disp. v2 (fig.6.10), MCNCC

(fig.6.12), PRSM (fig.6.14), and OSF (fig.6.16) obtain better results using 3D scene specific priors, data set specific training, or a segment based cost function.

6.7.2 Detailed analysis of KITTI 2012 test image 12

Here we make a visual comparison of our results with the competitors of tab.6.4. Fig.6.17 shows the left reference image "12" with overlaid boxes to evaluate low contrast regions in boxes 1) and 2). Our result is shown in fig.6.18 and is compared to Disp. v2 (fig.6.20), MCNCC (fig.6.22), PRSM (fig.6.24), and OSF (fig.6.26).

Figure 6.17: Kitti image "12" (left image).

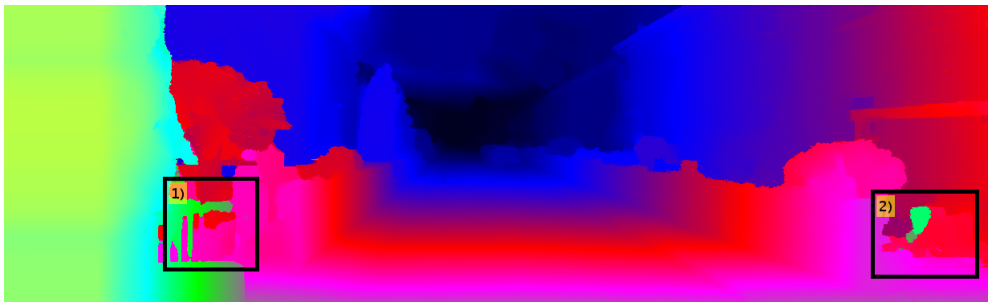


Figure 6.18: Kitti image disparity image "12" (our result).

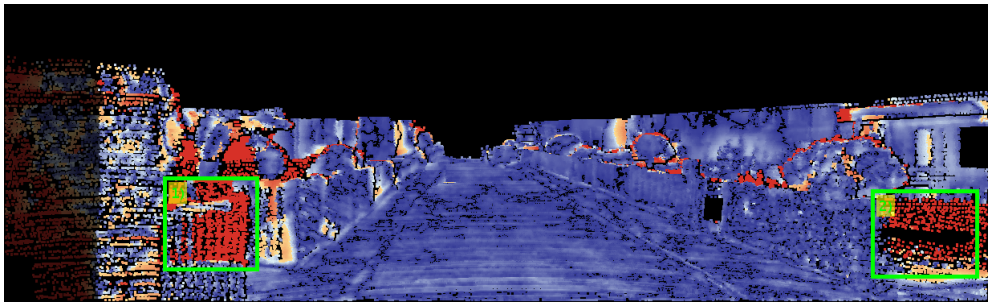


Figure 6.19: Kitti image error map "12" (our result).

In the low contrast regions **1)** and **2)** our algorithm (fig.6.19), Disp. v2 (fig.6.21), MCNCC (fig.6.23), and OSF (fig.6.27) fail to recover the low contrast and thin fence. By contrast PRSM (fig.6.25) shows the lowest error, which is explained by the use of multiple frames (3 stereo pairs) and segment based cost function.

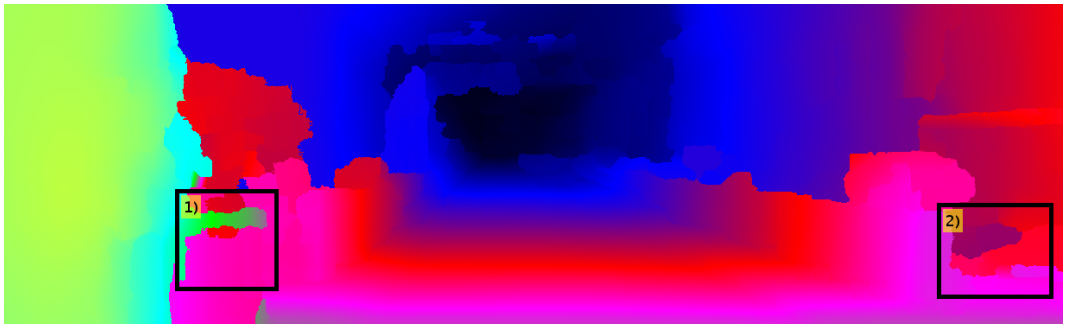


Figure 6.20: Kitti image disparity image “12” (Disp. v2 result).

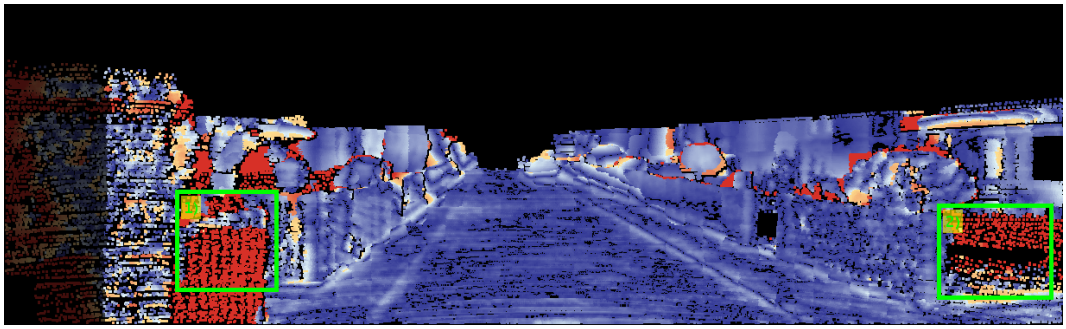


Figure 6.21: Kitti image error map “12” (Disp. v2 result).

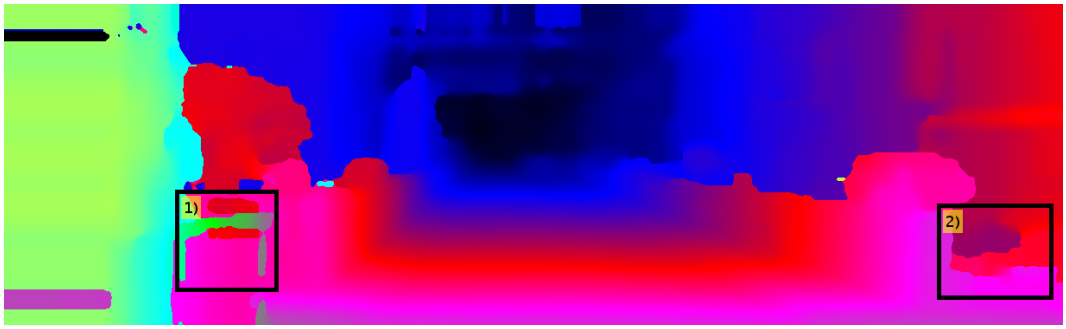


Figure 6.22: Kitti image disparity image “12” (MCNCC result).

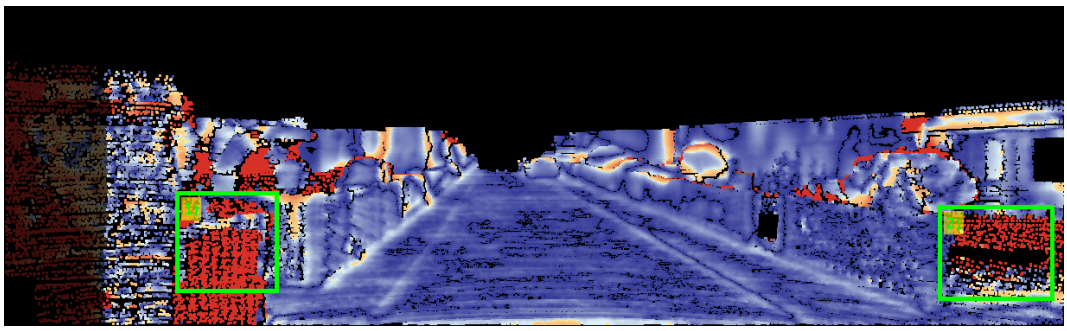


Figure 6.23: Kitti image error map “12” (MCNCC result).

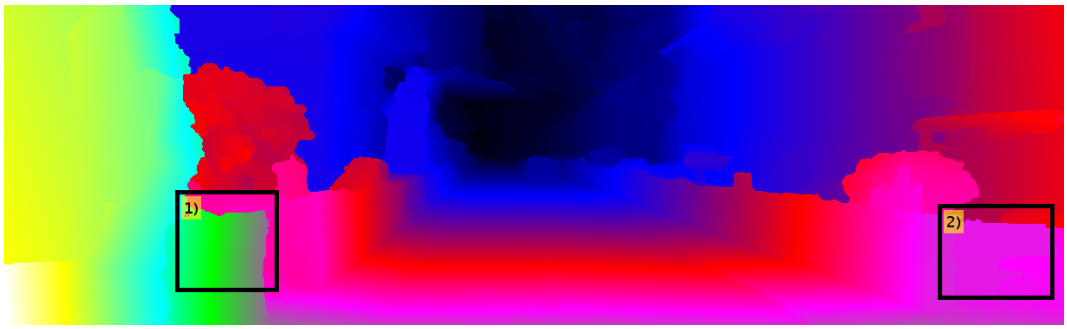


Figure 6.24: Kitti image disparity image “12” (PRSM result).

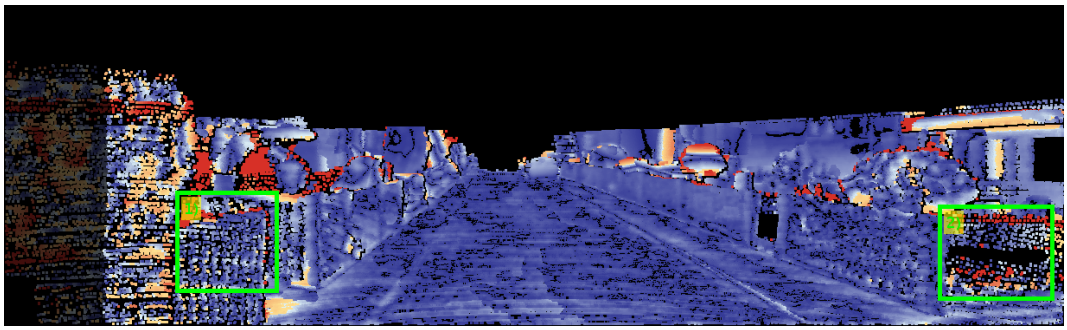


Figure 6.25: Kitti image error map “12” (PRSM result).

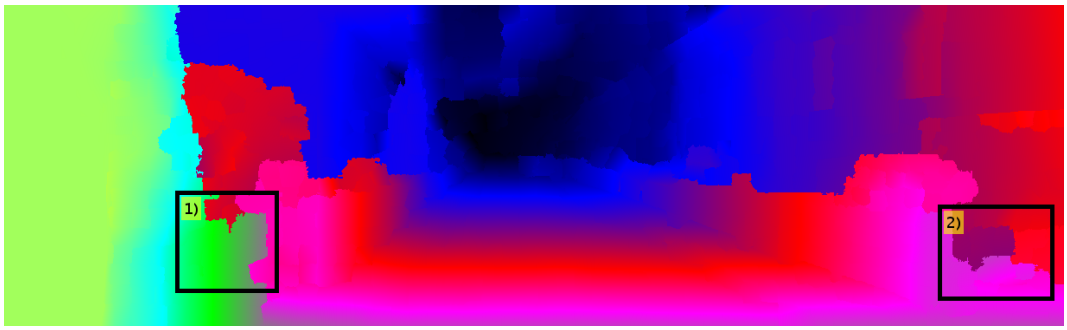


Figure 6.26: Kitti image disparity image “12” (OSF result).

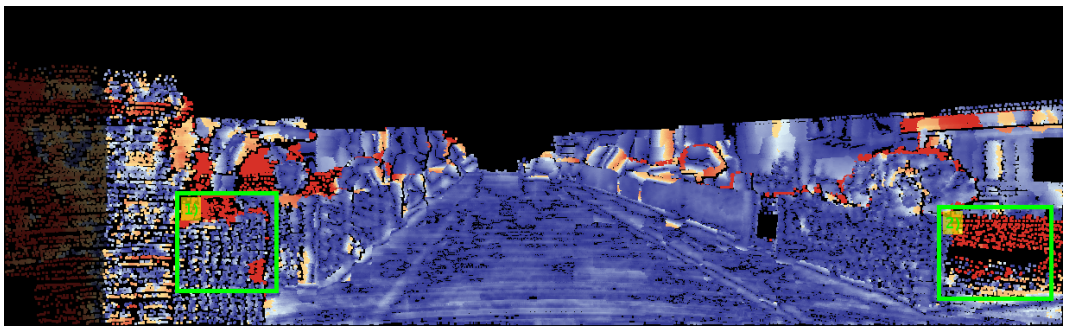


Figure 6.27: Kitti image error map “12” (OSF result).

6.7.3 Detailed analysis of KITTI 2012 test image 19

Here we make a visual comparison of our results with the competitors of tab.6.4. Fig.6.28 shows the left reference image “19” with overlaid boxes to evaluate three types of regions 1) textureless surface, 2) fine details, 3) reflective surface. Our result is shown in fig.6.29 and is compared to Disp. v2 (fig.6.31), MCNCC (fig.6.33), PRSM (fig.6.35), and OSF (fig.6.37).

Figure 6.28: Kitti image “19” (left image).

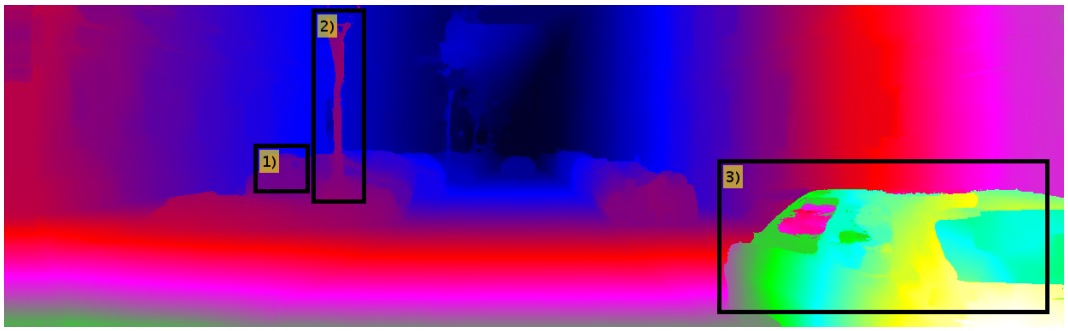


Figure 6.29: Kitti image disparity image “19” (our result).

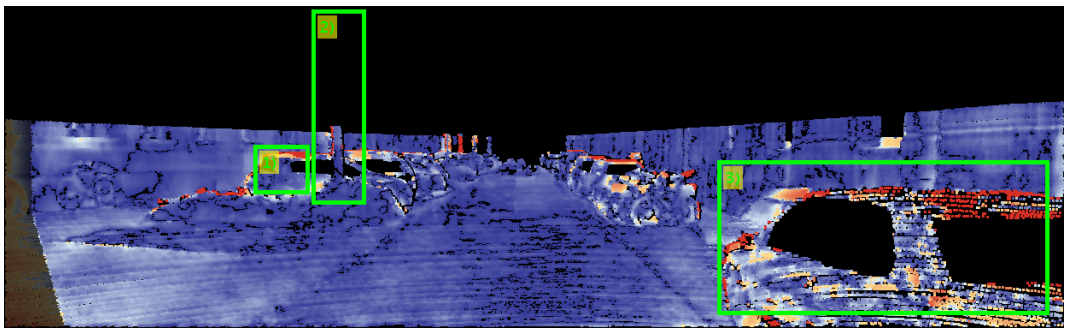


Figure 6.30: Kitti image error map “19” (our result).

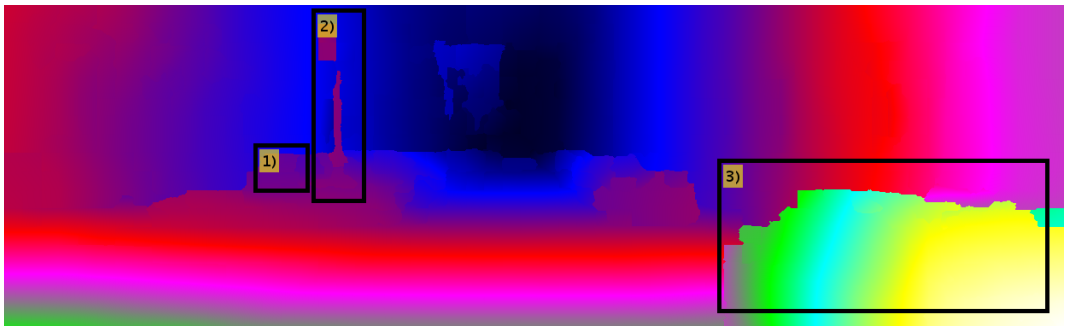


Figure 6.31: Kitti image disparity image “19” (Disp. v2 result).

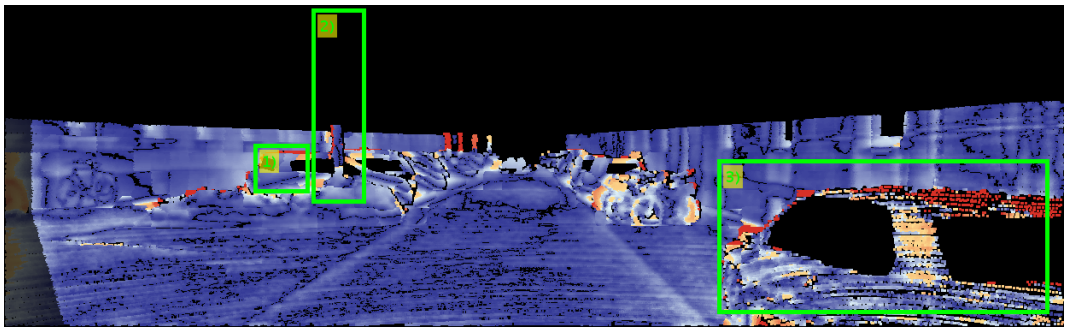


Figure 6.32: Kitti image error map “19” (Disp. v2 result).

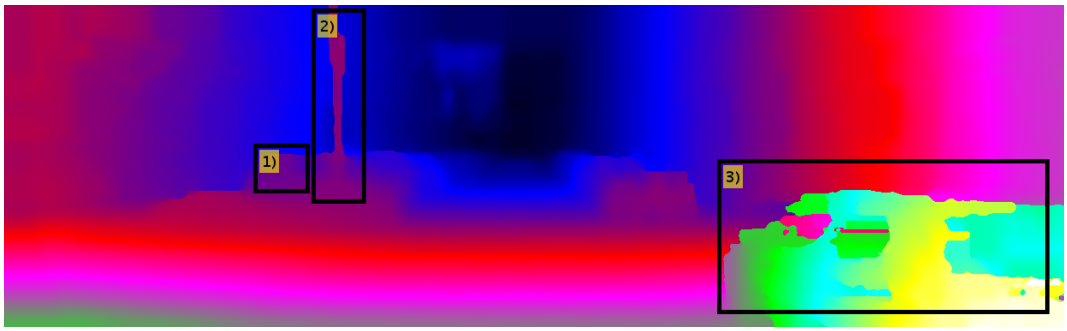


Figure 6.33: Kitti image disparity image “19” (MCNCC result).

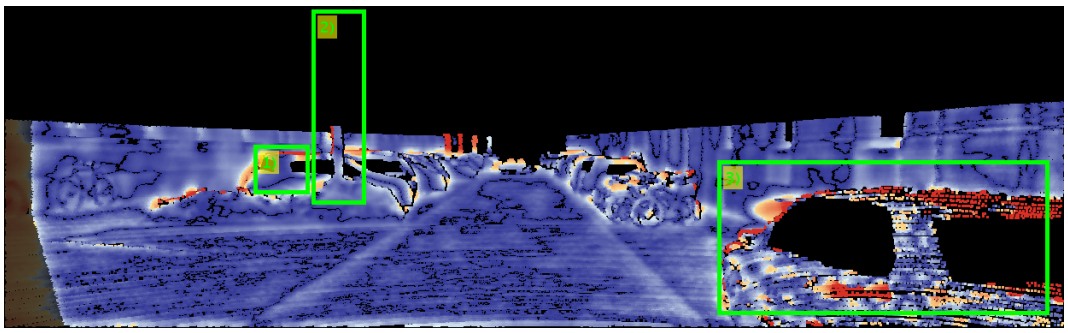


Figure 6.34: Kitti image error map “19” (MCNCC result).

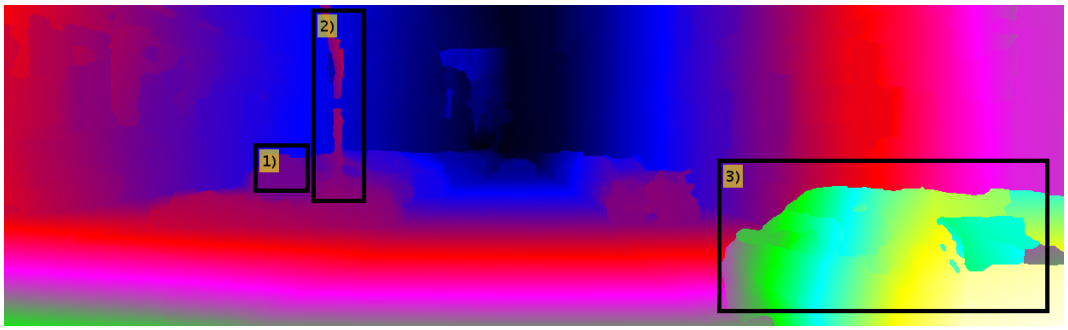


Figure 6.35: Kitti image disparity image “19” (PRSM result).

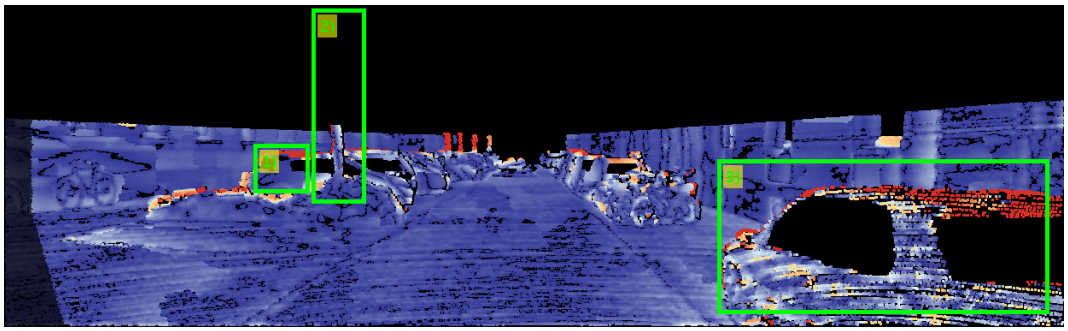


Figure 6.36: Kitti image error map “19” (PRSM result).

