2007

3D RECONSTRUCTION FROM STEREO/RANGE IMAGES

Qingxiong Yang
University of Kentucky, qyang2@uky.edu

ABSTRACT OF THESIS

3D RECONSTRUCTION FROM STEREO/RANGE IMAGES.

3D reconstruction from stereo/range image is one of the most fundamental and extensively researched topics in computer vision. Stereo research has recently experienced somewhat of a new era, as a result of publically available performance testing such as the Middlebury data set, which has allowed researchers to compare their algorithms against all the state-of-the-art algorithms. This thesis investigates into the general stereo problems in both the two-view stereo and multi-view stereo scopes.

In the two-view stereo scope, we formulate an algorithm for the stereo matching problem with careful handling of disparity, discontinuity and occlusion. The algorithm works with a global matching stereo model based on an energy minimization framework. The experimental results are evaluated on the Middlebury data set, showing that our algorithm is the top performer. A GPU approach of the Hierarchical BP algorithm is then proposed, which provides similar stereo quality to CPU Hierarchical BP while running at real-time speed. A fast-converging BP is also proposed to solve the slow convergence problem of general BP algorithms. Besides two-view stereo, efficient multi-view stereo for large scale urban reconstruction is carefully studied in this thesis. A novel approach for computing depth maps given urban imagery where often large parts of surfaces are weakly textured is presented. Finally, a new post-processing step to enhance the range images in both the both the spatial resolution and depth precision is proposed.

(Qingxiong Yang)

(Date)

3D RECONSTRUCTION FROM STEREO/RANGE IMAGES.

By

Qingxiong Yang

____

(Director of Thesis)

____

(Director of Graduate Studies)

____

(Date)

RULES FOR THE USE OF THESES

Unpublished theses submitted for the Master's degree and deposited in the University of Kentucky Library are as a rule open for inspection, but are to be used only with due regard to the rights of the authors. Bibliographical references may be noted, but quotations or summaries of parts may be published only with the permission of the author, and with the usual scholarly acknowledgments.

Extensive copying or publication of the theses in whole or in part also requires the consent of the Dean of the Graduate School of the University of Kentucky.

A library that borrows this thesis for use by its patrons is expected to secure the signature of each user.

Name	Date

THESIS

Qingxiong Yang

The Graduate School

University of Kentucky

2007

3D RECONSTRUCTION FROM STEREO/RANGE IMAGES.

THESIS

A thesis submitted in partial fulfillment of the
requirements for the degree of Master of Science
at the University of Kentucky

By

Qingxiong Yang

Lexington, Kentucky

Director: Dr. David Nistér, Professor of Computer Science

Lexington, Kentucky

2007

Acknowledgments

I thank Dr. David Nistér, my supervisory committee chair for his immeasurable support, guidance, and encouragement. I thank Dr. Ruigang Yang and Dr. Fuhua (Frank) Cheng for serving on my supervisory committee.

I thank my fellow students at the Center for visualization and virtual environments. From them I learned a great deal about computer vision, and found great friendships. I thank my family for their undying love and guidance. Without them, I would not be the person I am today.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

The main purpose of this thesis is to study the methodologies of automatically computing a three-dimensional reconstruction of the scene given two or multiple images taken from different viewpoints of the scene. The methodologies are in general called Shape from X in the literature, where X stands for various cues such as motion, shading, textures, lasers, structured light, focus, line drawings, etc. It is also possible to fuse different cues within a shape from X algorithm to enhance robustness. This thesis focus on 3D reconstruction problems belonging to the category shape from motion or structure from motion from a purely geometric viewpoint. In the thesis, the majority of the scene is rigid, which means that either the scene has to be rigid or the images should be taken at the same time.

This thesis is focus on automatic shape from motion, (also call 3D reconstruction from motion) and is organized in the way of solving the problem from easy to hard: Chapter 2 presents a novel stereo algorithm with top quality performance but time consuming within the two-view stereo scope. To accelerate the speed, a real-time hierarchical BP approach is proposed in Chapter 3. Chapter 4 further investigates into the problem of automatic 3D reconstruction from multi-view stereo images. A stereo pipeline that is able to handle large weakly-textured areas while running at near real-time speed is presented. It uses an approximate but much faster stereo algorithm of the one presented in Chapter 2. Finally, a new post-processing step to enhance the range images in both the both the spatial resolution and depth precision is presented in Chapter 5.