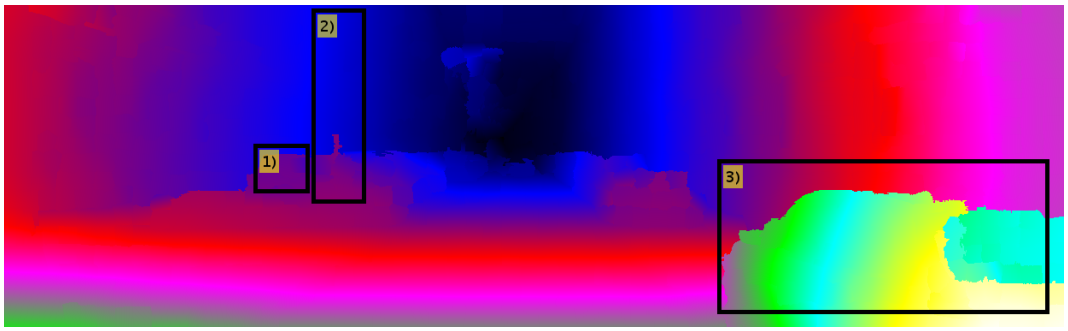


Figure 6.37: Kitti image disparity image “19” (OSF result).

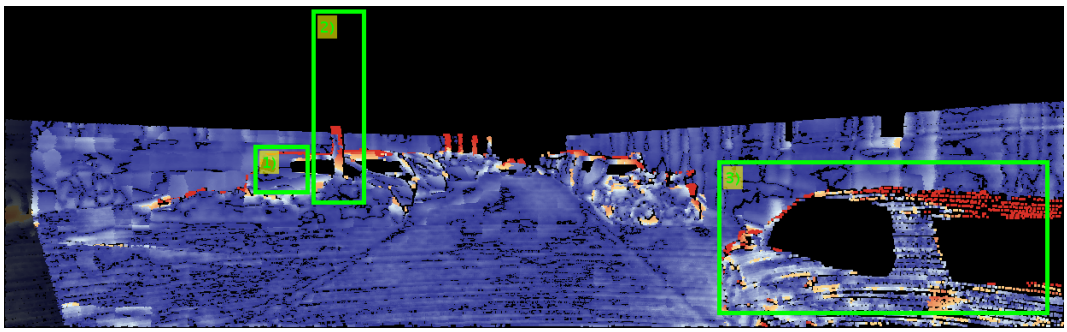


Figure 6.38: Kitti image error map “19” (OSF result).

1) In the textureless region our algorithm (fig.6.30), Disp. v2 (fig.6.32), MCNCC (fig.6.34), and PRSM (fig.6.36) all have problems with the textureless car roof resulting in high error in that area. By contrast OSF (fig.6.38) have a lower error, which can be explained by the use of disparity plane segments.
2) In the fine detail region our algorithm (fig.6.29) recovers the tree trunk and its branches. Disp. v2 (fig.6.31), MCNCC (fig.6.33), and PRSM (fig.6.36) achieve partial success in recovering the tree and branches. OSF (fig.6.38) fails to recover any detail in the same region.
3) In the reflective region our algorithm (fig.6.19), Disp. v2 (fig.6.21), MCNCC (fig.6.23), and OSF (fig.6.27) fail to recover the car roof as it is reflective which creates an ambiguous cost, and therefore it is difficult to estimate disparity.

6.7.4 Detailed analysis of performance using groundtruth initialisation

As described previously our algorithm can use either a precomputed disparity map or random initialisation. Taking advantage of this property we have initialised our algorithm using groundtruth disparity maps, and evaluated what are the effects of this on the algorithm performance.

Algorithm	%bad noc	%bad occ	avg. noc	avg. occ
GTI	2.98	3.96	0.66	0.82
TBR	3.07	4.13	0.69	0.86

Table 6.5: Comparative table proposed algorithm initialised with a precomputed disparity map (TBR) vs. groundtruth (GTI).

Tab.6.5 shows that using the groundtruth (GTI) as initialisation results in less error for all metrics on the KITTI training data set (using every 5th image). However, it would be expected that the error should be even lower. This can be explained by several factors 1) similarity cost noise, 2) smoothness model, and 3) reflective areas. In other words our model gives good results, but it is only an approximation of the process that produced the groundtruth.

Figure 6.39: Kitti image “180” (left image).

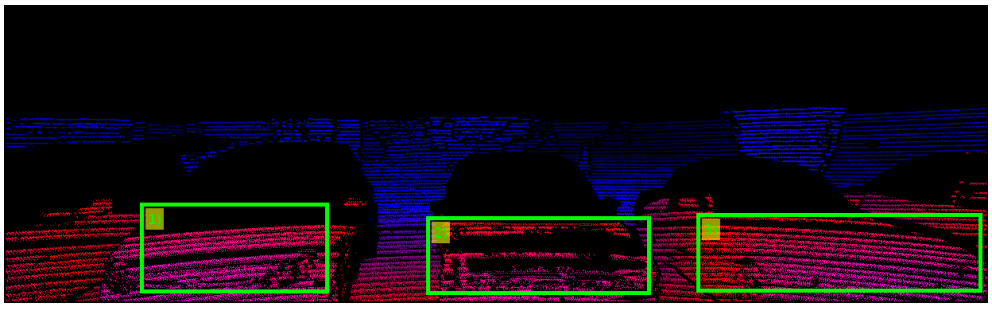


Figure 6.40: Kitti image “180” groundtruth.

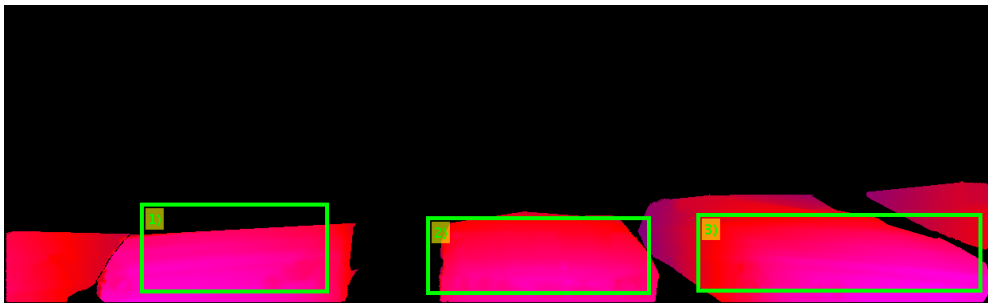


Figure 6.41: Kitti image “180” groundtruth for cars.

To further illustrate the effects of using groundtruth as initialisation we analyse the KITTI training image “180”. Fig.6.39 shows the left reference image, fig.6.40 is the groundtruth for non-reflective areas (the official evaluation for the leader table),

fig.6.41 is the groundtruth for cars (fitted using a 3D CAD model, see the KITTI benchmark website). Staring in fig.6.39 the are three regions evaluating the cars' disparity error in the areas enclosed by a box. Fig.6.42 is the result obtained by initialising with groundtruth, and fig.6.43 is the coloured error map. Fig.6.42 is the result obtained using our algorithm (TBR), and fig.6.45 is the coloured error map.

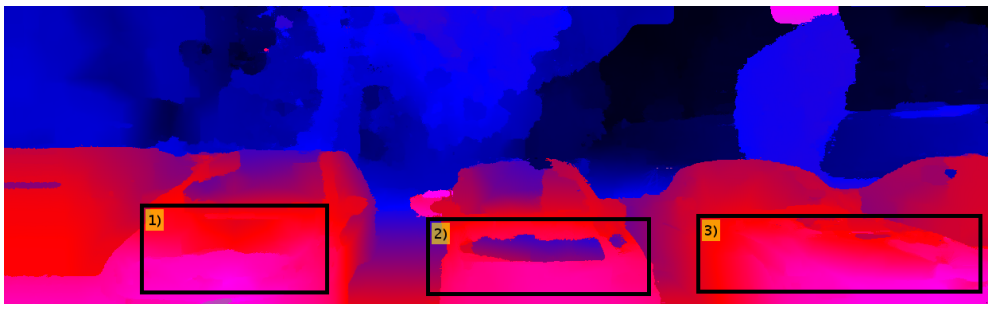


Figure 6.42: Kitti image "180" groundtruth initialised result.

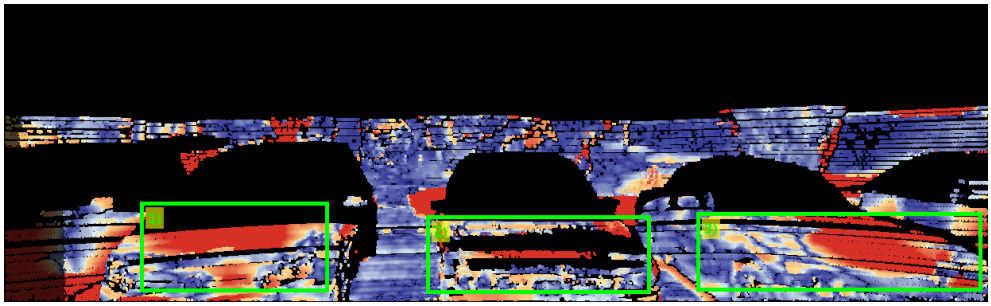


Figure 6.43: Kitti image "180" groundtruth initialised result error.

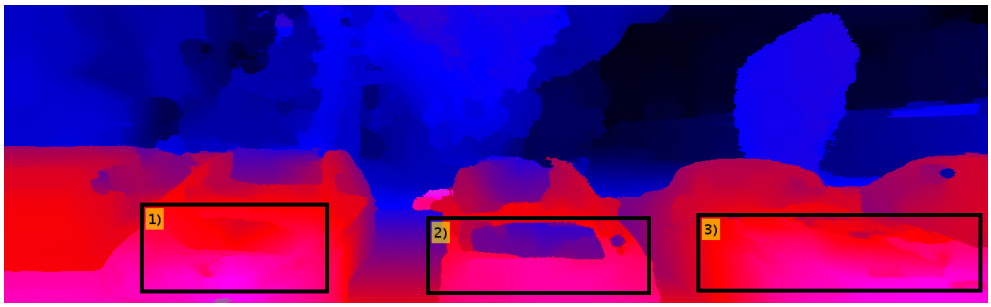


Figure 6.44: Kitti image "180" non-initialised result.

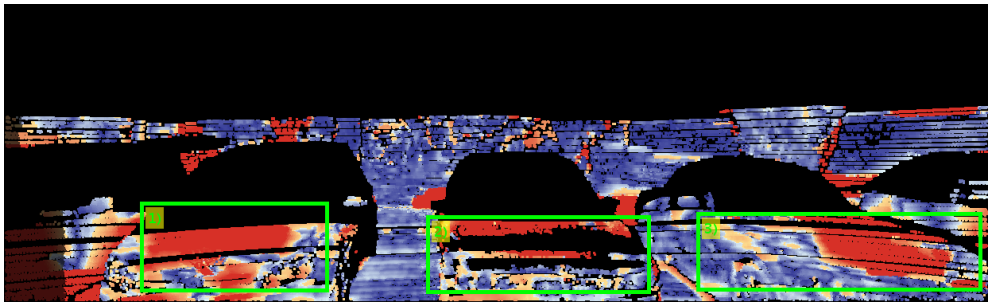


Figure 6.45: Kitti image "180" non-initialised error result.

In both initialised (fig.6.43) and non-initialised (fig.6.45) results there is large error in the surface of cars (i.e. reflective region). However, the groundtruth initialised algorithm has a lower error, but not completely low as it would be expected. This is caused due to the noisy and ambiguous similarity costs that are characteristic of reflective regions.

6.8 Summary

The baseline recovery algorithm is to the best of our knowledge a novel technique to recover camera motion that integrated easily in a DPI dense trinocular algorithm. The proposed algorithm successfully exploits the displacement of a third image to accurately recover camera motion and also delivers high performing optical flow and disparity estimation results even though only the general motion is computed, no pre-computed optical flow is used, and no convolutional neural networks (e.g. [97, 28, 3]) or prior 3D models (e.g. cars) are used.

The experiments described in this chapter have shown that using a third view reduces the error in the final disparity estimation, but most interestingly our optimization technique is faster, and more accurate than the well known 6 point algorithm. This can be explained by the fact that our approach successfully exploits the prior knowledge about camera calibration to refine the initial estimate of the camera pose. The results presented only evaluated the improvement in the disparity estimation, and the optical flow accuracy. However, it is left as future work to use the proposed algorithm to do SLAM and evaluate the accuracy of the motion estimates.

Chapter 7

Practical considerations for 3D plane labelling and trinocular matching

The core ideas of the proposed plane labelling algorithms are described from a theoretical point of view in chapters 5 and 6 . However, there are practical considerations that need to be taken into account when implementing the LPU and TBR algorithms. The following implementation details are covered/clarified:

- Implementation of the content aware adaptive aggregation strategy in a multi-core hardware architecture.
- Implementation of TRW-S in a multi-core hardware architecture.
- Implementation of trinocular keypoint matching.

From these points our parallel implementation of TRW-S is to the best our knowledge the only software that exploits modern CPU/GPU architectures and vectorised floating units.

7.1 Content aware adaptive aggregation considerations

The content aware function from chapter 5 (eq.5.9) depends heavily on pixel interpolation, and furthermore the raw pixel cost uses pixel-wise(gradient) and block-matching costs, which are aggregated using two different algorithms. This poses the following problems:

1. Expensive pixel interpolation.

2. Aggregation of pixel-wise and block-matching costs that have different ranges.

3. Issues with the scaling factor for the pixel-wise part.

4. Possible generation of *NaN* values due normal vector orientations.

In practice using 3D planes per pixel to represent disparity and evaluate eq.5.9 requires pixel interpolation whose associated cost of sampling a potential matching pixel results in at least in two memory accesses of pixels in the target image (if using linear interpolation). Furthermore the access may not be done in a regular pattern thus making the computation of eq.5.9 entirely dependant on the memory bandwidth of the hardware used. Implementing the proposed aggregation algorithm in a *CPU* can become impractical for the reasons mentioned above. We overcome this problem by using multiple *GPU*s programmed using *CUDA*, which provides the following advantages:

- Parallel architecture that allows multiple pixels to be processed simultaneously.
- High memory bandwidth that makes interpolation cheaper.
- Specialised cache for 2D memory access.
- Native hardware support for pixel interpolation.

Aggregating a pixel-wise and block matching cost together may result in the unwanted effect of the pixel-wise part losing some influence in the final aggregated cost. This effect is caused by the different range of each cost function, which may not be immediately obvious. To exemplify this problem, consider the following condition $[0, \tau_{grad}]$ with $\tau_{grad} \leq 1$ is the range of the pixel wise part, and $[0, \tau_{cen}]$ with $\tau_{cen} \leq 25$ is the range of the block-matching part assuming a census transform of 5×5. Each time the cost is aggregated together it keeps growing and due to the floating point hardware implementation (most noticeable in *GPU*s), the smaller contribution of the pixel wise part may be lost during rounding and normalisation. An intuitive solution would be to scale the values, but this results in further loss of accuracy due to the multiplication needed. The solution adopted to address this problem was to aggregate separately the pixel-wise and block-matching and scaling at the end of the aggregation algorithm.

The disparity produced by a 3D plane is dependent on its normal vector, which can result in a *NaN* number (i.e. disparity) depending on the (x, y) coordinates used to evaluate the plane. This is a non-trivial situation as the disparity is unknown, and thus

it is undefined which pixel to sample. To handle this problem it was chosen to flush the *NaN* values to zero (rounding option available in many compilers as an optimisation option) resulting in zero disparity, and possibly in a high matching cost, which would reduce the chances of selecting a plane which produces *NaN* values.

7.1.1 *GPU* specific algorithm

Eq.5.9 was implemented in a *GPU* because it is easy parallelise and multiple *GPU*s can be used to accelerate its computation. Fig.7.1 shows how each plane hypothesis H is divided into stripes (regions) H_{ri} of $16 \times W$ pixels, with each H_{ri} sub-divided into 16×32 pixels. Each H_{ri} is processed in three stages: 1) copy H_{ri} to *GPU* n, 2) compute matching cost, 3) compute uniqueness term and add to cost. Our implementation uses *CUDA* streams to overlap memory transfer with computation. Additional *GPU*s can be used to further accelerate the computation of the matching cost as shown in fig.7.1.

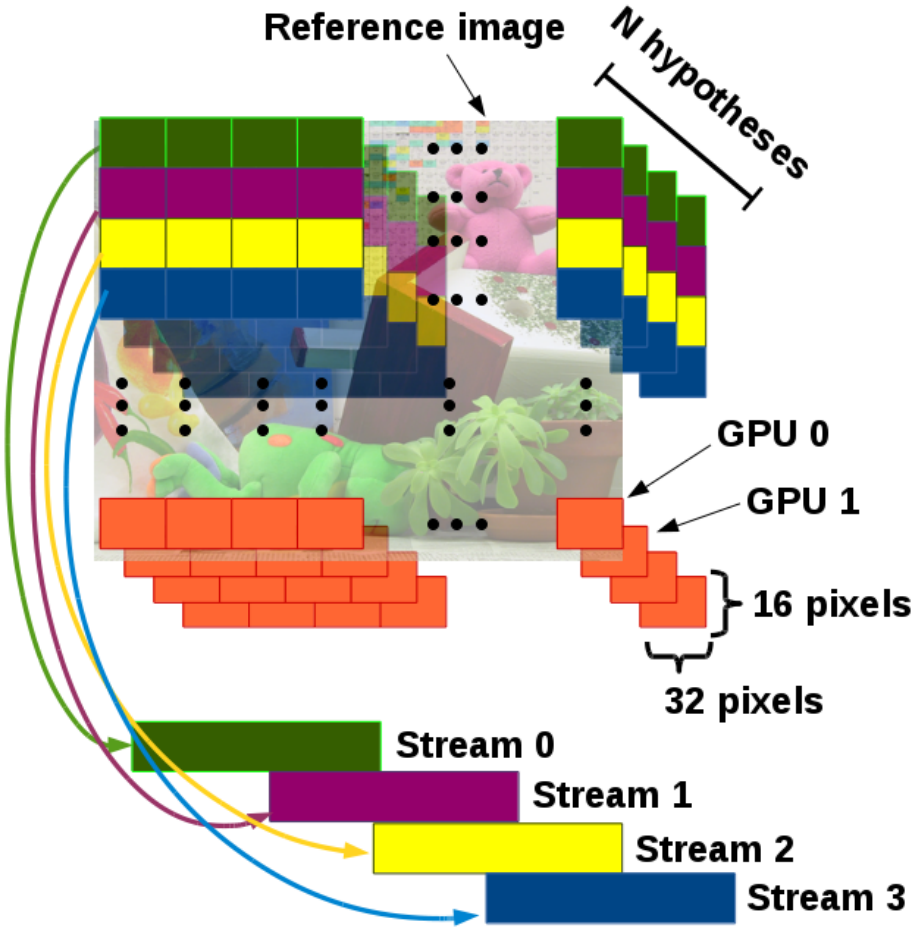


Figure 7.1: Computing cost with multiple GPUs and steams.

The reason to divide the hypotheses into 16×32 blocks is two fold: ensuring

enough data proximity to exploit 2D cache access optimisations, and eliminate register spillage that can slow down the execution of the code. Alg.1 describes how a single *GPU* divides a hypothesis H into multiple regions H_{ri}, and each one of them is processed by *CUDA* stream *ri*. Each procedure is asynchronous which means that the *CPU* submits a request to the *GPU* driver, which executes when possible and returns control to the *CPU* immediately. Using several streams *ri* allows one to use the multiple command queues available in a modern *GPU*, which means that a copy command can be executed in the background while the *GPU* is executing a procedure, and when completed the background copy may have already been performed thus saving some transfer/compute time. Alg.1 uses multiple separated loops, which was done to exploit the multiple transfer modules available in a modern *GPU*. Note that the images to match are shared among the different regions, and only the hypotheses are divided.

Algorithm 1 Computing adaptive matching cost

procedure $C =$ `ADAPTIVECOST`(H, I_1, I_2) ▷ compute binocular cost
  for each $H_{ri} \in H$ **do** ▷ (H, I_1, I_2) are padded, and divided into $16 \times W$ regions
    `copytogpu`$[ri]$`(`H_{ri}`)` ▷ Asynchronous procedure returns to CPU.
  end for
  for each $H_{ri} \in H$ **do**
    $C_{ri} \leftarrow$`adaptivecostgpu`$[ri]$`(`H_{ri}`,` I_1`,` I_2`)` ▷ Asynchronous procedure
  end for
  for each $H_{ri} \in H$ **do**
    `copytocpu`$[ri]$`(`C_{ri}`)` ▷ Asynchronous procedure
  end for
  `waitforgpu()` ▷ Wait until all asynchronous procedures are done
  return C ▷ The computed cost
end procedure

7.2 Practical considerations for inference with TRW-S

The inference algorithm used in this thesis is *TRW*-S, which was chosen due to fact that it can minimise pair-wise functions with non-submodular smoothness terms, and in general it is guaranteed to find lower energy configurations when compared to α-expansion *GC*. However, the sequential nature of *TRW*-S represents a problem especially when using a smoothness term that can not be computed with a distance trans-

form, e.g the $O(N^2)$ smoothness term used. Under this circumstance the computational time becomes a major concern. In order to address this issue we have implemented the parallel update scheduler from [12], which exploits the raster scan order characteristic of TRW-S to update messages diagonally allowing messages to be computed in parallel. The implementation of [12] is done in an *FPGA* using integer valued disparity, fronto parallel planes, and a Potts model for the smoothness term. By contrast our implementation is software based and can be used in both *CPU* and *GPU*, and uses 3D disparity planes with a curvature based smoothness term. To the best of our knowledge the implementation described here is the only software implementation of TRW-S which exploits modern multi-core hardware.