1. Two-view reconstruction: Stereo Matching with Color-Weighted Correlation, Hierarchical Belief Propagation and Occlusion Handling

   In this chapter, we formulate an algorithm for the stereo matching problem with careful handling of disparity, discontinuity and occlusion. The algorithm works with a global matching stereo model based on an energy-minimization framework. The global energy contains two terms, the data term and the smoothness term. The data term is first approximated by a color-weighted correlation, then refined in occluded and low-texture areas in a repeated application of a hierarchical loopy belief propagation (BP) algorithm. The experimental results are evaluated on the Middlebury data set, showing that our algorithm is the top performer.

2. Two-view reconstruction: Real-time Global Stereo Matching Using Hierarchical Belief Propagation

   In this chapter, we present a belief propagation based global algorithm that generates high quality results while maintaining real-time performance. To our knowledge, it is the first BP based global method that runs at real-time speed. Our efficiency performance gains mainly from the parallelism of graphics hardware,which leads to a 45 × speedup compared to the

CPU implementation. To qualify the accuracy of our approach, the experimental results are evaluated on the Middlebury data sets, showing that our approach is among the best (ranked first in the new evaluation system) for all real-time approaches. In addition, since the running time of general BP is linear to the number of iterations, adopting a large number of iterations is not feasible for practical applications. Hence a novel approach is proposed to adaptively update pixel cost. Unlike general BP methods, the running time of our proposed algorithm dramatically converges.

3. Multi-view Stereo: Near Real-time Robust Plane-fitting Stereo for Weakly-Textured Urban Imagery

   There is a growing interest in automatic 3D reconstruction of large scale scenes, especially for urban environments. Because of the huge amount of video data associated with capturing these scenes, a fast stereo algorithm is required. While several state-of-the-art real-time/near real-time stereo algorithms provide accurate 3D reconstructions for well-textured scenes, most of them fail for surface parts that are weakly-textured. In this chapter, we develop a stereo pipeline that is able to handle this problem while running at near real-time speed, more specifically around 8 frames per second for an image resolution of 512×384 and 48 depth hypotheses. Our stereo pipeline segments the image via a novel real-time color segmentation algorithm; each segment is subsequently fitted to a plane and refined using consistency constraints. To further improve the quality of our stereo algorithm, we integrated BP algorithm to correct potential errors caused by plane-fitting due to non-robustness of color segmentation; however at the cost of speed performance. By comparing the two proposed stereo approaches with a local window-based approach we show that the proposed algorithms perform very well in weakly-textured areas while maintaining a good speed performance. 100× with a single reference image.

4. Reconstruction from Range image: Spatial-Depth Super Resolution for Range Images

   We present a new post-processing step to enhance the resolution of range images. Using one or two registered and potentially high-resolution color images as reference, we iteratively refine the input low-resolution range image, in terms of both its spatial resolution and depth precision. Evaluation using the benchmark Middlebury data set shows across-the-board improvement for sub-pixel accuracy. We also demonstrated its effectiveness for spatial resolution enhancement up to 100× with a single reference image.

Chapter 2

Two-View Stereo: Stereo Matching with Color-Weighted Correlation, Hierarchical Belief Propagation and Occlusion Handling

2.1 Introduction

Stereo is one of the most extensively researched topics in computer vision. Stereo research has recently experienced somewhat of a new era, as a result of publically available performance testing such as the middlebury data set [42], which has allowed researchers to compare their algorithms against all the state-of-the-art algorithms.

In this chapter, we describe our stereo algorithm, which is currently evaluating as the top performer on the middlebury data set. The algorithm springs from the popular energy minimization framework that is the basis for most of the algorithms on the middlebury top-list, such as graph cuts [10, 34] and belief propagation [48, 47]. In this framework, there is typically a data term and a smoothness term, where the data term consists of the matching error implied by the extracted disparity map, and the smoothness term encodes the prior assumption that world surfaces are piecewise smooth.

However, the algorithm presented in this chapter departs somewhat from the normal framework, in that in the final stages of the algorithm, the data term is updated based on the current understanding of which pixels in the reference image are occluded or unstable due to low texture.

The chapter is organized as follows: Section 2.2 gives a high-level overview of the approach. In Section 2.3 we then give the detailed equations for all the building blocks. Section 2.4 reports results supporting the claims that the algorithm is currently the strongest available on the middlebury data set. Section 2.5 concludes.