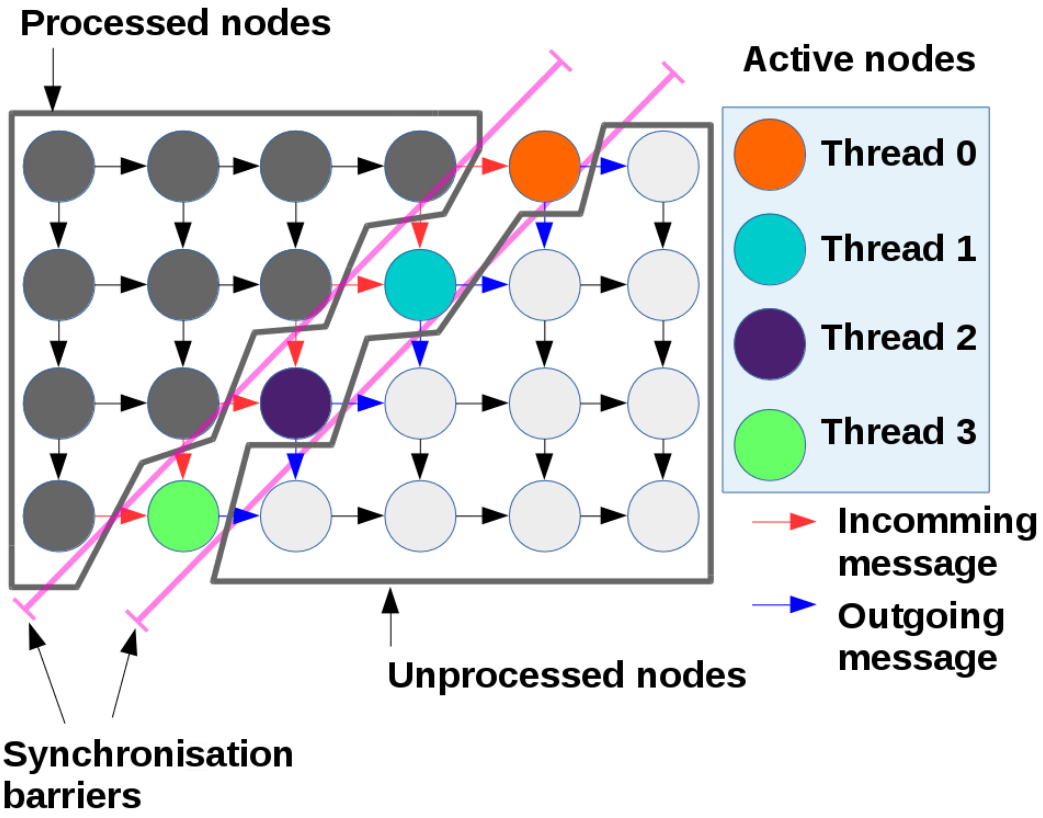


Figure 7.2: Parallel forward message passing using multiple threads (each node is a pixel).

Fig.7.2 shows an example of parallel message passing during the forward pass of TRW-S. Notice that the nodes along the diagonal are assigned to a different thread, and furthermore this respects the sequential order of TRW-S. However, this type of parallel update requires the use of synchronisation barriers meaning that all active nodes wait until the previous nodes have sent their messages (red arrows), and also wait until

current active nodes finish passing the new message (blue arrows). This same process is repeated in reverse order to cover all nodes, and therefore completing one iteration of TRW-S message passing (alg.2).

Algorithm 2 Parallel message passing for TRW-S

procedure $M^{t+1} =$ `MESSAGE_PASSING`(G, M^t, T) ▷ compute message passing
  $M^{t+1} \leftarrow M^t$
  for each $T_i \in T$ **do** ▷ Parallel region
    for each $G_{ij} \in \phi_{fw}(G)$ **do** ▷ Forward message passing
      $k \leftarrow \psi_{fw}(G_{ij}) \bmod |T|$
      if $k == Tid(T_i)$ **then**
        $i' \leftarrow i+1$
        $j' \leftarrow j+1$
        $M^{t+1}_{i'j} \leftarrow$ `compute_message`(G_{ij}, i', j)
        $M^{t+1}_{ij'} \leftarrow$ `compute_message`(G_{ij}, i, j')
      end if
      `synchronisation_barrier`(T)
    end for
    for each $G_{ij} \in \phi_{bw}(G)$ **do** ▷ Backward message passing
      $k \leftarrow \psi_{bw}(G_{ij}) \bmod |T|$
      if $k == Tid(T_i)$ **then**
        $i' \leftarrow i-1$
        $j' \leftarrow j-1$
        $M^{t+1}_{i'j} \leftarrow$ `compute_message`(G_{ij}, i', j)
        $M^{t+1}_{ij'} \leftarrow$ `compute_message`(G_{ij}, i, j')
      end if
      `synchronisation_barrier`(T)
    end for
  end for
  return M^{t+1} ▷ The computed messages
end procedure

In alg.2 M^t is the set containing the computed messages at time t, whereas M^{t+1} is the new round of messages. T is the set of threads, T_i is a single thread, $Tid(T_i) \in [0, |T|]$. G is the set of 4-connected nodes, $\phi_{fw}(G)/\phi_{bw}(G)$ is the forward/backward diagonally ordered version G and $\psi_{fw}(G_{ij})/\psi_{bw}(G_{ij})$ returns the index position of

the forward/backward diagonally ordered node G_{ij}. The forward/backward diagonal order follows the standard raster scan convention. It is important to note that using both $\psi_{fw}(G_{ij})/\psi_{bw}(G_{ij})$ results in a stridden memory access pattern, which prevents the processor cache from detecting a sequential access that could result in degraded performance when doing backward message passing.

The algorithm described above is generic and is directly implementable in CPU/GPU. However, to take advantage of the available characteristic of modern hardware the following considerations must be taken into account:

- For CPU: use vectorisation directives to expose fine grain parallelism when computing a message.
- For GPU using $CUDA$: threads T_i are replaced by "blocks", and each message entry is computed using a $CUDA$ "thread".

The main difference between CPU and GPU implementations is the fact that current $CUDA$ does not provide a global synchronisation barrier for separated blocks. This means that thread blocks must allocated dynamically for each diagonal to ensure that all blocks end at the same time. Creating threads dynamically may be an overkill, but creating threads results in a small overhead in the GPU. By contrast the implementation on a CPU creates threads statically and only once, since it's very expensive to do it dynamically.

7.3 Consistent trinocular keypoint matching

The main idea of the TBR algorithm is to use consistent keypoints in three images to recover the baseline. The following algorithm describes how consistent keypoints are obtained:

In alg.3 each $xk_l = \{xk_{l0}, ..., xk_{l\alpha}\}$, $xk_r = \{xk_{r0}, ..., xk_{r\beta}\}$, and $xk_u = \{xk_{u0}, ..., xk_{u\gamma}\}$ is a set of ASIFT keypoints descriptors with its 2D location, and $\alpha \neq \beta \neq \gamma$ means that the number of keypoints is not necessarily the same due to image lighting and occlusions. The function `compute_asift_keypoints_descriptor`(I_l, I_r, I_u) computes the keypoints for each image. The function `match_asift_keypoints_descriptor` (xk_l, xk_u) returns a set $xm_{lu} = \{xk_{lu0}, ..., xk_{lun}\}$ where each $xk_{lu0} = \{xk_{li}, xk_{uj}\}$ and $i \neq j$ under realistic conditions (in similar way xm_{ru} is computed). The next step is to obtain the consistent matching, which `rematch_asift_keypoints_descriptor`(xm_{lu}, xm_{ru})

Algorithm 3 Consistent trinocular keypoint matching

procedure $[x_l, x_r, x_u] =$ `TRINO_KEYPOINTS`(I_l, I_r, I_u) ▷ Compute keypoints
  $[xk_l, xk_r xk_u] \leftarrow$ `compute_asift_keypoints_descriptor`(I_l, I_r, I_u)
  $xm_{lu} \leftarrow$ `match_asift_keypoints_descriptor`(xk_l, xk_u)
  $xm_{ru} \leftarrow$ `match_asift_keypoints_descriptor`(xk_r, xk_u)
  $[x_l, x_r, x_u] \leftarrow$ `rematch_asift_keypoints_descriptor`(xm_{lu}, xm_{ru})
  return $[x_l, x_r, x_u]$ ▷ The computed matching keypoints
end procedure

does by matching the xk_{uj} from xm_{lu} with xk'_{uj} from xm_{ru}. This is done because matching points in the I_u image should have similar descriptors, same coordinates and similar slanted orientation. Another option is matching the left and right keypoints, but this results in less matches due to the slanted bias of ASIFT points.

7.4 Typical CPU/GPU work distribution

The implementation of the algorithm described in chapter 5 and chapter 6 uses both *CPU* and *GPU* in different stages of its computation. Fig.7.3 shows an example of how the workload is distributed between *CPU*, and *GPU*.

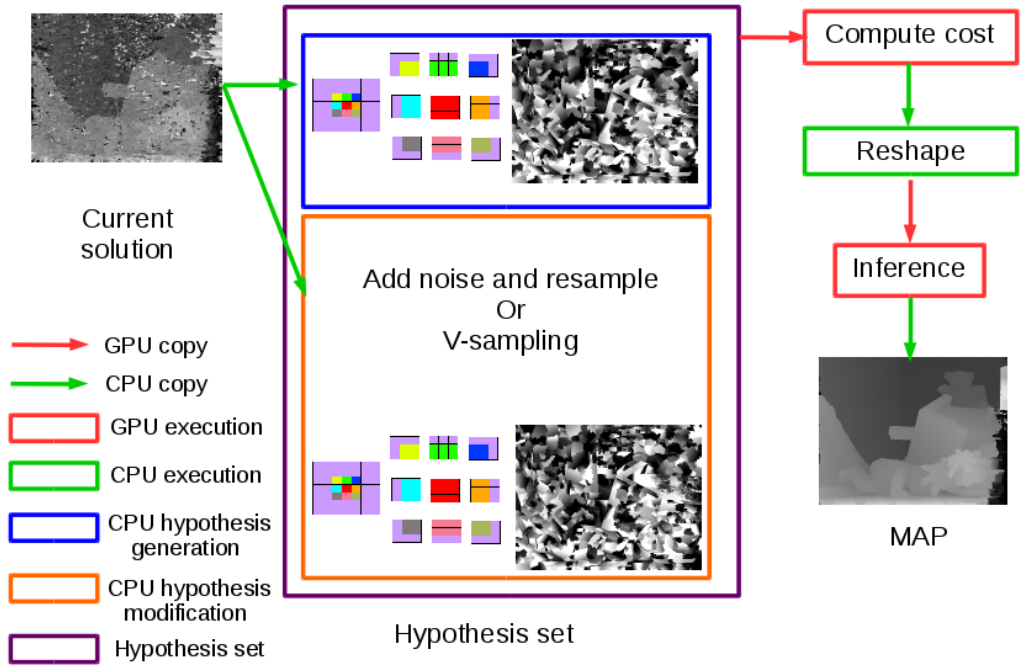


Figure 7.3: Example of CPU/GPU work distribution.

The process described in fig.7.3 is generic, which means it applies to either the propagation stage (r-sampling, P-sampling, and V-sampling) or to the refinement stage. The green arrows represent CPU memory copies either $CPU \rightarrow CPU$ or $GPU \rightarrow CPU$ depending on where the data resides. The red arrow represent GPU memory copies either $GPU \rightarrow GPU$ or $CPU \rightarrow GPU$ depending on where the data resides. The orange box can be either the propagation or refinement stage. The blue box always generates hypotheses using r-sampling, P-sampling, and V-sampling. The red boxes are the parts that are executed in the GPU. The red box "Compute cost" is alg.1, which produces the independent cost slices $C = C^1, C^2, ..., C^n$ such that $C_p = C_p^1, C_p^2, ..., C_p^n$ are not contiguous in memory. Whereas, the green box does a process called "reshape", which consist of transforming the independent cost slices C produced by alg.1 into $C'_p = C'^1_p, C'^2_p, ..., C'^n_p$ such that they are contiguous in memory, which is done to make the access to each cost entry faster during inference. In a similar way the smoothness term is pre-computed and reshaped. The reshaped cost and smoothness term is copied $CPU \rightarrow GPU$, inference is carried out , and when finalised sent back to the CPU.

The time required for the execution depends on the image size, and number of hypotheses used; for instance, using a total of 64 hypotheses the compute times for an image of $1470x970$ pixels the compute time would be: 12.0 seconds for CPU hypotheses generation (blue box) including $V - sampling$ (orange box) in the propagation stage, 12.7 seconds for CPU when adding noise and resampling (orange box) in the refinement stage, 25.4 seconds for computing the cost (GPU) including reshape operation (CPU), and 16.8 seconds for inference and copy back to CPU. This times assume the following hardware configuration: 1) $1x$ Intel Core $i5-4460$ at $3.2Ghz$ (4 cores CPU), 2) $16GB$ of $DDR3$ RAM at $1600Mhz$, and 3) $2x$ NVIDIA GTX 1080 (GPU). Thus the compute time for one propagation iteration would be $T_{prop} = 2 \times (12.0 + 25.4 + 16.8)$ seconds, as left and right disparity maps are computed. The time for a refinement iteration would be $T_{ref} = 2 \times (12.7 + 25.4 + 16.8)$ seconds as left and right disparity maps are computed. Therefore the compute time for the whole optimisation process is $T_{opti} = 2629.60 = 4 \times (T_{prop} + 5T_{ref})$ seconds. Additionally, the compute time to estimate segmentation and local cues (see chapter 4) is $T_{sg} = 242.15$ seconds. The compute time of initialisation using SGM for around 150 disparities is $T_{sgm} = 73$ seconds. The compute time required for the quarter and octave sizes is $T_{sc} = 418.0$ seconds (includes segmentation and local cues). Combining scales has cost of $T_{cb} = 89.6$ seconds. This brings the final time to $T_{total} = 3452.35 = T_{opti} + T_{sg} + T_{sgm} + T_{sc} + T_{cb}$ seconds.

It is worth noting that the memory requirement for the cost is $O(n)$ with n be-

ing the number of hypotheses. The memory requirement for the smoothness term is either $O(5n)$ (if all neighbouring pixel values are computed) or $O(4n)$ (if only the plane plus disparity are used). As mentioned in chapter 5 we use 64 hypothesis at a time, which is done due to the space need and *GPU* vectorisation requirements, as hypotheses must be multiples of SIMD width (32 in the *GPU* case). For an image of $1470x970$ pixels, and using 64 hypothesis the memory required is divided as follows: $Mem_{cost} = 64 \times 5.4394Mb$ (data cost), $Mem_{planes} = 4 \times 64 \times 5.4394Mb$, $Mem_{smooth} = 5 \times 64 \times 5.4394Mb$ (smoothness term), $Mem_{messages} = 2 \times 64 \times 5.4394Mb$ (messages used by TRW-S), $Mem_{gle} = 4 \times 5.4394Mb$ (edge growth limits), and $Mem_{weight} = 4 \times 5.4394Mb$ (edge weights). The memory cost for SMG is at a minimum $Mem_{sgm} = 150 \times 2 \times 5.4394Mb$ (assuming 150 disparities). This brings the total minimum memory required for a single image to $Mem_{total} = 5852.8Mb = Mem_{cost} + Mem_{planes} + Mem_{smooth} + Mem_{messages} + Mem_{gle} + Mem_{weight} + Mem_{sgm}$.

Chapter 8

Conclusions

In this thesis we have developed algorithms that address the stereo matching problem as a 3D plane labelling. The experiments we have conducted so far have shown that our approach has been able to successfully exploit the underlying 3D surface of the scene to adapt the window size, exploit local features of the texture to combine two different aggregation strategies, and explore a continuous 3D space. Additionally we have developed a novel baseline recovery algorithm that allows us to extend our binocular stereo matching algorithm to the trinocular case. This has resulted in good results in both indoor and outdoor environments when tested with the Middlebury and KITTI data sets.

In this chapter we discuss the accomplishments of our research, limitations of our proposed approach, and finally possible extension to our work either to improve its performance or propose new areas of research derived from the current thesis.

8.1 Thesis accomplishments

The 3D plane labelling stereo matching algorithms developed in this thesis were extensively validated with real data. It is worth noting that our stereo matching algorithms not only produce sub-pixel accurate disparity maps, but also produce a 3D disparity plane assignment per pixel, and recover camera motion. In order to obtain the performance reported in previous chapters, our research has introduced four novel techniques:

1. ***Hypothesis generation***: r-sampling, P-sampling, and V-sampling allow the generation of hypotheses, propagation of hypotheses, and exploration of a continu-

ous 3D space. Our approach is an alternative to the locally shared labels from [88] and has proved to match its performance.

2. ***Adaptive window estimation***: The proposed technique is able to exploit both image content, and underlying 3D structure to better adapt the window size and thus allowing us to recover small details that otherwise would not be recovered (see sec.5.5).

3. ***Local uniqueness term***: This term to the best of knowledge has not been used before, and furthermore it has proved to be fast to compute and reduces errors in textureless areas (see sec.5.12).

4. ***Baseline recovery***: This algorithm is a new technique to recover a third camera. Our results have shown that it delivers the same accuracy as the 6 point algorithm [30], but at lower computational cost.

One important characteristic of the algorithms presented is that they can be initialised with either random or precomputed disparity maps. This can potentially allow our algorithm to be initialised using low resolution or sparse disparity maps, which is an inexpensive way of enhancing the original data. Finally, the results described in previous chapters show that our algorithm might be improved by integrating modern machine learning techniques to compute the stereo matching cost with our 3D plane labelling approach.

8.2 Critical analysis discussion

The works discussed in chapter 5 and chapter 6 introduced the ideas of r-sampling, P-sampling, V-sampling, local uniqueness term, and baseline recovery. Although, the results presented showed good results in both indoors and outdoors scenes, it is important to point out their shortcomings. The areas that deserve particular attention can be listed in the following way:

- Window size: Using two windows sizes proved to help the algorithm to reduce the effect of the centre pixel bias, but there may be small/fine details in the scene that will need a smaller/larger window size, or will be lost otherwise.

- MRF connectivity: Our approach uses the classic 4-connected MRF, which has a tendency to generate “blocky” artifacts in the disparity map. This is particularly noticeable close to edges.

- MRF local interactions: Our approach does not use higher order cliques, which means that the local smoothness term estimated is likely to get stuck at a local minimum.

- Per pixel cost evaluation: Our approach uses a computationally expensive cost function that needs to be evaluated per pixel. This provides good results, but in very large textureless areas it causes the algorithm to take longer to propagate a plane.

- Local uniqueness cost: The proposed uniqueness cost obtained from the cost at the end of the optimisation process is not the "real" cost, but instead it has been passed down from the selected hypothesis at each step of the optimisation process, which is likely to cause the algorithm to get stuck in a local minimum.

- Combining trinocular cost: The proposed trinocular cost depends on the use of a transformation to compute the potential match. Although, not immediately obvious this may introduce unwanted noise in the cost due to the scale change for pixels that do not lie at the same disparity.

- Multiple motions: The baseline recovery algorithm only recovers the dominant motion, and therefore in dynamic scenes this means that moving objects are likely to have a different motion which results in introducing unwanted noise in the trinocular matching cost and degraded reliability of the estimated disparity.

- Noisy cost: The proposed cost function achieves good results. However, the cost is very noisy and ambiguous under challenging lighting conditions (e.g. one image is much brighter than the other). This issue could be addressed by introducing a convolutional neural network based function, but it will require to modify existing approaches (eg.[97]) to work on slanted surfaces.

8.3 Possible LPU/TBR algorithm extensions

The binocular and trinocular stereo matching algorithms described in previous chapters achieves good performance on challenging data sets. However there are areas of improvement which were not covered due to time constraints. The possible areas of future exploration are:

- Use an 8-connected MRF for increased accuracy at depth discontinuities.

- Accelerate computation of the matching cost by using a combination of segment and pixel matching costs.
- Replace the local uniqueness term with a global one.
- Use machine learning techniques to compute the adaptive windows shape/size.
- Use an MRF that adapts its grid dynamically to the image content for increased accuracy on textureless areas.

Another important subject of future research is the new applications that can be derived from the work presented in this thesis, which includes:

Baseline recovery and SLAM: The proposed algorithm to recover motion has the potential of being used in a *SLAM* application. The experiments of chapter 6 show that the baseline recovery algorithm has an average error of 0.5 pixels of error, which indicates that the accuracy of the estimated transformations could be used for mapping purposes. However, for such application the baseline recovery algorithm must be extended to use four views of a stereo rig moving, which would help further constrain the optimization algorithm. Additionally, the keypoint matching must be extended as well to take into account dynamic scenes, such that the baseline recovery algorithm can recover the multiple motions present in a scene.

LPU for optical flow: The algorithm presented in chapter 5 could be extended to compute optical flow. However, this would require an upgrade from the plane representation to homography to take into account scale change and large displacements in optical flow. To accomplish this there are three alternatives: 1) Estimate the camera motion from a fundamental matrix (multiple transformations may be present if it is a dynamic scene), and upgrade to Euclidean reconstruction (as initialization) using known camera calibration parameters. 2) Compute projective camera from the estimated fundamental matrix, and use it to compute initial homography estimates. 3) Parametrize the homography such that scale and shearing can be controlled. This approach would assume that no reliable camera estimate could be obtained for initialization. The homography approach could also be used as a way to compute residual vertical disparity to compensate for rectification errors.

TBR for joint disparity and optical flow estimation: The trinocular approach described in chapter 6 can only handle static scenes. In the case of dynamic scenes it is necessary to compute the optical flow to improve the disparity estimation of objects that change position over time. This requires extending the baseline recovery algorithm to extract multiple camera motions from the scene, and also requires to upgrade from 3D planes to moving 3D planes using a homography to take into account the change of scale. This approach would recover both disparity and optical flow simultaneously.

Generalise the baseline recovery algorithm for non-rectified cameras: The algorithm presented in chapter 6 works with rectified cameras. However, this limits its applications and therefore it is necessary to extend the baseline recovery algorithm to the general case of non-rectified cameras, for instance a stereo system that can change the camera's vergence and baseline, or a single camera moving to three different locations. The main reason to extend the baseline recovery algorithm is the result presented in chapter 6 in which our approach outperforms the classical six point algorithm. This could lead to a new camera motion recovery algorithm that is faster and more accurate.

Generalise the baseline recovery algorithm for multi-view stereo: Extending the baseline recovery algorithm (in chapter 6) for multi-view stereo could potentially be used to improved motion estimation as there would be more views to constrain the problem and make it better defined. Multiple baselines could be used to constrain optimization problem, and furthermore the baseline recovery algorithm could be used to refine extrinsic camera calibration.

Convolutional neural networks for slanted cost computation: The *LPU* (chapter 5) and *TBR* (chapter 6) algorithms could be further improved by using convolutional neural networks to compute a less noisy similarity cost. To accomplish this the following issues have to addressed: 1) Find efficient an algorithm to compute the pixel cost in a fast when each pixel has a plane, as recomputing the cost for each plane is computationally expensive. 2) Change from a plane per pixel approach to a plane per segment approach to make it feasible to use convolutional neural networks to compute the similarity cost.