2.2 Overview of the Approach

The algorithm can be partitioned into three blocks, initial stereo (Figure 2.1), pixel classification (Figure 2.2) and iterative refinement (Figure 2.3). In the initial stereo, see Figure 2.1, the correlation volume is first computed. A basic way to construct the correlation volume is to compute the absolute difference of luminances of the corresponding pixels in the left and right images, but there are many other methods for correlation volume construction. For instance, Sun et al. [48] use Birchfield and Tomasi's pixel dissimilarity [8] to construct the correlation volume, and Felzenszwalb [18] suggests to smooth the image first before calculating the pixel difference. In this work, we are using color-weighted correlation to build the correlation volume, in a similar manner as was recently described by Yoon and Kweon [59]. The color-weighting makes the match scores less sensitive to occlusion boundaries by using the fact that occlusion boundaries most often cause color discontinuities as well. The initial stereo is run in turn with both the left and the right image as reference images. This is done just to support a subsequent mutual consistency check (often called left-right check) that takes place in the pixel classification block. Functions E_S^L and E_S^R defining the smoothness costs in the left and right reference images, respectively, are determined based on the color gradients in the input images. The left and right smoothness costs and the left and right correlation costs are then optimized using two separate hierarchical belief propagation processes. The hierarchical belief propagation is performed in a manner similar to Felzenszwalb [18], resulting in the initial left and right disparity maps $D_L^{(0)}$ and D_R, respectively. The left disparity map is given an iteration index $i = 0$ here, because it will be repeatedly refined in the iterative refinement module. The outputs needed from the initial stereo are the initial left and right disparity maps $D_L^{(0)}$ and D_R, the left correlation volume C_L, the left image I_L and smoothness cost E_S^L.

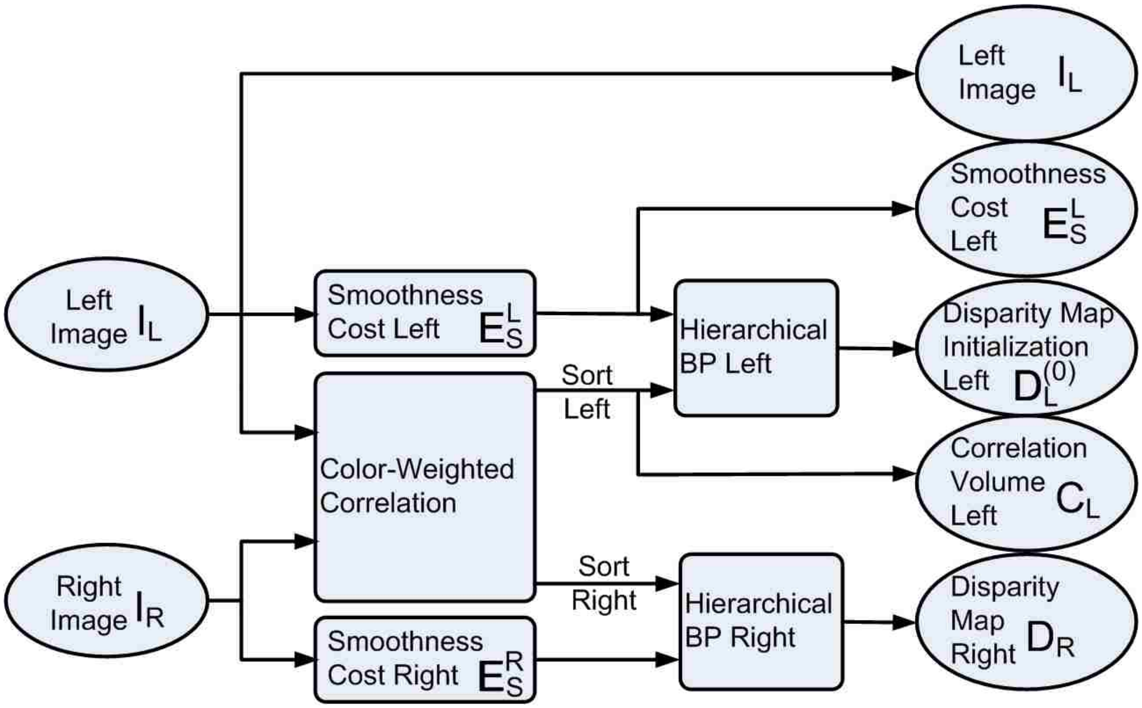

Left Image I_L
Right Image I_R
Smoothness Cost Left E_S^L
Color-Weighted Correlation
Smoothness Cost Right E_S^R
Sort Left
Sort Right
Hierarchical BP Left
Hierarchical BP Right
Left Image I_L
Smoothness Cost Left E_S^L
Disparity Map Initialization Left $D_L^{(0)}$
Correlation Volume Left C_L
Disparity Map Right D_R

Figure 2.1: The initial stereo module. Hierarchical belief propagation is run with both the left and right images as reference image. The data term used is based on the color-weighted correlation, and the smoothness term is computed based on the color gradients in the reference image, see the text for more details.

In the pixel classification module, see Figure 2.2, pixels are given one out of three possible labels: occluded, stable or unstable. The occluded pixels are the ones that fail the mutual consistency check that is performed using the left and right disparity maps. The pixels that pass the mutual consistency check are then labeled stable or unstable based on a confidence measure derived from the left correlation volume, which measures if the peak in the correlation score is distinct enough that the local disparity can be considered stable. The output from the pixel classification module is simply the pixel class membership.

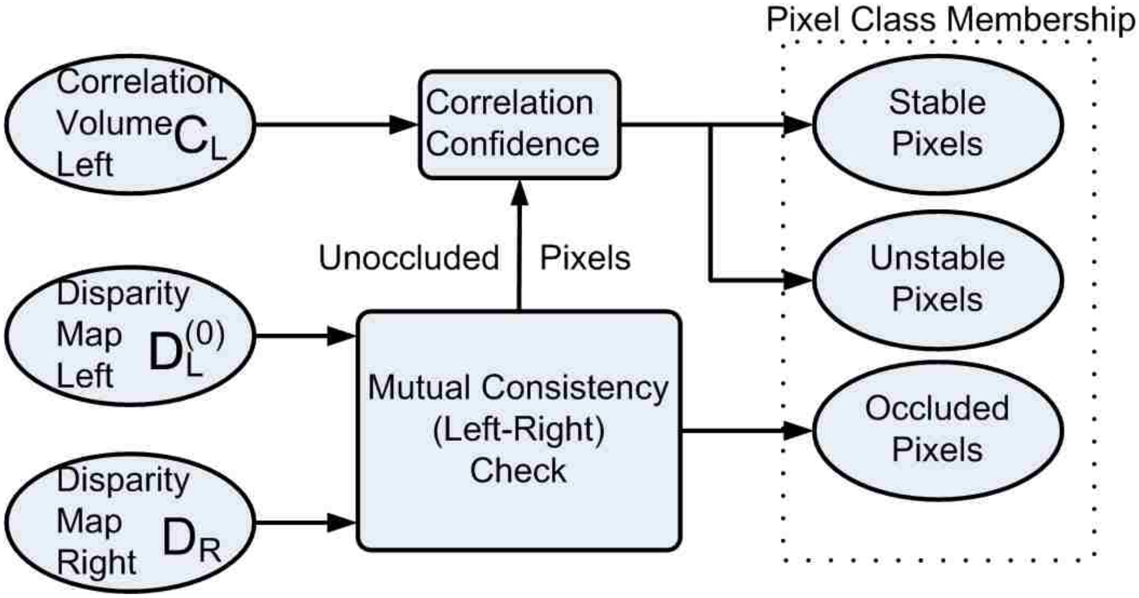

Pixel Class Membership
Correlation Volume Left C_L
Correlation Confidence
Stable Pixels
Unoccluded Pixels
Unstable Pixels
Disparity Map Left $D_L^{(0)}$
Mutual Consistency (Left-Right) Check
Occluded Pixels
Disparity Map Right D_R

Figure 2.2: The pixel classification module. Pixels are classified into occluded pixels, unstable pixels and stable pixels. The occluded pixels are the ones that fail a mutual consistency check. The unoccluded pixels are then further divided into stable and unstable pixels based on a confidence measure derived from the correlation volume.