Appendix A

Parameter exploration for LPU

A.1 Effect of the cost function parameters

Parameter variation	%bad noc	avg. error	rms
$k = 3$	13.02	1.87	7.52
$k = 9$	13.14	1.86	7.33
$r = 1$	13.21	1.82	7.35
$P = 1$	13.21	1.81	**7.13**
$P = 3$	13.10	1.82	7.44
$\lambda = 0.12$	13.23	1.84	7.35
$\lambda = 0.24$	13.22	1.83	7.21
$t_{diff} = 0.075$	13.09	1.85	7.33
$t_{diff} = 0.025$	12.96	1.86	7.42
$\tau_{cen} = 14$	13.50	1.88	7.44
$\tau_{cen} = 19$	13.62	1.92	7.57
$\omega_1 = 35, \omega_2 = 19$	13.17	1.81	7.21
$\omega_1 = 29, \omega_2 = 13$	13.43	1.83	7.33
$\alpha = 15$	13.42	1.90	7.52
$\alpha = 45$	**12.91**	**1.78**	7.24
$\tau_{grad} = 6$	13.16	1.92	7.66
$\tau_{grad} = 9$	13.93	2.11	8.40
$\tau_w = 1.5$	13.18	1.86	7.38
$\tau_w = 4.0$	13.05	1.85	7.25
LPU	12.95	1.82	7.28

Table A.1: Average error metrics (in non-occluded areas of the Middlebury dataset V3) of the proposed pixel similarity cost function with different parameter.

In tab.A.1 %bad noc is the ranking table criterion used in the Middlebury benchmark. The experiments in tab.A.1 show how the parameter choice affects the error metrics on the Middlebury data set. In tab.A.1 *LPU* is our algorithm using the following parameter settings $\omega_1 = 41 \times 41$, $\omega_2 = 25 \times 25$, $\tau_{grad} = 3/255$, $\tau_{cen} = 9/25$, $\sigma_r = 10/255$, $\sigma_d = 0.5$, $\tau_w = 2.5$, $\tau_{diff} = 0.07$, $\tau_{unique} = 0.01$, $\alpha = 30$, $K_1 = 1$, $K_2 = 6$, $r = 2$, $P = 6$.

Notice that in tab.A.1 $\alpha = 45$ provides the best results, but the results in chapter 5 use $\alpha = 30$. The larger $\alpha = 45$ gives more influence to the pixel-wise cost, as the similarity cost has less noise (in good quality images as the ones from Middlebury), which results in less fattening effect. However, our algorithm was also tested with outdoor images, where the pixel-wise cost is noisy. Thus our parameter selection was based on having good performance in both the indoor (e.g. the Middlebury data set) and outdoor environments (e.g. the KITTI data set). Only $r = 1$ and $r = 2$ were tested as larger values require more memory.

A.2 Example of window estimation (Jadeplant image)

Figure A.1: Jadeplant image.

Here we discuss the effects of the window estimation process with different parameters. Fig.A.1 shows the "Jadeplant" image from the Middlebury data set, fig.A.2 the entropy filter output (blue low value, red high value), fig.A.3 the initial disparity estimate, and fig.A.4 is the mask that indicates where the window size should be changed

with parameters $\tau_h = 2$, $\tau'_w = 0.5$ (i.e. at least half of segment must be a depth discontinuity), and $K_w = 8$ as in chapter 5. The boxes show three different regions 1) textured and slanted, 2) occluded and low contrast region, 3) textured fine details, 4) textured curved, and 5) occluded textureless with fine details.

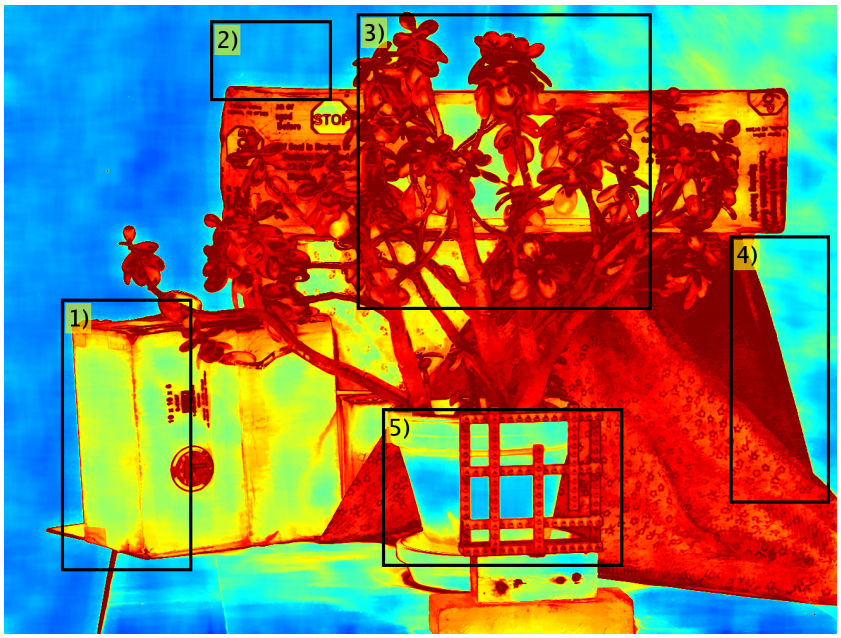


Figure A.2: Adaptively filtered entropy filter.

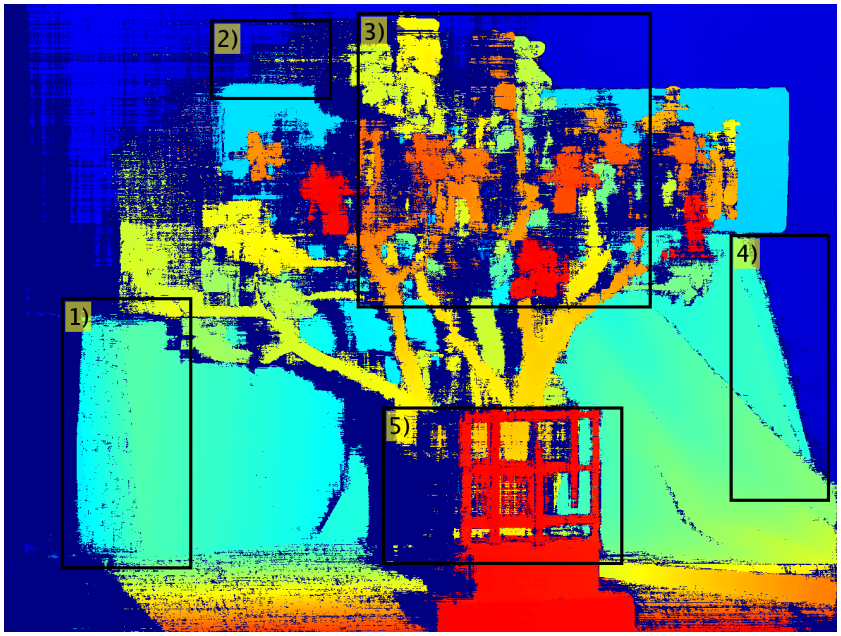


Figure A.3: Jadeplant initial disparity.

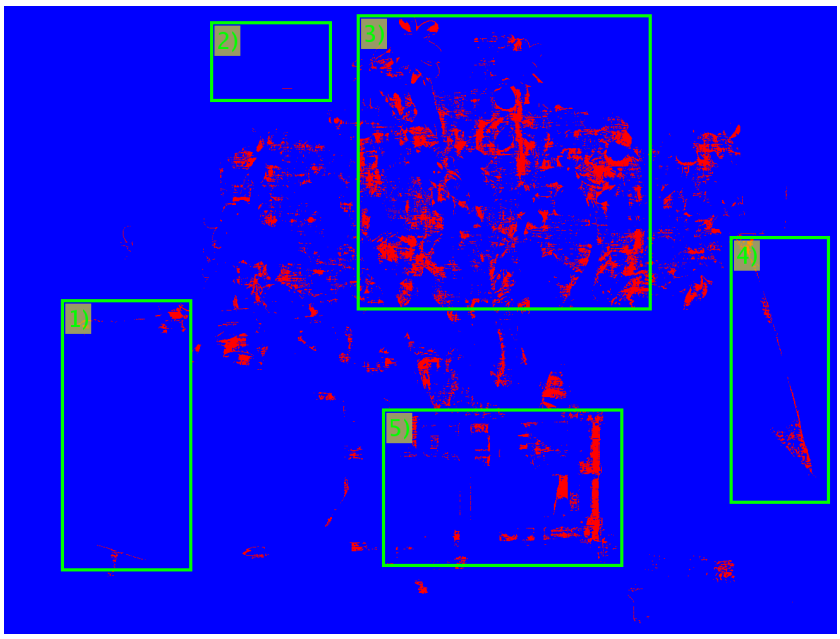


Figure A.4: Estimated window mask using parameters from chapter5.

Fig.A.4 shows that region 3) is where most of the adaptive window size change takes place. In the rest of the areas the window size is not changed as much, which is explained by little variation of disparity (e.g. flat areas).

A.2.1 Effect of τ_h on the window estimation

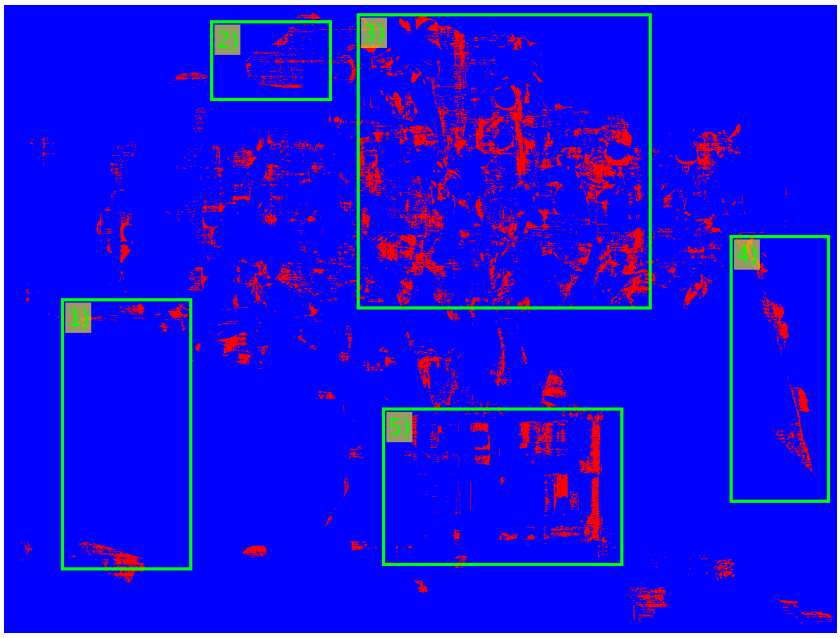


Figure A.5: Estimated window mask using $\tau_h = 1$.

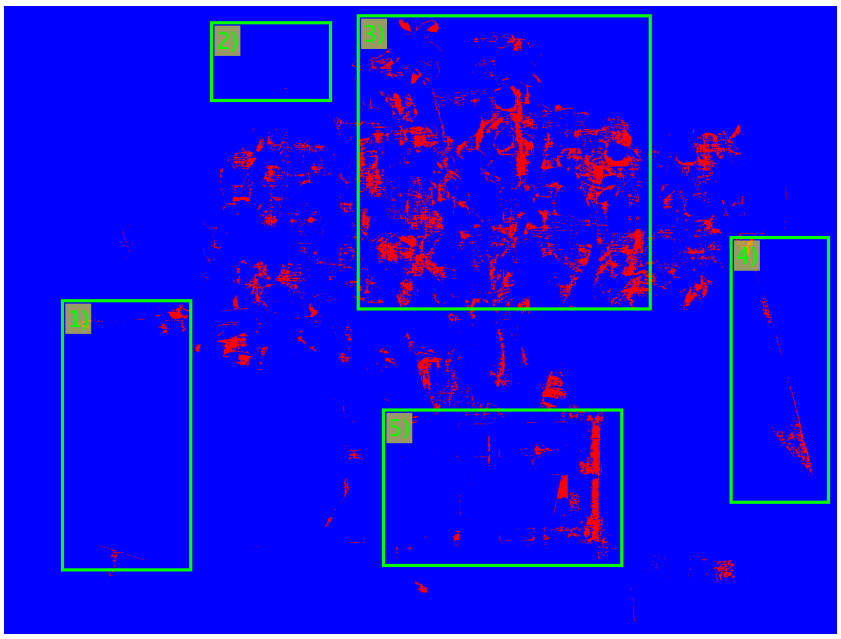


Figure A.6: Estimated window mask using $\tau_h = 2.5$.

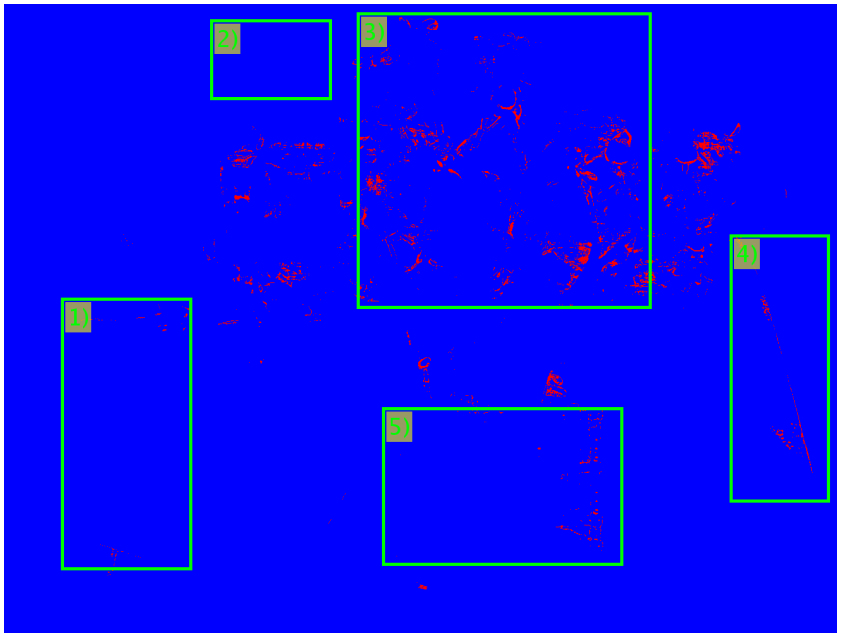


Figure A.7: Estimated window mask using $\tau_h = 4.0$.

1) In the textured and slanted region fig.A.5 ($\tau_h = 1$), fig.A.6 ($\tau_h = 2.5$) , and fig.A.7 ($\tau_h = 4.0$) there is little change as expected due to low variation in disparity, texture changes.

2) In the occluded and low contrast region only fig.A.5 ($\tau_h = 1$) estimates different window size, because of the low entropy threshold and variation in disparity (due to noise).

3) In the textured fine details region only fig.A.7 ($\tau_h = 4.0$) shows very few pixels need a different window size, which is caused due the high entropy threshold used.
4) In the textured curved region only fig.A.5 ($\tau_h = 1$) estimates different window size, because of the low entropy threshold.
5) In the occluded textureless area with fine details region only fig.A.7 ($\tau_h = 4.0$) shows very few pixels are estimated to need a different window size, which is caused due the high entropy threshold used. In this case the adaptive windows are disabled.

A.2.2 Effect of K_w on the window estimation

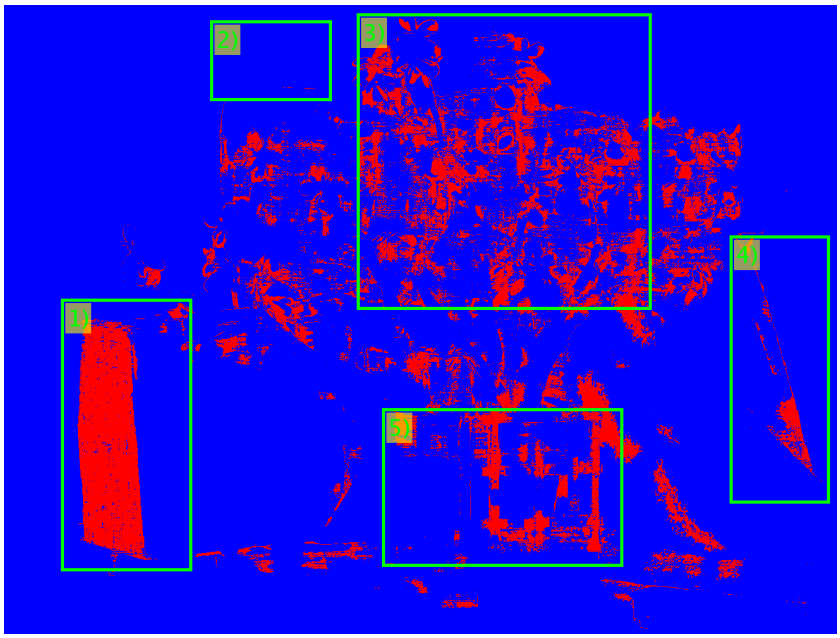


Figure A.8: Estimated window mask using $K_w = 4.0$.

1) In the textured and slanted region fig.A.4 ($K_w = 8.0$) shows no change as the threshold is large enough. Whereas fig.A.8 ($K_w = 4.0$), and fig.A.7 ($K_w = 6.0$) detect the whole region as needing different window size. However, this in not desirable as the region is textured and slanted. This issue is caused due to the median filter used to smooth each segment, and later gradient analysis using a low threshold will cause all pixel to be marked as needing a window size change.
2) In the occluded and low contrast region all parameters fig.A.8 ($K_w = 4.0$), and fig.A.9 ($K_w = 6.0$) have no change of window size. This is expected due to the use of median filter in the segment, which eliminates the noisy disparity value.
3) In the textured fine details region only fig.A.8 ($K_w = 4.0$) shows a large number of

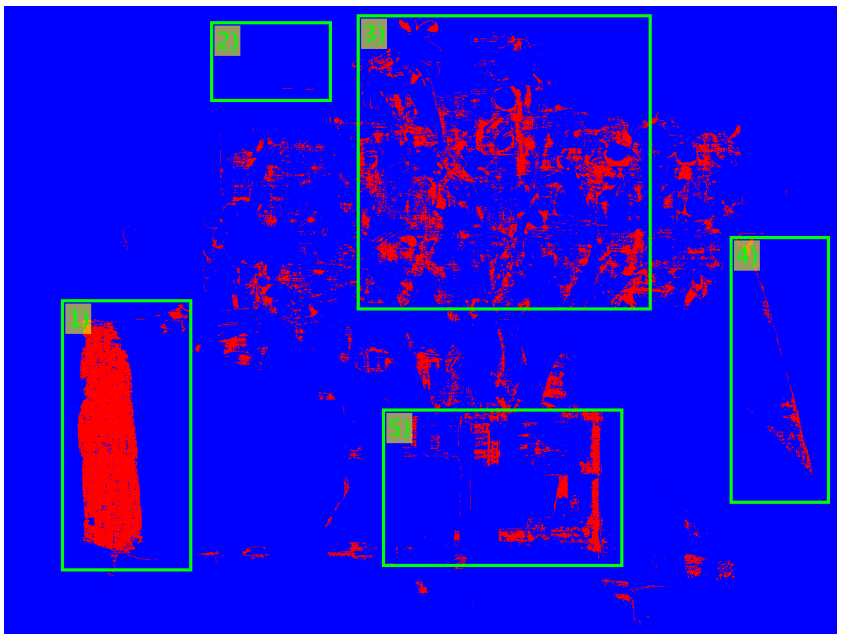


Figure A.9: Estimated window mask using $K_w = 6.0$.

pixels needing a different window size, which is caused due the low disparity threshold used in region with large disparity changes.
4) In the textured curved region only fig.A.8 ($K_w = 4.0$) shows a large number of pixels needing a different window size, which is caused due the median filter applied to the segments, and low disparity threshold used in region with large disparity changes.
5) In the occluded textureless with fine details region only fig.A.8 ($K_w = 4.0$) shows a large number of pixels needing a different window size, which is caused due the low disparity threshold used in region with large disparity changes.

A.2.3 Effect of τ'_w on the window estimation

1) In the textured and slanted region fig.A.16 ($\tau'_w = 0.5$), fig.A.21 ($\tau'_w = 0.75$) show similar results as the threshold is large enough. Whereas fig.A.21 ($\tau'_w = 0.25$) detects a large number of pixels as needing different window size. However, this in not desirable as the region is textured and slanted. This issue is caused due to the median filter used to smooth each segment, and later gradient analysis using a low threshold will cause all pixel to be marked as needing a window size change.
2), 3), 4) and 5) All regions in fig.A.16 ($\tau'_w = 0.5$), fig.A.21 ($\tau'_w = 0.25$) , and fig.A.22 ($\tau'_w = 0.75$) show increased number of pixels needing window size change is reduced as the value of the threshold used is increased. This is expected as the parameter controls the total number of pixels that need to be a depth discontinuity along the segment

perimeter.

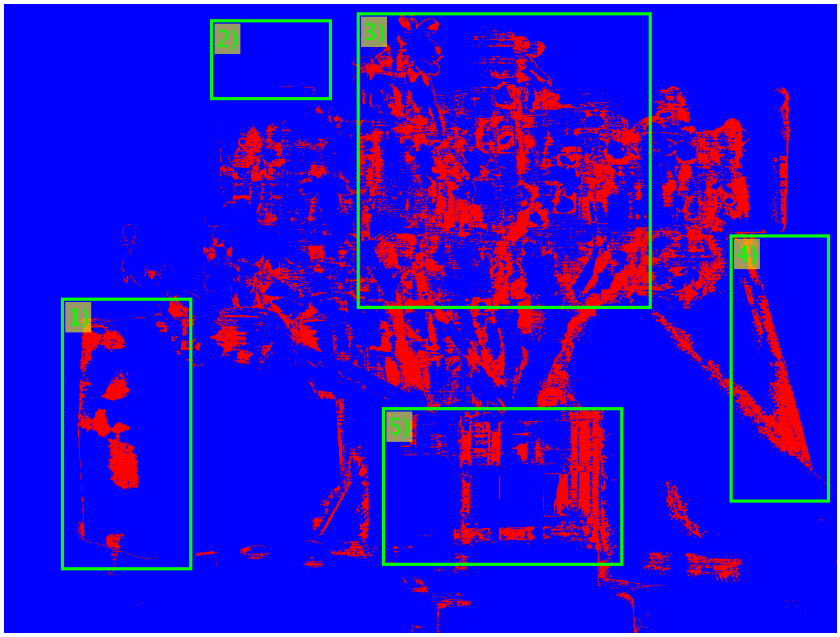


Figure A.10: Estimated window mask using $\tau'_w = 0.25$.