In the iterative refinement module, see Figure 2.3, the left smoothness cost E_S^L, initial left disparity map $D_L^{(0)}$, left image I_L, pixel class membership and left correlation volume C_L are all used as input. The goal here is to propagate information from the stable pixels to the unstable and the occluded pixels. This is done using color segmentation and plane fitting in a way inspired by [50]. In our work, we use color segments extracted by mean shift [14] applied to the left input image. In each color segment, the disparity values for the stable pixels are used in a plane fitting procedure. Note that the disparity values used here are taken from the current hypothesis $D_L^{(i)}$ for the left disparity map. This disparity map is first initialized with the left disparity map $D_L^{(0)}$ given by the initial stereo module. The result of plane fitting within color segments is then used together with the pixel class membership and the left correlation volume to give the current data term hypothesis $E_D^{(i+1)}$, which is used with the original smoothness cost E_S^L in hierarchical belief propagation. Effectively, the plane fitted depth map is used as a regularization for the new disparity estimation. The hierarchical belief propagation yields the updated disparity map hypothesis $D_L^{(i+1)}$, which is iteratively fed back into the plane fitting.

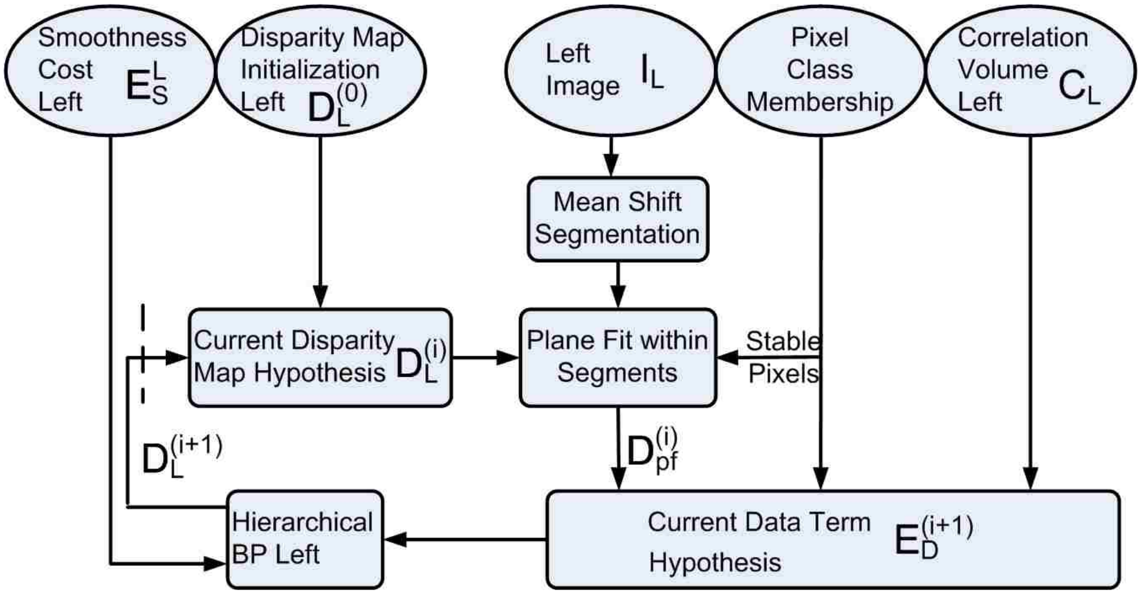

Smoothness Cost Left E_S^L
Disparity Map Initialization Left $D_L^{(0)}$
Left Image I_L
Pixel Class Membership
Correlation Volume Left C_L
Mean Shift Segmentation
Current Disparity Map Hypothesis $D_L^{(i)}$
Plane Fit within Segments
Stable Pixels
$D_L^{(i+1)}$
$D_{pf}^{(i)}$
Hierarchical BP Left
Current Data Term Hypothesis $E_D^{(i+1)}$

Figure 2.3: The iterative refinement block. The goal is to propagate information from the stable pixels to the unstable and the occluded pixels. Mean shift color segmentation is used to derive segments. Within each segment plane fitting is then applied to the stable pixels, using the depth values from the current disparity map hypothesis. The result $D_{pf}^{(i)}$ from the plane fitting is then used together with the correlation volume and the pixel class membership to produce a new approximation $E_D^{(i+1)}$ of the data term. The data term is used with the original smoothness term in another round of hierarchical belief propagation. This gives a new disparity map hypothesis $D_L^{(i+1)}$, which is fed back into the process.

2.3 Detailed Description

In this section, we give a more detailed description of the building blocks outlined above. The order of description follows the above outline through Figures 2.1,2.2 and 2.3.

2.3.1 Initial Stereo

The main building blocks of the initial stereo module, see Figure 2.1, are color-weighted correlation, smoothness cost definition and hierarchical belief propagation.

The objective of the color-weighted cost aggregation is to initialize a reliable correlation volume. To obtain more accurate results on both smooth and discontinuous regions, an appropriate window should be selected adaptively for each pixel during the cost aggregation step. That is, the window should be large enough to cover sufficient area in textureless regions, while small enough to avoid crossing depth discontinuities. Many methods [30, 11, 51, 52, 26, 9] have been proposed to solve this ambiguity problem.

In our implementation, we use an amended version of the color based weight approach proposed recently by Yoon and Kweon [59]. In this method, instead of finding an optimal support window, adaptive support-weights are assigned to pixels in some large window with side-length α_{cw} based both on the color proximity and the spatial proximity to the pixel under consideration (the central pixel of the support window).

In Yoon and Kweon's work, the similarity between two pixels within the support window is measured in the CIELab color space. Our approach however simply measures it in the RGB color space. Due to our post-refinement process, this change does not prevent us from achieving state-of-the-art results. However, instead of using a raw pixel difference, we use Birchfield and Tomasi's pixel dissimilarity [8].

The color difference $\Delta_{\mathbf{xy}}$ between pixel $\mathbf{x}$ and $\mathbf{y}$ (in the same image) is expressed as

$$\Delta_{\mathbf{xy}} = \sum_{c\in\{r,g,b\}} |I_c(\mathbf{x}) - I_c(\mathbf{y})|, \tag{2.1}$$

where I_c is the intensity of the color channel c. The weight of pixel $\mathbf{x}$ in the support window of $\mathbf{y}$ (or vice versa) is then determined using both the color and spatial differences as

$$w_{\mathbf{xy}} = e^{-(\beta_{cw}^{-1}\Delta_{\mathbf{xy}} + \gamma_{cw}^{-1}\|\mathbf{x}-\mathbf{y}\|_2)}, \tag{2.2}$$

where β_{cw} and γ_{cw} are parameters determined empirically.

The data term is then an aggregation with the soft windows defined by the weights, as

$$C(\mathbf{x_L},\mathbf{x_R}) = \frac{\sum_{(\mathbf{y}_L,\mathbf{y}_R)\in W_{\mathbf{x}_L}\times W_{\mathbf{x}_R}} w_{\mathbf{x}_L\mathbf{y}_L} w_{\mathbf{x}_R\mathbf{y}_R} d(\mathbf{y}_L,\mathbf{y}_R)}{\sum_{(\mathbf{y}_L,\mathbf{y}_R)\in W_{\mathbf{x}_L}\times W_{\mathbf{x}_R}} w_{\mathbf{x}_L\mathbf{y}_L} w_{\mathbf{x}_R\mathbf{y}_R}}, \tag{2.3}$$

where $W_{\mathbf{x}}$ is the support window around $\mathbf{x}$ and $d(\mathbf{y}_L, \mathbf{y}_R)$ represents Birchfield and Tomasi's pixel dissimilarity, $\mathbf{x}_L$ and $\mathbf{y}_L$ are pixels in the left image $\mathbf{I}_L$, $\mathbf{x}_R$ and $\mathbf{y}_R$ are pixels in the right image $\mathbf{I}_R$.

The smoothness cost should be decreased at depth edges, since these are likely to coincide with color edges, the luminance difference

$$\delta_{\mathbf{xy}} = |I(\mathbf{x}) - I(\mathbf{y})| \tag{2.4}$$

between neighboring pixels $\mathbf{x}$ and $\mathbf{y}$ is used to decrease the cost. The difference $\delta_{\mathbf{xy}}$ is normalized to span the interval $[0, 1]$. The average over the whole frame is then subtracted out to yield the normalized difference δ_{norm}. Defining the cost coefficient

$$\rho_s = 1 - \delta_{norm}, \tag{2.5}$$

the cost assigned to the pixel pair $(\mathbf{x}, \mathbf{y})$ is then

$$E_S = \rho_{bp}\rho_s|D(\mathbf{x}) - D(\mathbf{y})|, \tag{2.6}$$

where ρ_{bp} is set empirically and $D(\mathbf{x})$ and $D(\mathbf{y})$ are the disparities of $\mathbf{x}$ and $\mathbf{y}$.