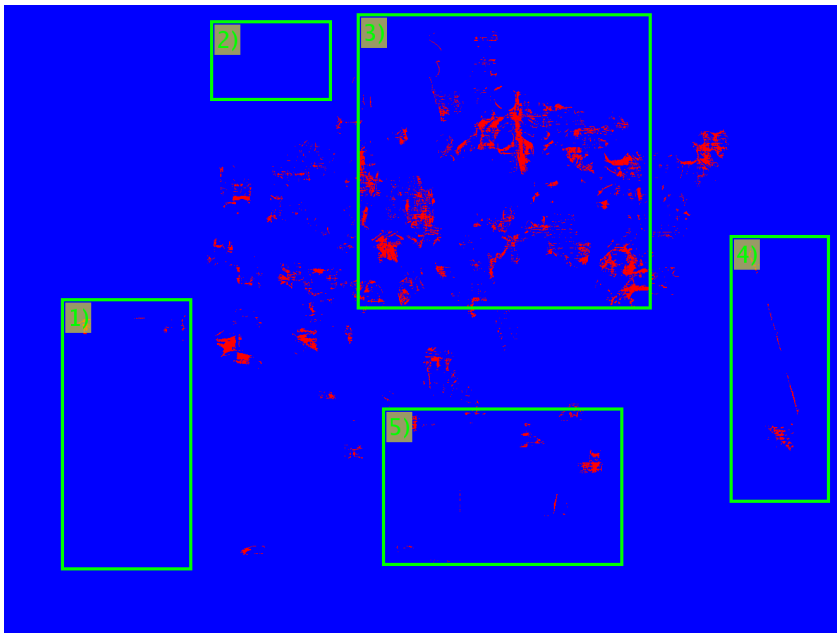


Figure A.11: Estimated window mask using $\tau'_w = 0.75$.

A.3 Example of window estimation (Pipes image)

Here we discuss the effects of the window estimation process with different parameters. Fig.A.12 shows the "Pipes" image from the Middlebury data set, fig.A.14 the entropy filter output (blue low value, red high value), fig.A.13 the initial disparity estimate, and fig.A.15 is the mask that indicates where the window size should be changed with parameters $\tau_h = 2$, $\tau'_w = 0.5$ (i.e. at least half of segment must be a depth discontinuity), and $K_w = 8$ as in chapter 5. The boxes show there different regions 1) low contrast with depth discontinuities, 2) multiple depths, 3) low texture curved, 4) textured with depth discontinuity, and 5) multiple depths with low entropy.

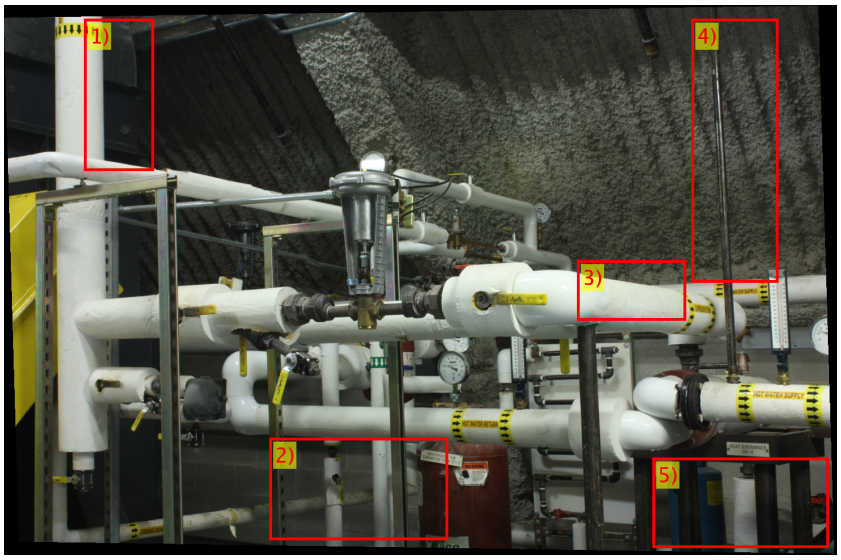


Figure A.12: Pipes image.

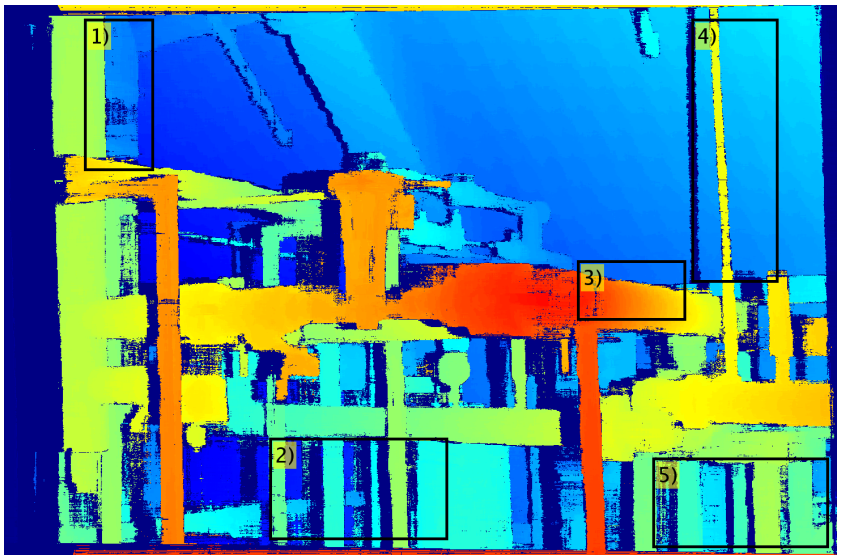


Figure A.13: Pipes initial disparity.

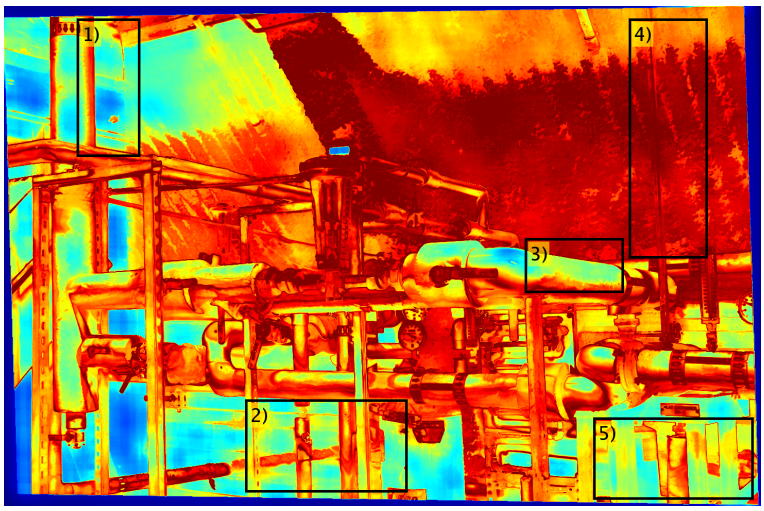


Figure A.14: Adaptively filtered entropy filter.

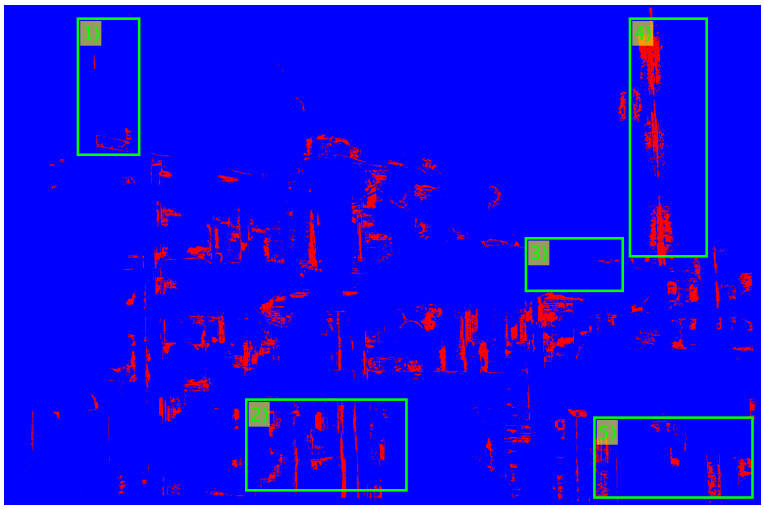


Figure A.15: Estimated window mask using parameters from chapter5.

Fig.A.15 that how region 1) is least of the adaptive window size change takes place. In the rest of the areas there window size is changed due to depth discontinuities and entropy value (as expected in this type of situation).

A.3.1 Effect of τ_h on the window estimation

1) In the low contrast with depth discontinuities fig.A.16 ($\tau_h = 1$), fig.A.17 ($\tau_h = 2.5$) , and fig.A.18 ($\tau_h = 4.0$) there is little change as expected due to low variation in disparity, and texture changes.

2) In the multiple depths region only fig.A.16 ($\tau_h = 1$) estimates different window size, because of the low entropy of the background and the difference in depth from

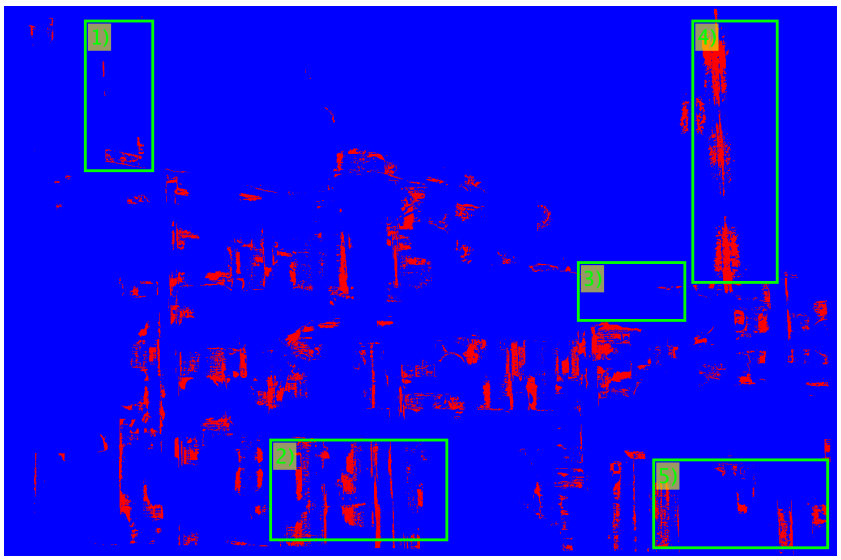


Figure A.16: Estimated window mask using $\tau_h = 1$.

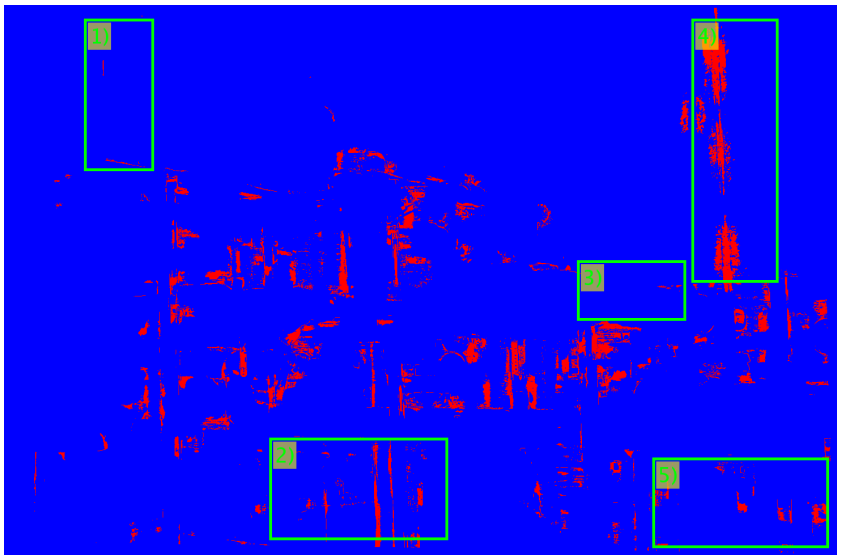


Figure A.17: Estimated window mask using $\tau_h = 2.5$.

the different pipes (mostly close to edges).

3) In the low texture curved region fig.A.16 ($\tau_h = 1$), fig.A.17 ($\tau_h = 2.5$) , and fig.A.18 ($\tau_h = 4.0$) do not show any pixel as needing change in window size. This is due to the smooth change in disparities (curved surface) and low entropy value, but it can result in a strong planar bias.

4) In the textured with depth discontinuity only fig.A.18 ($\tau_h = 4.0$) is unable to detect parts of the pipe due to the large threshold used.

5) In the multiple depths with low entropy with fine details region only fig.A.18 ($\tau_h = 4.0$) is unable to detect any pixel that may need window size change. Whereas, fig.A.16,

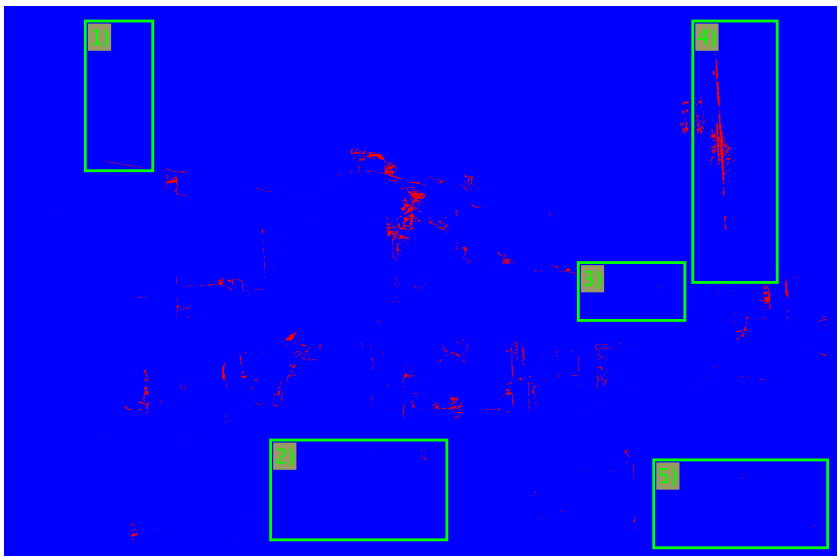


Figure A.18: Estimated window mask using $\tau_h = 4.0$.

fig.A.17 ($\tau_h = 2.5$) and fig.A.18 ($\tau_h = 4.0$) show reduced number of pixel as the threshold value is increased.

A.3.2 Effect of K_w on the window estimation

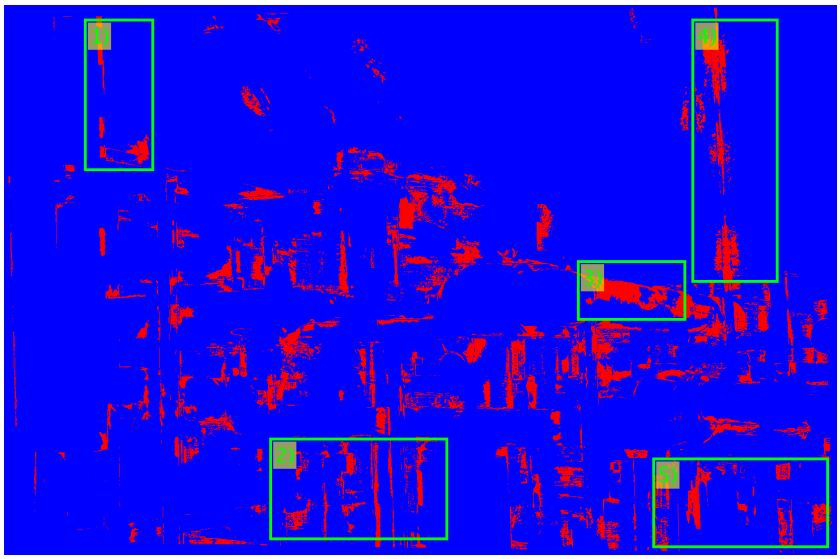


Figure A.19: Estimated window mask using $K_w = 4.0$.

1) In the low contrast with depth discontinuities fig.A.16 ($K_w = 8.0$), fig.A.20 ($K_w = 6.0$) , and fig.A.19 ($K_w = 4.0$) show reduced the number of pixel as needing window size change is reduced as the value of the threshold used is increased. This is expected as the depth discontinuities are small.

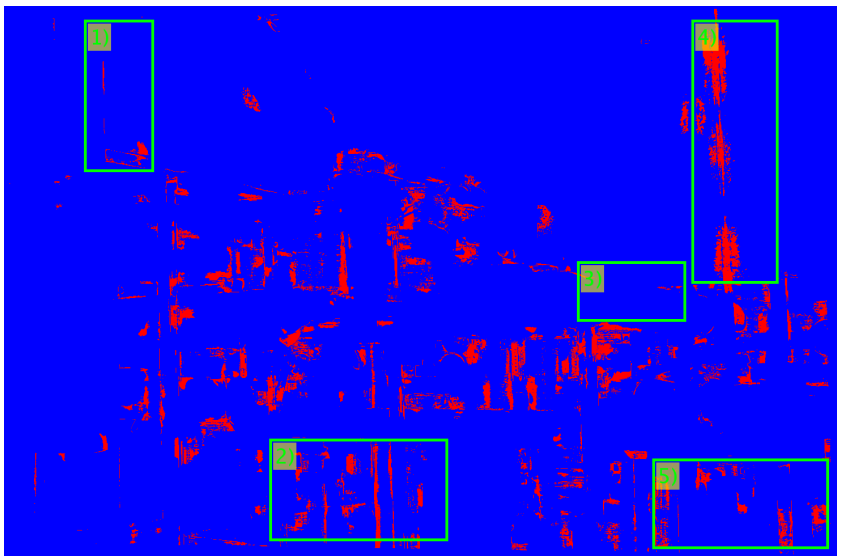


Figure A.20: Estimated window mask using $K_w = 6.0$.

2) In the multiple depths region fig.A.16 ($K_w = 8.0$), fig.A.20 ($K_w = 6.0$) , and fig.A.18 ($K_w = 4.0$) behave similarly as the disparity discontinuities are large, thus making the threshold value less relevant.

3) In the low texture curved region fig.A.16 ($K_w = 8.0$) and fig.A.20 ($K_w = 6.0$) show no pixel needing window size change. Whereas fig.A.19 ($K_w = 4.0$) shows a large number of pixels needing a different window size, which is caused due the median filter applied to the segments, and low disparity threshold used in region with large disparity changes.

4) In the textured with depth discontinuity fig.A.16 ($K_w = 8.0$), fig.A.20 ($K_w = 6.0$), and fig.A.19 ($K_w = 4.0$) is unable to detect parts of the pipe due to the large threshold used.

5) In the multiple depths with low entropy with fine details region fig.A.16 ($K_w = 8.0$), fig.A.19 ($K_w = 6.0$) , and fig.A.19 ($K_w = 4.0$) show reduced the number of pixel as needing window size change is reduced as the value of the threshold used is increased. This is expected as the depth discontinuities are small.

A.3.3 Effect of τ'_w on the window estimation

In regions 1) 2), 3), 4) and 5) all regions in fig.A.16 ($\tau'_w = 0.5$), fig.A.21 ($\tau'_w = 0.25$) , and fig.A.22 ($\tau'_w = 0.75$) show increased number of pixels needing window size change is reduced as the value of the threshold used is increased. This is expected as the parameter controls the total number of pixels that need to be a depth discontinuity

along the segment perimeter.

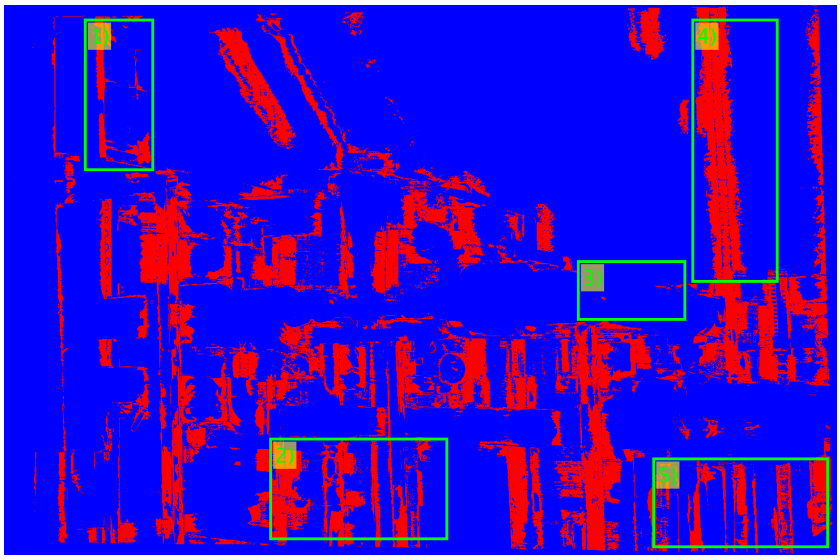


Figure A.21: Estimated window mask using $\tau'_w = 0.25$.

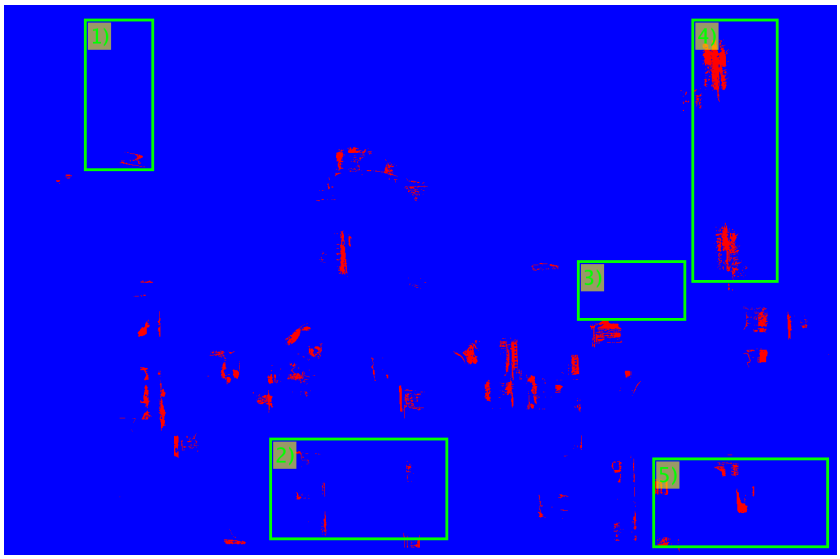


Figure A.22: Estimated window mask using $\tau'_w = 0.75$.

Appendix B

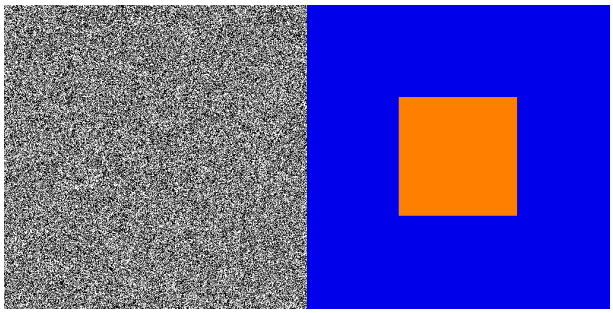
LPU response to random dot stereogram and delta functions

In this appendix we examine the behaviour of the LPU algorithm developed in chapter 5 when presented with the classical test of a random dot stereogram, responses to delta functions, and variations in texture. The main reason is to evaluate these behaviours is to find out how our algorithm behaves with images that have a perfect correspondence and how our propagation algorithm behaves in the presence of limited features in the images. All images used in these experiments have a resolution of 512×512 pixels.