Hierarchical loopy belief propagation [18] is employed to realize the iterative optimization that trades off between the data and the smoothness term. The difference between the hierarchical BP and general BP is that the hierarchical BP works in a coarse-to-fine manner, first performing BP at the coarsest scale, then using the output from the coarser scale to initialize the input for the next scale. Two main parameters s_{bp} and n_{bp} define the behavior of this hierarchical belief propagation algorithm, s_{bp} is the number of scales and n_{bp} is the number of iterations in each scale.

2.3.2 Pixel Classification

The main building blocks of the pixel classification, see Figure 2.2, are the mutual consistency check and the correlation confidence measure.

The mutual consistency check requires that on the pixel grid that the left and right disparity maps are computed, they are perfectly consistent, i.e. that

$$D_L(\mathbf{x}_L) = D_R(\mathbf{x}_L - D_L(\mathbf{x}_L)) \tag{2.7}$$

for a particular pixel $\mathbf{x}_L$ in the left image. If this relation does not hold, the pixel is declared occluded. If it does hold, the pixel is declared unoccluded and passed on to the correlation confidence measure.

The correlation confidence is measuring how distinct the best peak in the correlation is for a particular pixel. Assume that the cost for the best disparity value is C_1, and the cost for the second best disparity value is C_2. The correlation confidence is then

$$|\frac{C_1 - C_2}{C_2}|. \tag{2.8}$$

If it is above a threshold α_s the pixel is declared stable, otherwise unstable.

2.3.3 Iterative Refinement

The main building blocks of the iterative refinement, see Figure 2.3, are the mean shift color segmentation, plane fitting within segments, the data term formulation, and another hierarchical belief propagation process identical to the previous ones.

The mean shift color segmentation is performed as described in [14].

The plane fitting is performed in the disparity space, and is applied per segment. This is done robustly using acm-1981-fischler [20] on the disparity values of the stable pixels only. The output $D_{pf}^{(i)}$ from this step is computed individually for each segment and depends on on the ratio of stable pixels of this segment. If the ratio of stable pixels is above a parameter value η_s, this means most of the current disparity values for the segment are approximated accurately so we use $D_L^{(i)}$ for the stable pixels, and for the unstable and occluded pixels we use the result of the plane fitting. If the ratio of stable pixels is below η_s we use the result of the plane fitting for all pixels.

Mean Shift	α_{ms}	β_{ms}	γ_{ms}			
Segmentation	7	6	50			
Color-Weigh.	α_{cw}	β_{cw}	γ_{cw}			
Correlation	33	10	21			
Hierarchical	α_{bp}	η_{bp}	ρ_{bp}	λ_{bp}	s_{bp}	n_{bp}
BP	$n_d/8$	$2\overline{c}$	1	0.2	5	5
Iterative	κ_s	κ_u	κ_o	α_s	η_s	n_s
Refinement	0.05	0.5	2	0.04	0.7	5

Table 2.1: Parameter settings used throughout. n_d is the number of disparity levels. $\overline{c}$ is the average of the values in the correlation volume.

The data term is formulated differently for the occluded, unstable and stable pixels. The absolute difference

$$a_i = |D_L^{(i+1)} - D_{pf}^{(i)}| \tag{2.9}$$

between the new disparity map $D_L^{(i+1)}$ and the plane fitted disparity map $D_{pf}^{(i)}$ is used to regularize the new estimation process. The difference is used to define the data term at the occluded, unstable and stable pixels as

$$\begin{aligned} E_D^{(i+1)} &= \kappa_o a_i, & (2.10) \\ E_D^{(i+1)} &= C_L + \kappa_u a_i, & (2.11) \\ E_D^{(i+1)} &= C_L + \kappa_s a_i, & (2.12) \end{aligned}$$

respectively. The constants $\kappa_o, \kappa_u, \kappa_s$ reflect the fact that the unstable and occluded pixels need the most regularization.

2.3.4 Parameter Settings

In this section, we provide all the parameter settings used in the algorithm. The same parameter settings were used throughout.

The parameters are shown in Table 2.1 and separated into 4 parts: 3 parameters (α_{ms}, β_{ms}, γ_{ms}) for the mean shift segmentation, 3 parameters (α_{cw}, β_{cw}, γ_{cw}) for color-weighted correlation, 6 parameters (α_{bp}, η_{bp}, ρ_{bp}, λ_{bp}, s_{bp}, n_{bp}) for hierarchical belief propagation, and 6 parameters (κ_s, κ_u, κ_o, α_s, η_s, n_s) for iterative refinement.