B.1 Random dot stereogram

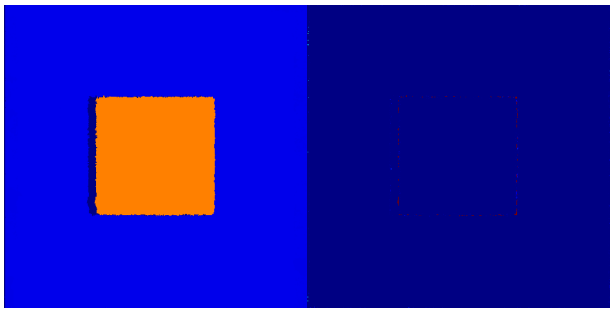
In this experiment we test two scenarios: 1) One block is displaced in the right image (fig.B.1), and 2) Two blocks are separated by different distances. The purpose of this is to evaluate how well edges are preserved, and how well can our algorithm detect occluded regions.

Fig.B.1 shows the groundtruth disparity map (fig.B.1b) and the output of the LPU algorithm (fig.B.2a). Notice that our algorithm estimates the correct disparity in non occluded areas. However, the corners of the square are rounded, which is explained by the lack of well defined features in the stereogram causing the aggregation algorithm to behave more like block matching. Another issue is the serrated square edge (see error map fig.B.2b). In this case, the truncation for the adaptive smoothness term uses the different values (as it is computed from the image gradient), and also uses a different weighting constant causing the serrated edges (see fig.B.3 small green dots along the border).



(a) (b)

Figure B.1: Random dot stereogram: reference left image ((a)) and groundtruth disparity map ((b)).



(a) (b)

Figure B.2: Random dot stereogram: LPU result ((a)) and error map ((b)).

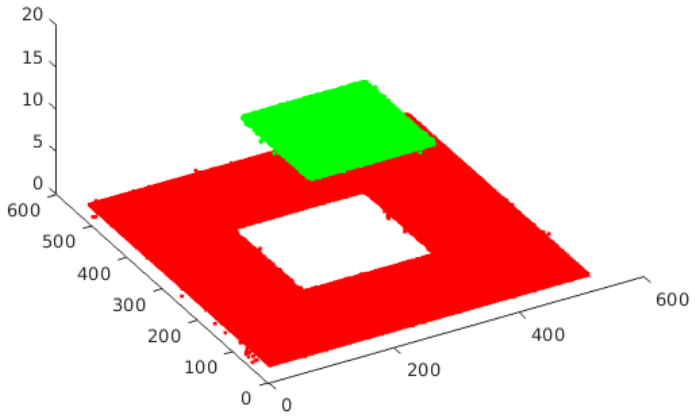
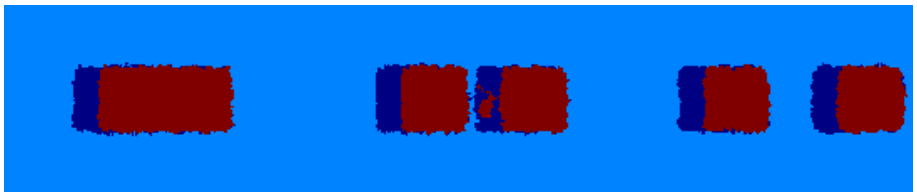


Figure B.3: Random dot stereogram: 3D result visualisation.



(a) no separation (b) separation of 20 pixels (c) separation of 40 pixels

Figure B.4: Blocks of 40×40 pixels (zoomed in for visualisation) at the same disparity separated by different distances: 0 pixels ((a)); 20 pixels ((b)); 40 pixels ((c)).

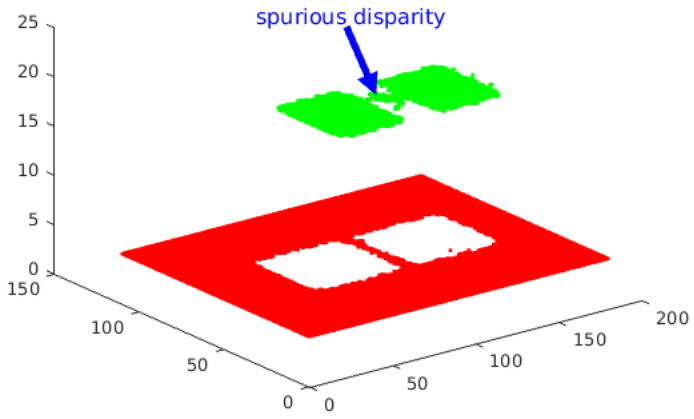


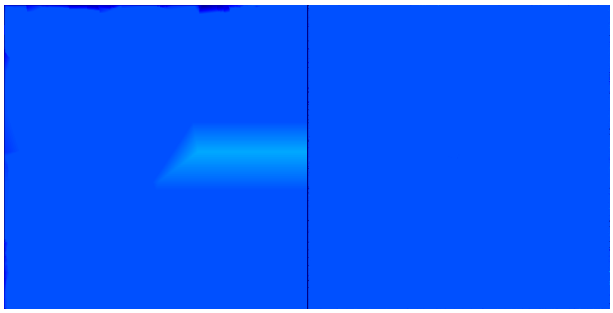
Figure B.5: Floating boxes error: 3D result visualisation for 20 pixels of separation.

Fig.B.4 shows the output of the LPU algorithm when two blocks of 40×40 pixels at the same disparity are separated by different distances. The proposed algorithm produces similar output as in fig.B.2 and is able to correctly match the two blocks as they separate. However, notice that in fig.B.4b the disparity map has an error in the occluded area (see fig.B.5). This is caused is caused by the local uniqueness constraint that has got stuck at a local minimum, which could be addressed using the current solution as an extra hypothesis and recomputing the local uniqueness penalty rather than using a previously estimated penalty.

B.2 Impulse response

In this experiment we test two scenarios 1) One point (delta function) at the centre of the left image is displaced in the right image by 40 disparities, and 2) Two delta func-

tions separated (0 pixels, 20 pixels, and 40 pixels) horizontally leftwards and vertically upwards in the left image centre, and the displaced in the right image with either constant disparity (40 disparities) or varying disparities (10 disparities for the centre, and 20 disparities for the separated delta function). The purpose of this is to evaluate the behaviour of the disparity propagation scheduler of the proposed algorithm.



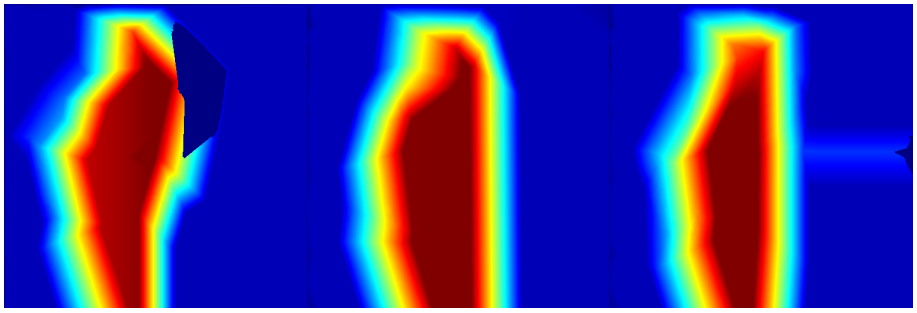
(a) no background (b) with background

Figure B.6: Resulting disparity map of a single dot at disparity 40: single dot with no background((a)); single dot with random dot background((b)).

Fig.B.6 shows the resulting disparity maps of the LPU algorithm when a single displaced dot with constant background for two cases: textureless case fig.B.6a, and random dot textured case fig.B.6b. Notice that our algorithm estimates a large spread pyramid like shape to the right of the delta function for the textureless case, and disparity almost constant (i.e. the dot has been lost). The ideal result should be single point in the disparity map, but in the textureless case (single white dot with black background) all pixels have a low matching cost, the smoothness term favours solutions with similar disparities, and the single pixel propagates its disparity to neighbouring areas. The algorithm is working as it is supposed to work in textureless areas. In the textured case from fig.B.6b the disparity is constant and the dot is not present in the final result because of the use of multi-scale and smoothing.

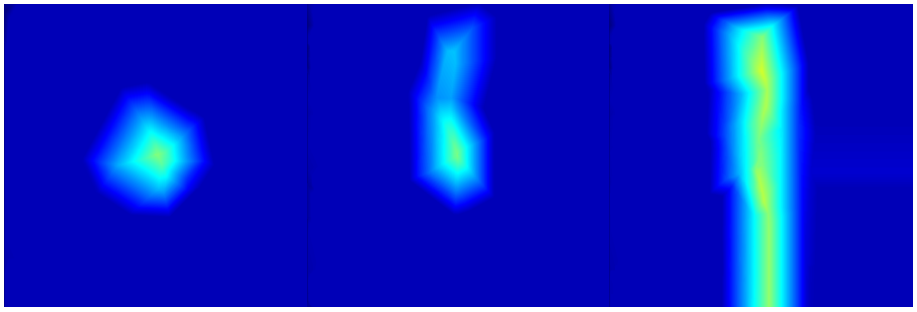
Fig.B.7 shows the result of two dots at the same disparity, but horizontally separated by different distances. The proposed algorithm produces a large pyramid-like structure (larger than in B.6a) and our algorithm propagates in the vertical direction (up and down) in the space between the two dots. This behaviour is expected and indicates that our approach might work in textureless areas as long as there are a few matching points on the textureless surface that are close, otherwise the result will be a disparity map with small bumps on the textureless surface. However, when the dots are at different disparities the effect of the large propagation is reduced significantly

as shown in fig.B.8. This can be explained by the local uniqueness constraint that is penalizing the different disparities of the dots that can cause multiple matches.



(a) no separation (b) separation of 20 pixels (c) separation of 40 pixels

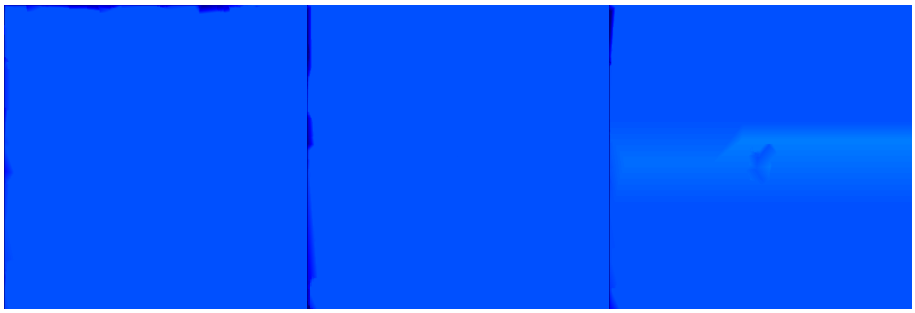
Figure B.7: Disparity maps of two delta functions at the same disparity and horizontally separated by different distances: 0 pixels ((a)); 20 pixels ((b)); 40 pixels ((c)).



(a) no separation (b) separation of 20 pixels (c) separation of 40 pixels

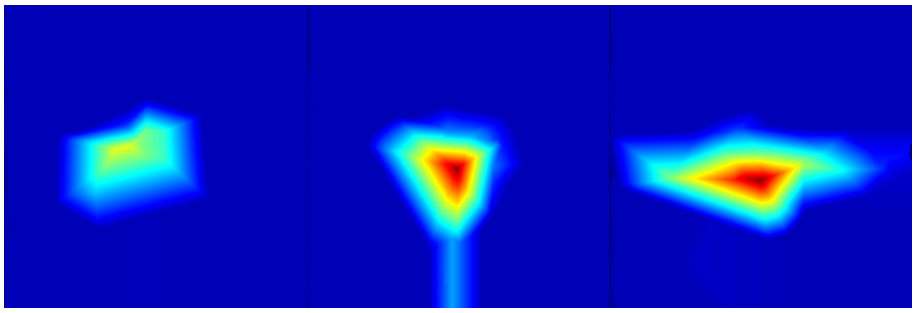
Figure B.8: Disparity maps of two delta functions at different disparities (10 disparities, and 20 disparities) and horizontally separated by different distances: 0 pixels ((a)); 20 pixels ((b)); 40 pixels ((c)).

Fig.B.9 shows the result of two dots at the same disparity, but vertically separated by different distances. The proposed algorithm is unable to detect the single dots, whereas fig.B.9c produces a similar output as in fig.B.6a and our algorithm propagates in the horizontal direction (left and right) in the space between the two dots. This behaviour is expected and indicates that our approach might work in textureless areas as long as there are a few matching points on the textureless surface that are close, otherwise the result will be a disparity map with small bumps on the textureless surface. However, when the dots are at different disparities the effect of the large propagation is significant as shown in fig.B.10, which exhibits the same behaviour of fig.B.8.



(a) no separation (b) separation of 20 pixels (c) separation of 40 pixels

Figure B.9: Disparity maps of two delta functions at the same disparity and vertically separated by different distances: 0 pixels ((a)); 20 pixels ((b)); 40 pixels ((c)).



(a) no separation (b) separation of 20 pixels (c) separation of 40 pixels

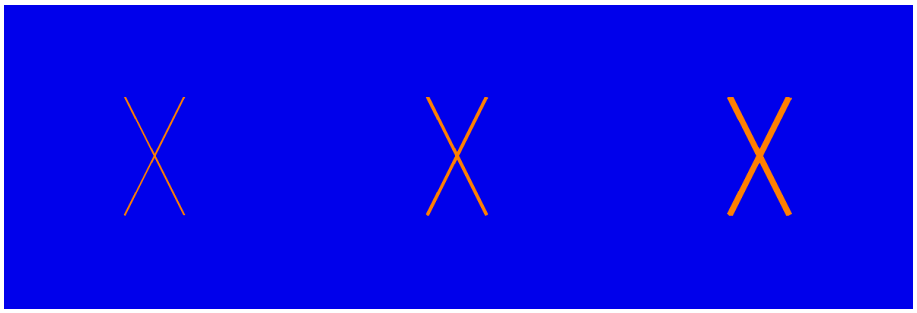
Figure B.10: Disparity maps of two delta functions at different disparities (10 disparities, and 20 disparities) and vertically separated by different distances: 0 pixels ((a)); 20 pixels ((b)); 40 pixels ((c)).

B.3 Effect of texture

(a) (b) (c)

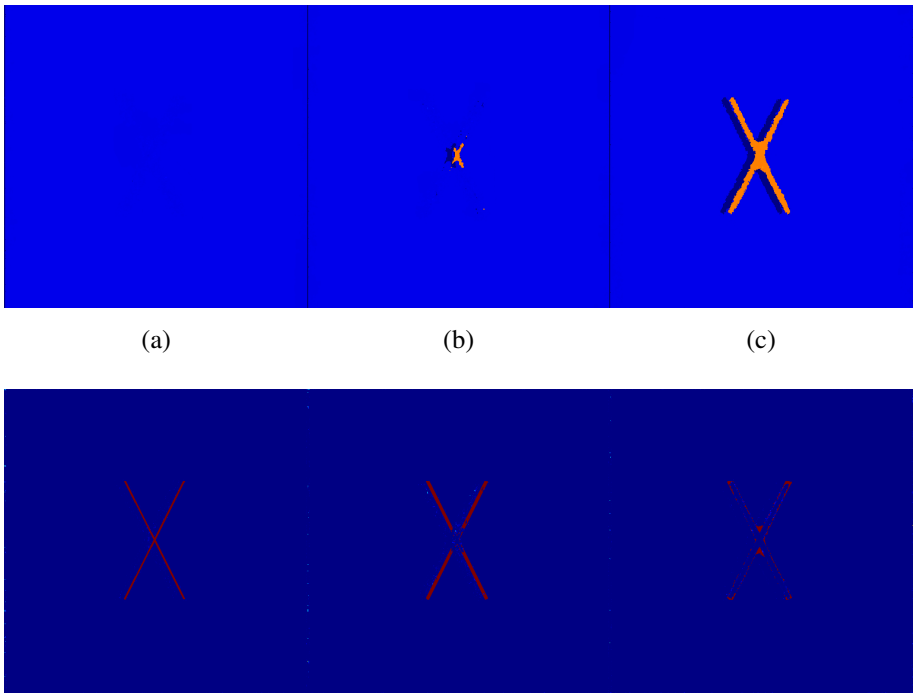
Figure B.11: X image with well defined texture: 2 pixels width ((a)); 4 pixels width ((b)); 8 pixels width ((c)).

In this experiment we evaluate both the effect of texture and object size in the image when estimating a disparity map. We use an X-shaped figure with variations on the width of the lines used (2 pixels, 4 pixels, and 8 pixels). Two cases are analysed: 1) Image using random dot stereograms, and 2) Image using well defined textures, see fig.B.11 for texture image and fig.B.12 for groundtruth disparity maps.



(a) (b) (c)

Figure B.12: X image disparity maps: 2 pixels width ((a)); 4 pixels width ((b)); 8 pixels width ((c)).

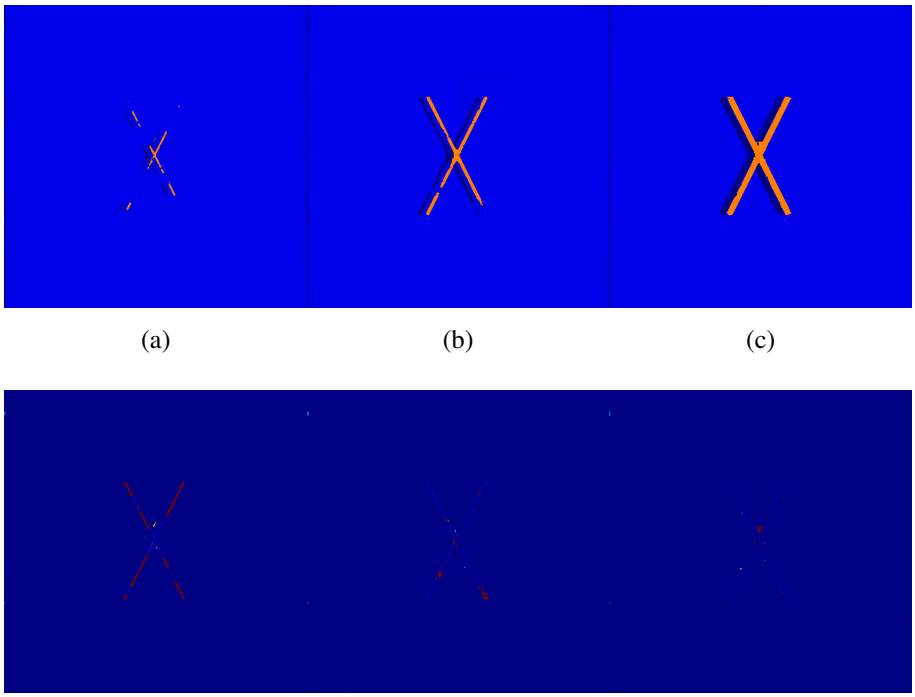


(a) (b) (c)

(d) (e) (f)

Figure B.13: Result with random dot texture: 2 pixels width result (a) with error map (d); 4 pixels width result (b)) with error map (e); 8 pixels width result (c) with error map (f).

Fig.B.13 shows the behaviour of the proposed algorithm when images lack clearly defined features as in a random dot stereogram. The proposed algorithm was unable to recover the X-shaped figure when the lines were 2 (fig.B.13a) and 4 (fig.B.13b) pixels wide. However, when the line is 8 pixels wide our algorithm recovers the X-shaped figure, but as shown in fig.B.13f our result has large errors: 1) Around the external corners disparity is shrunk, and 2) At the centre intersection the disparity is overgrown. This behaviour is expected as the algorithm was unable to use any features to estimate the growth limit for the edge model, multi-scale removed the X at lower scales, and over-smoothing caused disparity to either shrink or disappear.



(a) (b) (c)

(d) (e) (f)

Figure B.14: Result with well defined texture: 2 pixels width result ((a)) with error map (d); 4 pixels width result (b) with error map (e); 8 pixels width result (c) with error map (f).

Fig.B.14 shows the results of our algorithm on well textured images. The results are clearly better when compared to fig.B.13. However, the proposed algorithm still struggles to obtain correct results when the line is 2 and 4 pixels wide. There are several possible explanations: 1) The multi-scale approach is unable to recover the details at lower scale, and thus when propagating to a larger scale causes the uniqueness constraint to get stuck at a local minimum. 2) The window size is too large causing

small details to be lost. 3) The smoothness term is only 4-connected, which means that diagonal depth edges will be prone to either overgrowing or being overtaken by the background. In fig.B.14c our algorithm has been able to correctly estimate the X-shaped figure with sharp external edges, and small overgrowth around internal line intersection, as shown by fig.B.14f.

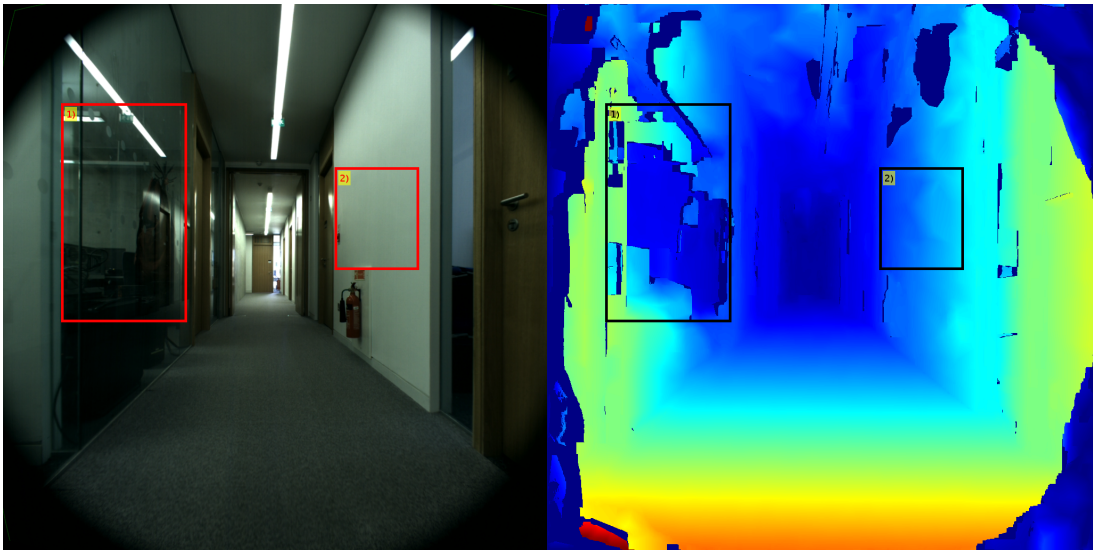
B.4 Conclusions

The proposed algorithm is able to estimate disparity even when the lack of features (using the random dot stereogram) makes it default to the classical behaviour of block matching with a smoothness term. Our algorithm is able to recover fine details as long as they are well textured, and the multi-scale stage leaves them untouched. Finally, the response of our algorithm to delta functions shows how the proposed propagation scheduler behaves on textureless areas.

Appendix C

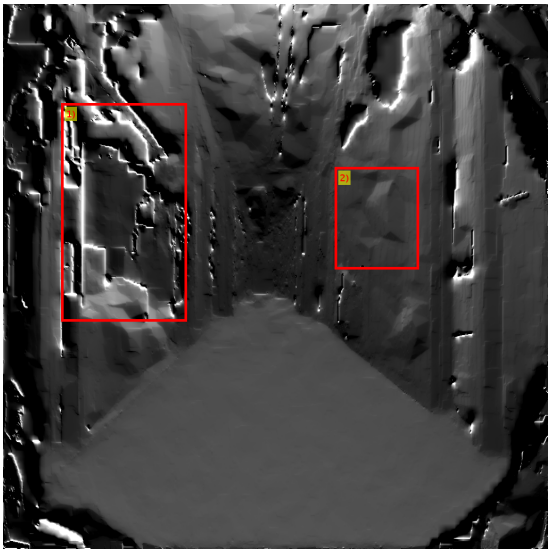
Additional images and test cases

In this appendix we present additional results (without post-processing) for three different test cases. The first case consists of a corridor with textureless walls and transparent surfaces (expected bad result). The second case consists of a corridor with some textureless and transparent regions, but with additional well textured regions (expected reasonable results). The third case consist of a scene where all surfaces have well textured regions (expected good result). The images presented here were captured by the author using a *MultiSense* camera with a short baseline of 7cm. The images are from the second floor of the Informatics Forum at the University of Edinburgh, and the outside sequence corresponds to George Square just outside the Informatics Forum.



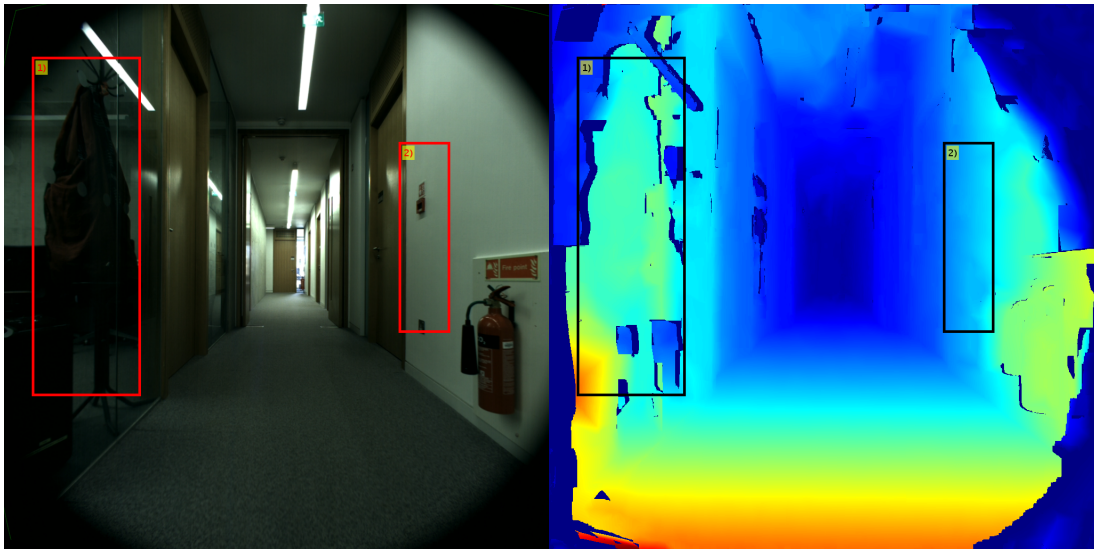
(a) (b)

Figure C.1: "Corridor 0" test image ((a)) and *Our* result ((b))



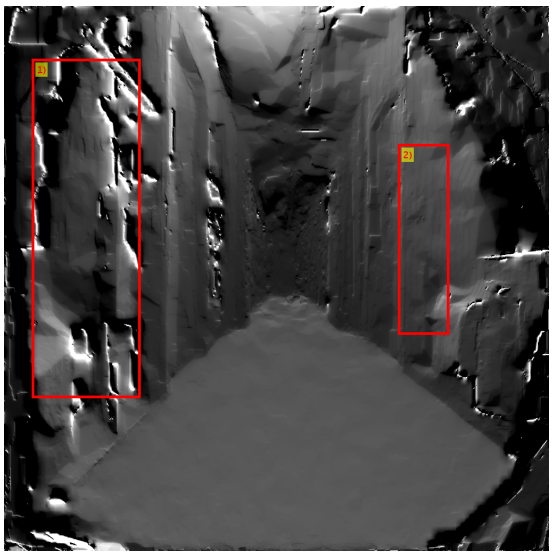
(a)

Figure C.2: “Corridor 0” test image: Cosine shaded disparity map.



(a) (b)

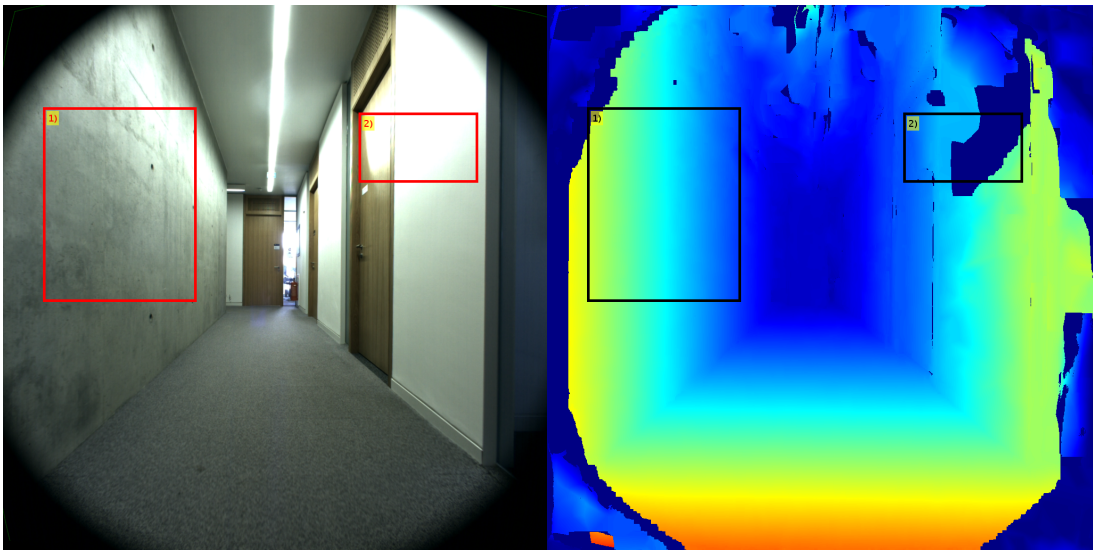
Figure C.3: “Corridor 4” test image ((a)) and Our result ((b))



(a)

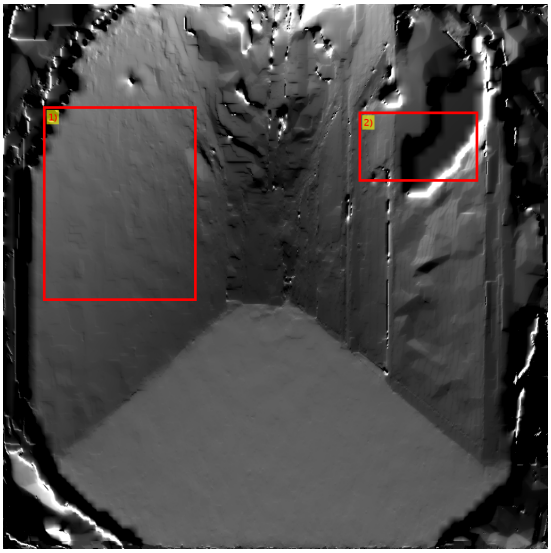
Figure C.4: “Corridor 4” test image: Cosine shaded disparity map.

Fig.C.1 and fig.C.3 show incorrect results on images with two regions: 1) Transparent area where our approach was unable to recover the correct disparity of the glass, and instead the background disparity was estimated. 2) Textureless area where our approach has estimated small bumps (fig.C.4) instead of the correct flat surface.



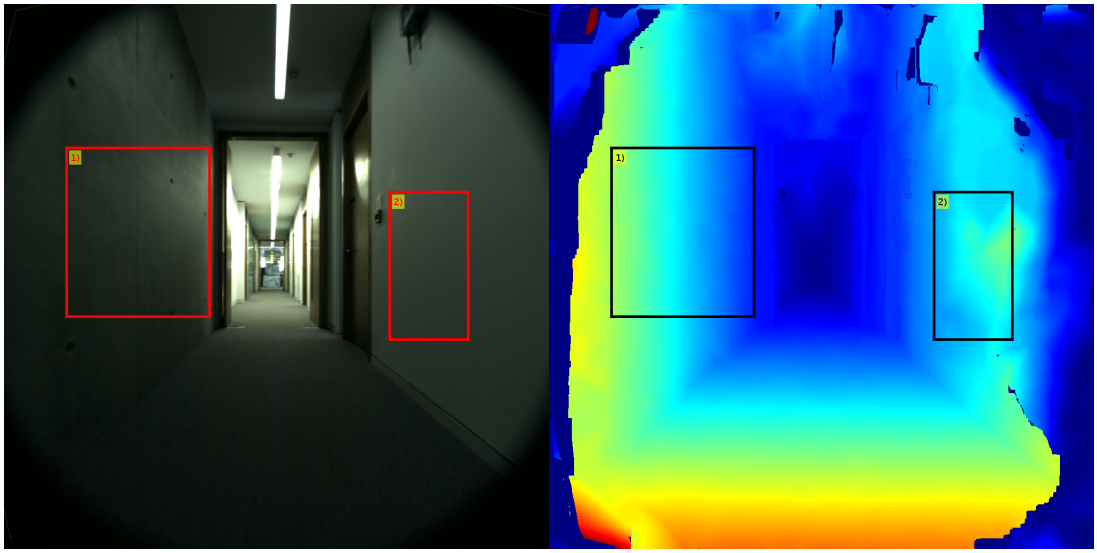
(a) (b)

Figure C.5: “Corridor 8” test image ((a)) and *Our* result ((b)).



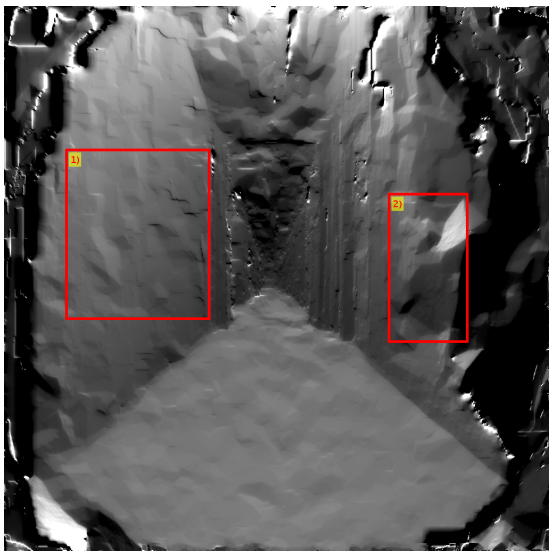
(a)

Figure C.6: “Corridor 8” test image: Cosine shaded disparity map.



(a) (b)

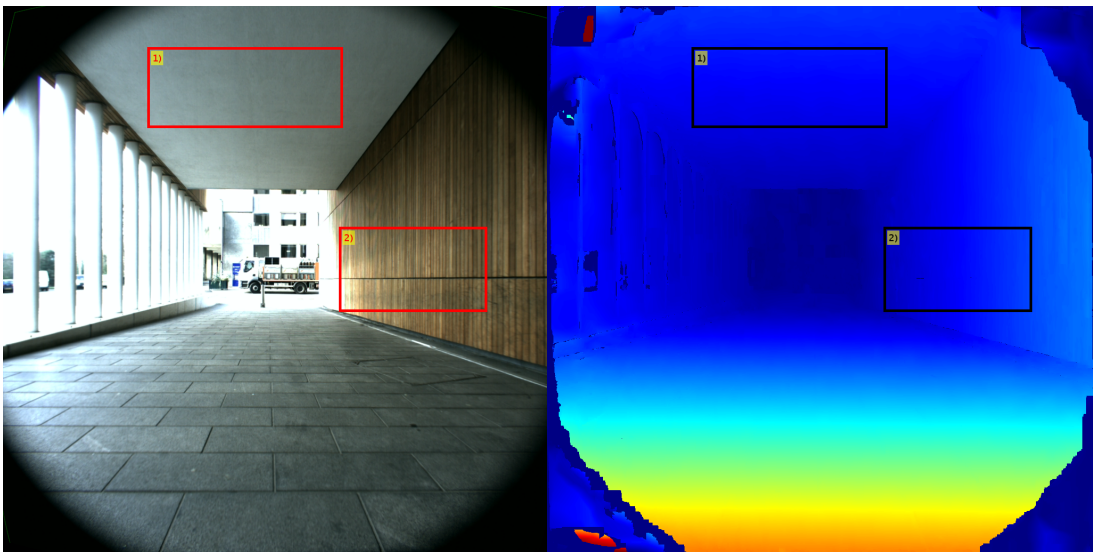
Figure C.7: “Corridor 16” test image ((a)) and *Our* result ((b)).



(a)

Figure C.8: “Corridor 16” test image: Cosine shaded disparity map.

Fig.C.5 and fig.C.7 show partially correct results on images with two regions: 1) Textured area where our approach was able to correctly recover the disparity of the wall. 2) Textureless area where our approach has been either unable to recover consistent disparity (box 2 in fig.C.5b and fig.C.7b) or estimated bumps (fig.C.6 and fig.C.8) instead of a flat surface.



(a) (b)

Figure C.9: “Outside 0” test image ((a)) and *Our* result ((b)).

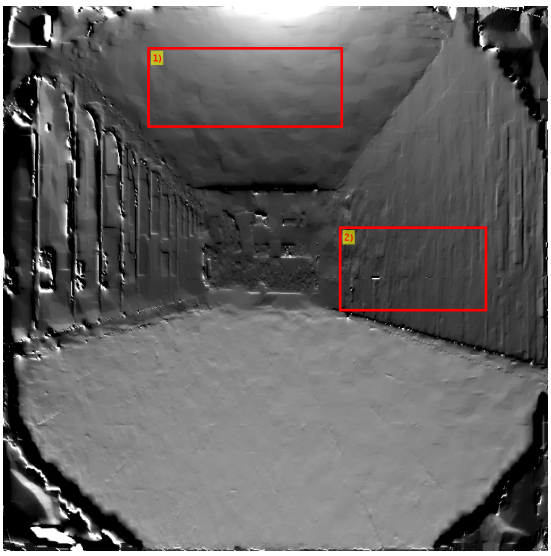
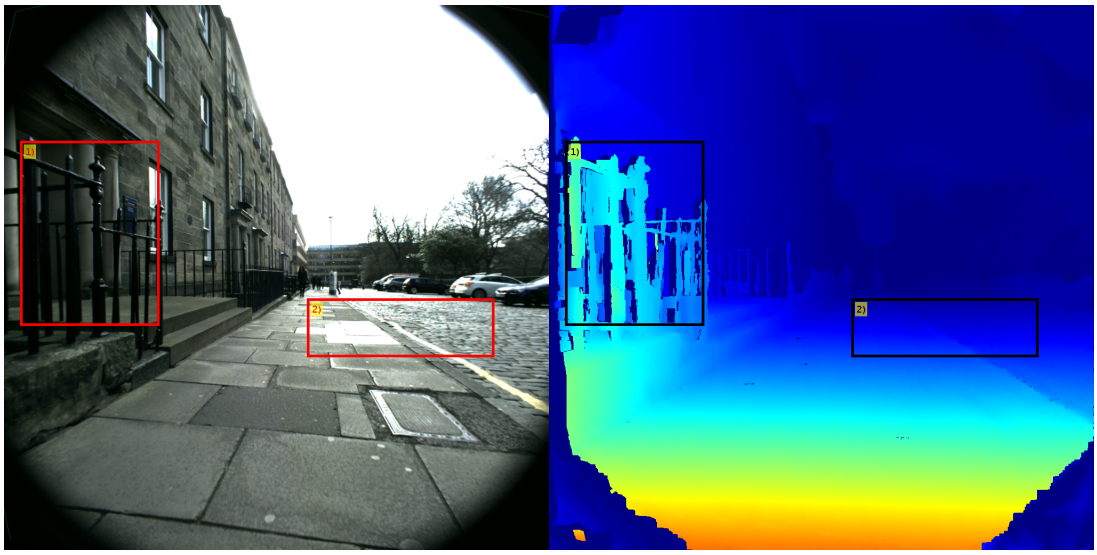


Figure C.10: “Outside 0” test image: Cosine shaded disparity map.

Fig.C.9 shows correct results on an image with two regions: 1) Textured area where our approach was able to correctly recover the disparity of the ceiling. 2) Textured repetitive region where our approach has correctly recovered the slanted wall. The good results are due to good lighting conditions in the scene, and well textured surfaces.



(a) (b)

Figure C.11: “Outside 12” test image ((a)) and *Our* result ((b)).

Fig.C.11 shows the results on a image with two regions: 1) Occluded thin regions where our approach was able to recover partially correctly the disparity of the fence,

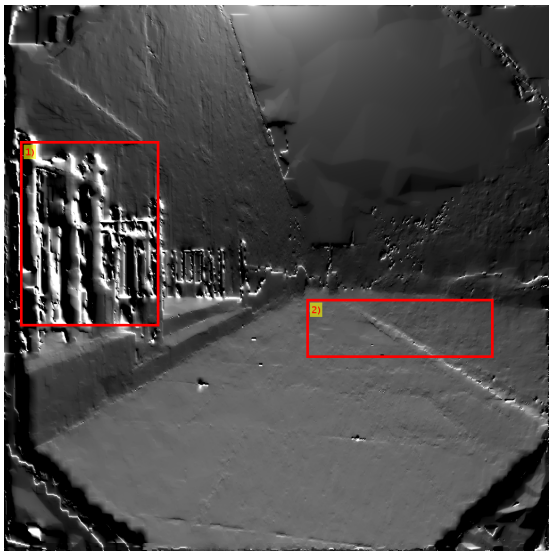
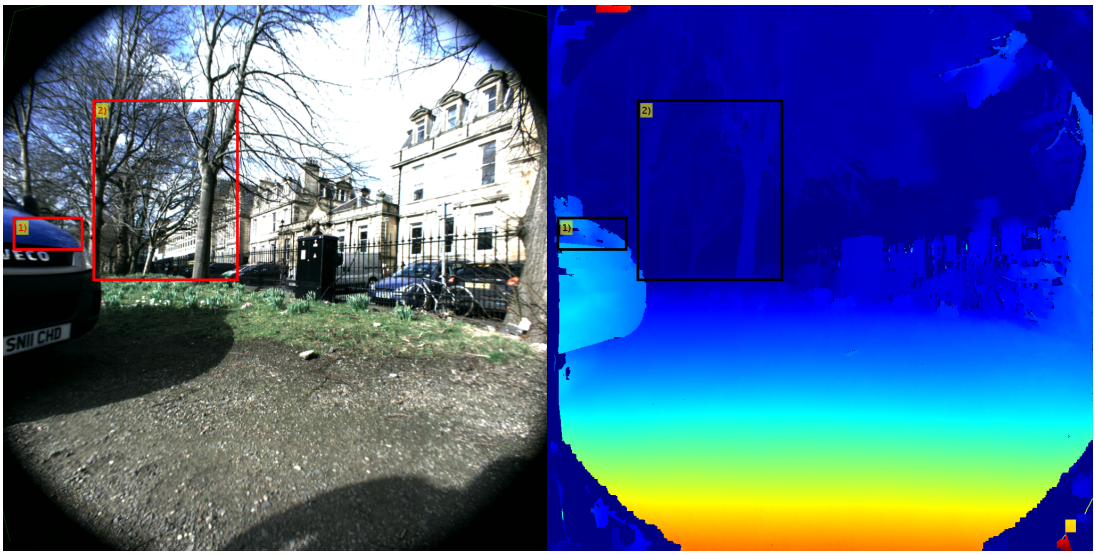


Figure C.12: “Outside 12” test image: Cosine shaded disparity map.



(a) (b)

Figure C.13: “Outside 28” test image ((a)) and *Our* result ((b)).

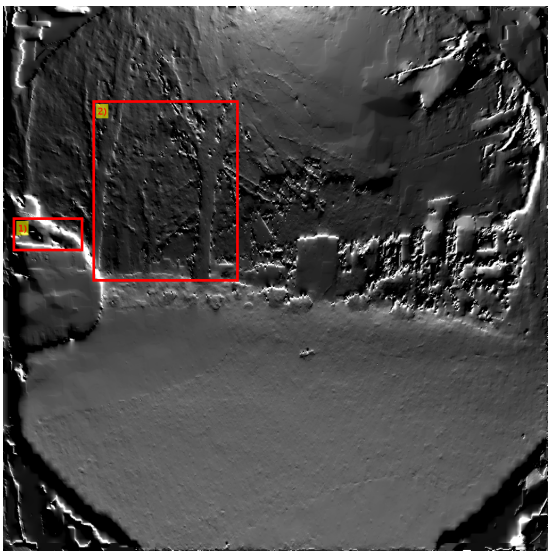


Figure C.14: “Outside 28” test image: Cosine shaded disparity map.

but one part of the fence has been lost due to the lack of texture in that part of the fence. 2) Textured region where our approach has correctly recovered the disparity of the ground as it is well textured and along the epipolar lines. Finally, fig.C.13 shows the results on a image with two regions: 1) Reflective region where our approach was unable to recover correct the disparity of the car, mostly due to the reflections present. 2) Thin background detail regions where our approach has correctly recovered the disparity of the trees while keeping well defined edges.

C.1 Conclusions

The results presented are consistent with the reported results in chapter 5 in both good and bad cases. Furthermore, we have used the same parameters from chapter 5 to keep the comparison valid. This shows that our results are not the consequence from over-fitting, but instead our approach works well with images outside the data sets used for training.

Appendix D

Parameter exploration for TBR

Parameter variation	%bad noc	%bad occ	avg. noc	avg. occ
$\alpha_t = 0.50$	5.78	6.76	1.13	1.35
$\alpha_t = 0.65$	5.10	6.11	0.99	1.25
$\tau^t_{cen} = \tau^b_{cen}$	3.40	4.44	0.72	0.89
$\tau^t_{cen} = 1.4\tau^b_{cen}$	3.28	4.29	0.71	0.88
$\tau^t_{grad} = \tau^b_{grad}$	3.21	4.19	0.71	0.87
$\tau^t_{grad} = 1.66\tau^b_{grad}$	3.41	4.43	0.72	0.90
TBR	**3.07**	**4.13**	**0.69**	**0.86**

Table D.1: TBR parameter comparison.

Tab.D.1 shows how the parameter choice affects the error metrics on the KITTI data set. The last row of tab.D.1 TBR is our algorithm using the parameter settings (as in chapter 6) $\tau^b_{grad} = 3/255$, $\tau^t_{grad} = 1.33\tau^b_{grad}$, $\tau^b_{cen} = 9$, $\tau^t_{cen} = 1.2\tau^b_{cen}$, and $\alpha_t = 0.80$. Notice that using $\alpha_t < 0.80$ results in greatly increased error, because of the scale issue introduced by the transformation used to map points to the third image (as explained in chapter 6). In a similar way τ^t_{grad} and τ^t_{cen} are affected, as low values are unable to provide additional confidence, while high values result in a noisy cost. The parameters used by TBR were selected taking into account the issues described above.

Glossary

Acronyms

CG Graph Cuts.

CPL Combined Potts-Linear model.

CPU Central Processing Unit.

DPI Dynamic Plane Inference.

FE Fattening Effect.

FPGA Field Programmable Gate Array.

FPI Fixed Plane Inference.

GPU Graphic Processing Unit.

IHVMp Independent Horizontal and Vertical Message passing.

LBP Loopy Belief Propagation.

MSSM Multi-Scale Signal Matching.

OG Overgrowth effect.

QPBO Quadratic Pseudo Boolean Optimization.

SGM Semi-Global Matching.

SIMD Single Instruction Multiple Data.

TLM Truncated Linear Model.

TRW Tree Re-Weighted.

Definitions

aggregation Process that adds up all pixel similarity cost values within the same neighbourhood and disparity.

CUDA CUDA is parallel computing platform and programming model invented by NVIDIA (definition according to NVIDIA).

non-submodular refers to functions that do not fulfil all the conditions described in [46], e.g. a truncated metric function.

submodular refers to functions that fulfil the conditions described in [46], e.g. a metric function.

Bibliography

[1] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Süsstrunk. Slic superpixels compared to state-of-the-art superpixel methods. *PAMI*, 34(11):2274–2282, 2012.

[2] S. Agarwal, Y. Furukawa, N. Snavely, I. Simon, B. Curless, S. Seitz, and R. Szeliski. Building rome in a day. *Commun. ACM*, 54(10):105–112, 2011.

[3] M. Bai, W. Luo, K. Kundu, and R. Urtasun. Exploiting semantic information and deep matching for optical flow. *European Conference on Computer Vision*, pages 154–170, 2016.

[4] S. Barnard. Stereo matching by hierarchical microcannonical annealing. Artificial Intelligence Center SRI International, 1987.

[5] C. Barnes, E. Shechtman, A. Finkelstein, and D. Goldman. Patchmatch: A randomized correspondence algorithm for structural image editing. *ACM Trans. Graph.*, 28(3):24:1–24:11, 2009.

[6] F. Besse, C. Rother, A. Fitzgibbon, and J. Kautz. Pmbp: Patchmatch belief propagation for correspondence field estimation. *International Journal of Computer Vision*, 110(1):2–13, 2012.

[7] V. Bevilacqua, G. Mastronardi, F. Menolascina, and D. Nitti. Stereo-matching techniques optimisation using evolutionary algorithms. *Intelligent Computing*, 4113:612–621, 2006.

[8] S. Birchfield and C. Tomasi. A pixel dissimilarity measure that is insensitive to image sampling. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(4):401–406, 1998.

[9] M. Bleyer, C. Rhemann, and C. Rother. Patchmatch stereo - stereo matching with slanted support windows. *In Proceedings of the British Machine Vision Conference*, 11:1–11, 2011.

[10] M. Bleyer, C. Rother, C. Kohli, and D. Scharstein. Object stereo - joint stereo matching and object segmentation. *Conference on Computer Vision and Pattern Recognition*, pages 3081–3088, 2011.

[11] T. Buchanan. Photogrammetry and projective geometry: an historical survey. *Proc. SPIE 1944, Integrating Photogrammetric Techniques with Scene Analysis and Machine Vision*, 8, 1993.

[12] J. Choi and R. Rutenbar. Video-rate stereo matching using markov random field trw-s inference on a hybrid cpu+fpga computing platform. *Proceedings of the ACM/SIGDA International Symposium on Field Programmable Gate Arrays*, pages 63–72, 2013.

[13] C.Menard and W. Kropatsch. Adaptive scale selection for hierarchical stereo. *International Conference on Image Analysis and Processing*, pages 677–684, 2005.

[14] P. Cockshott, S. Oehler, T. Xu, P. Siebert, and G. Aragon. Parallel stereo vision algorithm. *Many-Core Applications Research Community Symposium*, 2012.

[15] J. Davis, S. Marschner, M. Garr, and M. Levoy. Filling holes in complex surfaces using volumetric diffusion. *International Symposium on 3D Data Processing, Visualization, and Transmission*, 2002.

[16] J. Dáz, E. Ros, S. Sabatini, F. Solari, and S. Mota. A phase-based stereo vision system-on-a-chip. *Biosystems*, pages 314–312, 2007.

[17] A. Delaunoy, E. Prados, P. Gargallo, J.-P. Pons, and P. Sturm. Minimizing the multi-view stereo reprojection error for triangular surface meshes. *British Machine Vision Conference*, 2008.

[18] N. Einecke and J. Eggert. Anisotropic median filtering for stereo disparity map refinement. *Vision, Imaging and Applications*, pages 189–198, 2013.

[19] G. Facciolo, C. de Franchis, and E. Meinhardt. Mgm: A significantly more global matching for stereovision. *In Proceedings of the British Machine Vision Conference*, pages 90:1–90:12, 2015.

[20] P. Falkingham, D. Marty, and A. Richter. Dinosaur tracks. 2016.

[21] P. Felzenszwalb and D. Huttenlocher. Efficient belief propagation for early vision. *International Journal of Computer Vision*, 70(1), 2006.

[22] C. for Photogrammetric Training. History of photogrammetry. *Ferris State University (Michigan)*, 2008.

[23] T. Frohlinghaus and J. Buhmann. Regularizing phase-based stereo. *International Conference on Pattern Recognition*, 1996.

[24] A. Geiger, P. Lenz, and R.Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. *Conference on Computer Vision and Pattern Recognition*, pages 3354–336, 2012.

[25] Y. Geng, Y. Zhao, and H. Chen. Improved belief propagation based on rgb vector measure for stereo matching. *Wireless Communications and Signal Processing*, pages 1–5, 2011.

[26] S. Ghosh. Analytical photogrammetry. 1988.

[27] D. Greig, B. Porteous, and A. Seheult. Exact maximum a posteriori estimation for binary images. *Journal of the Royal Statistical Society*, 51(2):271–279, 1989.

[28] F. Guney and A. Geiger. Displets: Resolving stereo ambiguities using object knowledge. *Conference on Computer Vision and Pattern Recognition*, 2015.

[29] R. Gupta and S. Cho. Real-time stereo matching using adaptive binary window. *3D Data Processing, Visualization and Transmission*, 2010.

[30] R. Hartley and A. Zisserman. Multiple view geometry in computer vision. 2004.

[31] K. He, J. Sun, and X. Tang. Guided image filtering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(6):1397–1409, 2013.

[32] P. Heise, S. Klose, B. Jensen, and A. Knoll. Patchmatch with huber regularization for stereo matching. *International Conference on Computer Vision*, pages 2360–2367, 2013.

[33] Y. Heo, K. Lee, and S. Lee. Robust stereo matching using adaptive normalized cross-correlation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(4):807–822, 2011.

[34] H. Hirschmüller. Stereo processing by semiglobal matching and mutual information. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(2):328–341, 2008.

[35] H. Hirschmüller and D. Scharstein. Evaluation of stereo matching costs on images with radiometric differences. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(9):1582–1599, 2009.

[36] L. Hong and G. Chen. Segment-based stereo matching using graph cuts. *Conference on Computer Vision and Pattern Recognition*, 1:74–81, 2004.

[37] L. Horna and R. Fisher. 3d plane labeling stereo matching with content aware adaptive windows. *International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISAPP)*, 2017.

[38] L. Horna and R. Fisher. Plane labeling trinocular stereo matching with baseline recovery. *International Conference on Machine Vision Applications (MVA)*, 2017.

[39] W. Hu, K. Zhang, L. Sun, and J. Li. Virtual support window for adaptive-weight stereo matching. *IEEE Visual Communications and Image Processing*, pages 1–4, 2011.

[40] V. Jacob and S. Gupta. Colorization of grayscale images and videos using a semiautomatic approach. *International Conference on Image Processing (ICIP)*, 2009.

[41] Y. Jen, E. Dunn, P. Fite-Georgel, and J. Frah. Adaptive scale selection for hierarchical stereo. *British Machine Vision Conference*, pages 95.1–95.10, 2011.

[42] A. G. Jones and C. J. Taylor. Scale space surface recovery using binocular shading and stereo information. *British Machine Vision Conference*, pages 77–86, 1995.

[43] A. Klaus, M. Sormann, and K. Karner. Segment-based stereo matching using belief propagation and a self-adapting dissimilarity measure. *International Conference on Pattern Recognition*, 3:15–18, 2006.

[44] V. Kolmogorov. Convergent tree-reweighted message passing for energy minimization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(10):1568–1583, 2007.

[45] V. Kolmogorov and R. Zabih. Computing visual correspondence with occlusions using graph cuts. *International Conference on Computer Vision*, 2:508–515, 2001.

[46] V. Kolmogorov and R. Zabih. Computing visual correspondence with occlusions using graph cuts. *European Conference on Computer Vision*, 3:82–96, 2002.

[47] G. Konecny. Geoinformation: Remote sensing, photogrammetry and geographic information systems. 2014.

[48] J. Kowalczuk, E. Psota, and L. Perez. Real-time stereo matching on cuda using an iterative refinement method for adaptive support-weight correspondences.

IEEE Transactions on Circuits and Systems for Video Technology (TCSVT), 23(1):94–104, 2013.

[49] K. Kutulakos and S. Seitz. Photorealistic scene reconstruction by voxel coloring. *International Journal of Computer Vision*, 1999.

[50] K. Kutulakos and S. Seitz. A theory of shape by space carving. *International Conference on Computer Vision*, 1999.

[51] A. Levin, D. Lischinski, and Y. Weiss. Colorization using optimization. *Proceedings of ACM SIGGRAPH*, 23(3):689–694, 2004.

[52] A. Levin, A. Zomet, and Y. Weiss. Learning how to inpaint from global image statistics. *International Conference on Computer Vision*, 3, 2003.

[53] A. Li, D. Chen, Y. Liu, and Z. Yuan. Coordinating multiple disparity proposals for stereo computation. *Conference on Computer Vision and Pattern Recognition*, 2016.

[54] L. Li, S. Zhang, X. Yu, and L. Zhang. Pmsc: Patchmatch-based superpixel cut for accurate stereo matching. *IEEE Transactions on Circuits and Systems for Video Technology*, 2016.

[55] Y. Li, D. Min, M. S. Brown, M. N. Do, and J. Lu. Sped-up patchmatch belief propagation. *International Conference on Computer Vision*, 2015.

[56] J. Liu, C. Li, F. Mei, and Z. Wang. 3d entity-based stereo matching with ground control points and joint second order smoothness prior. *The Visual Computer*, pages 1–17, 2014.

[57] C. Loop and Z.Zhang. Computing rectifying homographies for stereo vision. *Conference on Computer Vision and Pattern Recognition*, 1999.

[58] J. Lu, H. Yang, D. Min, and M. Do. Patchmatch filter: efficient edge-aware filtering meets randomized search for fast correspondence field estimation. *Conference on Computer Vision and Pattern Recognition*, pages 1854–1861, 2013.

[59] C. Luo, J. Lei, G. Hu, and K. Fan. Stereo matching with semi-limited belief propagation. *Genetic and Evolutionary Computing*, pages 1–4, 2012.

[60] W. Luo, A. Schwing, and R. Urtasun. Efficient deep learning for stereo matching. *Conference on Computer Vision and Pattern Recognition*, 2016.

[61] Z. Ma, K. He, Y. Wei, J. Sun, and E. Wu. Constant time weighted median filtering for stereo matching and beyond. *International Conference on Computer Vision*, pages 49–56, 2013.

[62] E. Magdaleno, J. Lüke, M. Rodrǵuez, and J. Rodrǵuez-Ramos. Design of belief propagation based on fpga for the multistereo cafadis camera. *Sensors (Basel)*, 10, 2010.

[63] D. Marr and T. Poggio. Cooperative computation of stereo disparity. *Science*, 194(4262):231–240, 1976.

[64] X. Mei, X. Sun, W. Dong, H. Wang, and X. Zhang. Segment-tree based cost aggregation for stereo matching. *Conference on Computer Vision and Pattern Recognition*, 2013.

[65] X. Mei, X. Sun, M. Zhou, S. Jiao, H. Wang, and X. Zhang. On building an accurate stereo matching system on graphics hardware. *International Conference on Computer Vision*, pages 467–474, 2011.

[66] M. Menze and A. Geiger. Object scene flow for autonomous vehicles. *Conference on Computer Vision and Pattern Recognition*, 2015.

[67] M. Miura, K. Fudano, K. Ito, T. Aoki, H. Takizawa, and H. Kobayashi. Gpu implementation of phase-based stereo correspondence and its application. *International Conference on Image Processing*, 2012.

[68] Y. Mizukami, K. Okada, A. Nomura, S. Nakanishi, and K. Tadamura. Sub-pixel disparity search for binocular stereo vision. *International Conference on Pattern Recognition*, pages 364–367, 2012.

[69] P. Monasse. Quasi-euclidean epipolar rectification. *Image Processing On Line*, 2011.

[70] J. Morel and G.Yu. Asift: A new framework for fully affine invariant image comparison. *SIAM Journal on Imaging Sciences*, 2(2):438–469, 2009.

[71] M. Mozerov and J. van Weijer. Accurate stereo matching by two step global optimization. *Submitted to Transactions on Image Processing*, 2014.

[72] V. Muninder, U. Soumik, and A. Krishna. Robust segment-based stereo using cost aggregation. *Proceedings of the British Machine Vision Conference*, 2014.

[73] C. Olsson, J. Ulén, and Y. Boykov. In defense of 3d-label stereo. *Conference on Computer Vision and Pattern Recognition*, pages 1730–1737, 2013.

[74] C. Pal, J. Weinman, L. Tran, and D. Scharstein. On learning conditional random fields for stereo. *International Journal of Computer Vision*, 99(3):319–337, 2012.

[75] M.-G. Park and K.-J. Yoon. As-planar-as-possible depth map estimation. *Submitted to IEEE TPAMI*, 2016.

[76] J. Perez, P. Sanchez, and M. Martinez. High memory throughput fpga architecture for high-definition belief-propagation stereo matching. *International Conference on Signals, Circuits and Systems*, pages 1–6, 2009.

[77] J. Porrill, S. Pollard, T. Pridmore, J. Bowen, J. Mayhew, and J. Frisby. Tina: a 3d vision system for pick and place. *Journal of the Royal Statistical Society*, 6(2):91–99, 1988.

[78] E. Psarakis and G. Evangelidis. An enhanced correlation-based method for stereo correspondence with subpixel accuracy. *International Conference on Computer Vision*, 1:907–912, 2005.

[79] C. Rother, V. Kolmogorov, V. Lempitsky, and M. Szummer. Optimizing binary mrfs via extended roof duality. *Conference on Computer Vision and Pattern Recognition*, 2007.

[80] A. Rövid and T. Hashimoto. On phase based stereo matching and its related issues. *International Conference on Computer Science and Information Engineering*, 2014.

[81] D. Scharstein, H. Hirschmüller, Y. Kitajima, G. Krathwohl, N. Nesic, X. Wang, and P. Westling. High-resolution stereo datasets with subpixel-accurate ground truth. *In German Conference on Pattern Recognition (GCPR)*, pages 31–42, 2014.

[82] D. Scharstein and R. Szeliski. A taxonomy and evaluation of dense two frame stereo correspondence algorithms. *International Journal of Computer Vision*, 47(1–3):7–42, 2002.

[83] C. Shi, G. Wang, X. Pei, H. Bei, and X. Lin. High-accuracy stereo matching based on adaptive ground control points. *IEEE Transactions on Image Processing*, 2012.

[84] J. P. Siebert and C. W. Urquhart. C3d: a novel vision-based 3-d data acquisition system. *Image Processing for Broadcast and Video Production*, pages 170–180, 1995.

[85] S. Sinha, D. Scharstein, and R. Szeliski. Efficient high-resolution stereo matching using local plane sweeps. *Conference on Computer Vision and Pattern Recognition*, pages 1582–1589, 2014.

[86] J. P. Starink and E. Backer. Finding point correspondences using simulated annealing. *Pattern Recognition*, 28(2):231–240, 1995.

[87] L. Tang, M. Garvin, K. Lee, W. Alward, Y. Kwon, and M. Abràmoff. Robust multiscale stereo matching from fundus images with radiometric difference. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(11):2245–2258, 2011.

[88] T. Taniai, Y. Matsushita, and T. Naemura. Graph cut based continuous stereo matching using locally shared labels. *Conference on Computer Vision and Pattern Recognition*, pages 1613–1620, 2014.

[89] M. Tappen and W. Freeman. Comparison of graph cuts with belief propagation for stereo using identical mrf parameters. *International Conference on Computer Vision*, 3, 2003.

[90] E. Tola, V. Lepetit, and P. Fua. Daisy: An efficient dense descriptor applied to wide baseline stereo. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(5):815–830, 2010.

[91] M. Trager, J. Ponce, and M. Hebert. Trinocular geometry revisited. *International Journal of Computer Vision*, 120(2):134–152, 2016.

[92] B. Triggs, P. McLauchlan, R. Hartley, and A. Fitzgibbon. Bundle adjustment — a modern synthesis. *Vision Algorithms: Theory and Practice: International Workshop on Vision Algorithms*, pages 298–372, 2000.

[93] A. Vedaldi and B. Fulkerson. VLFeat: An open and portable library of computer vision algorithms. `http://www.vlfeat.org/`, 2008.

[94] A. Vedaldi and S. Soatto. Quick shift and kernel methods for mode seeking. *European Conference on Computer Vision*, 5305:705–718, 2008.

[95] O. Veksler. Efficient graph-based energy minimization methods in computer vision. phd thesis. Cornell University, 1999.

[96] C. Vogel, K. Schindler, Konrad, and S. Roth. 3d scene flow estimation with a piecewise rigid scene model. *International Journal of Computer Vision*, pages 1–28, 2015.

[97] J. Žbontar and Y. LeCun. Stereo matching by training a convolutional neural network to compare image patches. *Submitted to JMLR*, 2015.

[98] D. Wang and K. Lim. Obtaining depth map from segment-based stereo matching using graph cuts. *Journal of Visual Communication and Image Representation*, 22(4):325–331, 2011.

[99] L. Wang, M. Liao, M. Gong, R. Yang, and D. Nister. High-quality real-time stereo using adaptive cost aggregation and dynamic programming. *3DPVT*, pages 98–805, 2006.

[100] L. Wang, M. Liao, M. Gong, R. Yang, and D. Nistr. High-quality real-time stereo using adaptive cost aggregation and dynamic programming. *International Symposium on 3D Data Processing, Visualization, and Transmission*, pages 798–805, 2006.

[101] L. Wang and R. Yang. Global stereo matching leveraged by sparse ground control points. *Conference on Computer Vision and Pattern Recognition*, pages 3033–3040, 2011.

[102] L. Wang and R. Yang. Global stereo matching leveraged by sparse ground control points. *Conference on Computer Vision and Pattern Recognition*, pages 3033–3040, 2011.

[103] G. Whitten. Scale space tracking and deformable sheet models for computational vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(7):697–706, 1993.

[104] O. J. Woodford, I. D. Reid, P. H. S. Torr, and A. W. Fitzgibbon. On new view synthesis using multiview stereo. *British Machine Vision Conference*, pages 1–10, 2007.

[105] Y. Xu, J. Zhou, and G. Zhai. 2d phase-based matching in uncalibrated images. *Workshop on Signal Processing Systems Design and Implementation*, 2005.

[106] K. Yamaguchi, D. McAllester, and R. Urtasun. Efficient joint segmentation, occlusion labeling, stereo and flow estimation. *European Conference on Computer Vision*, pages 756–771, 2014.

[107] D. Yang, L. Chu, C. Chen, and J. Gan. Low complexity stereo matching algorithm using adaptive sized square window. *VLSI Design, Automation and Test*, pages 1–4, 2014.

[108] Q. Yang. A non-local cost aggregation method for stereo matching. *Conference on Computer Vision and Pattern Recognition*, 2012.

[109] Q. Yang. Recursive bilateral filtering. *European Conference on Computer Vision*, pages 399–413, 2012.

[110] Q. Yang, L. Wang, R. Yang, H. Stewnius, and D. Nistr. Stereo matching with color-weighted correlation, hierarchical belief propagation and occlusion handling. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(3):492–504, 2009.

[111] Q. Yang, R. Yang, J. Davis, and D. Nistér. Spatial-depth super resolution for range images. *Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2007.

[112] K. Yoon and I. Kweon. Adaptive support-weight approach for correspondence search. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(4):650–656, 2006.

[113] T. Yu, R. Lin, B. Super, and B.Tang. Efficient message representations for belief propagation. *International Conference on Computer Vision*, 8(1):14–21, 2007.

[114] Y.Weiss and W. Freeman. On the optimality of solutions of the max-product belief propagation algorithm in arbitrary graphs. *IEEE Transactions on Information Theory*, 47(2):736–744, 2001.

[115] R. Zabih and J. Li. Non-parametric local transforms for computing visual correspondence. *European Conference on Computer Vision*, 12:151–158, 1994.

[116] S. Zagoruyko and N. Komodakis. Learning to compare image patches via convolutional neural networks. *Conference on Computer Vision and Pattern Recognition*, pages 4353–4361, 2015.

[117] C. Zhang, Z. Li, Y. Cheng, R. Cai, H. Chao, and Y. Rui. Meshstereo: A global stereo model with mesh alignment regularization for view interpolation. *International Conference on Computer Vision*, 2015.

[118] H. Zhang, F. Cheng, D. Yuan, and Y. Li. Stereo matching with global edge constraint and graph cuts. *International Conference on Pattern Recognition*, pages 372–375, 2012.

[119] 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*, 19(7):1073–1079, 2009.

[120] M. Zhao and R. Chung. Critical configurations of lines to geometry determination of three cameras. *International Conference on Pattern Recognition*, pages 1–5, 2008.