For mean shift color segmentation, α_{ms} is spatial bandwidth, β_{ms} is color bandwidth, and γ_{ms} is the minimum region size.

For color-weighted correlation α_{cw} is the size of the support window and β_{cw} and γ_{cw} are defined in Equation (2.2).

For hierarchical BP, α_{bp} and η_{bp} are truncations of the smoothness and data terms, respectively. The parameter ρ_{bp} is the constant weight factor applied to the smoothness term and λ_{bp} is a constant weight factor applied to the data term after the truncation. The parameter s_{bp} is the number of scales and n_{bp} is the number of iterations, as defined in Section 2.3.1.

Parameters κ_s, κ_u and κ_o for the iterative refinement are defined in Equations (2.10), (2.11) and (2.12), respectively. α_s is the threshold on correlation confidence defined in Section 2.3.2. Parameter η_s is related to the plane fitting process, as defined in Section 2.3.3. The parameter n_s is the number of iterations for the iterative refinement process.

2.4 Experimental Results

	Avg.	Tsukuba			Venus			Teddy			Cones		
Algorithm	Rank	nonocc	all	disc	nonocc	all	disc	nonocc	all	disc	nonocc	all	disc
Our Algorithm	**1.3**	**0.88** $_1$	**1.29** $_1$	**4.76** $_1$	**0.14** $_1$	0.60 $_2$	**2.00** $_1$	**3.55** $_1$	8.71 $_2$	**9.70** $_1$	**2.90** $_1$	9.24 $_2$	**7.80** $_1$
Segm+visib [9]	3.3	1.30 $_5$	1.57 $_2$	6.92 $_6$	0.79 $_4$	1.06 $_3$	6.76 $_6$	5.00 $_2$	**6.54** $_1$	12.3 $_2$	3.72 $_3$	**8.62** $_1$	10.2 $_4$
SymBP+occ [47]	3.4	0.97 $_2$	1.75 $_3$	5.09 $_2$	0.16 $_2$	**0.33** $_1$	2.19 $_2$	6.47 $_4$	10.7 $_3$	17.0 $_4$	4.79 $_7$	10.7 $_6$	10.9 $_5$
AdaptWeight [59]	4.4	1.38 $_6$	1.85 $_4$	6.90 $_5$	0.71 $_3$	1.19 $_4$	6.13 $_4$	7.88 $_5$	13.3 $_5$	18.6 $_6$	3.97 $_5$	9.79 $_4$	8.26 $_2$
SemiGlob [26]	5.8	3.26 $_{10}$	3.96 $_9$	12.8 $_{13}$	1.00 $_5$	1.57 $_5$	11.3 $_{10}$	6.02 $_3$	12.2 $_4$	16.3 $_3$	3.06 $_2$	9.75 $_3$	8.90 $_3$

Table 2.2: Comparison of results on the middlebury data set. The numbers are the percentage of pixels with misestimated disparities on the different subsets of the images. The subscript of each number is the rank of that score. Our algorithm has rank 1 for most categories and rank 2 at worst. This gives an average rank of 1.3.

We evaluate our algorithm on the middlebury data set and we show in Table 2.2 that our algorithm outperforms all the other algorithms listed on the middlebury homepage. The result on each data set is computed by measuring the percentage of pixels with an incorrect disparity estimate. This measure is computed for three subsets of the image:

1. The subset of non-occluded pixels, denoted "nonoccl".

2. The subset of pixels near the occluded regions, denoted "disc".

3. The subset of pixels being either non-occluded or half-occluded, denoted "all".

For the first two categories our algorithm takes the first place for all four test sets. For the third category we take first or second place for all test sets. By consistently performing first or second on all test subsets our average rank is 1.3.

In Figure 2.6 the results after different intermediate stages are shown. This provides a visual explanation of how the different stages in the pipeline improves the results. For comparison we also give the ground truth. The scores for the intermediate results are given in Figure 2.3 along with $D_L^{(5)}$ SPECIAL, which is the same as $D_L^{(5)}$ except that we do not use the colors of the reference image to define the smoothness cost, which has a strong impact on the Teddy and Cones data sets.

In Figure 2.4 and Figure 2.5, it is shown how an increased number of iterations in estimating E_D improves the result. Zero iterations in Figure 2.4 means that we use $D_L^{(0)}$, the initial disparity

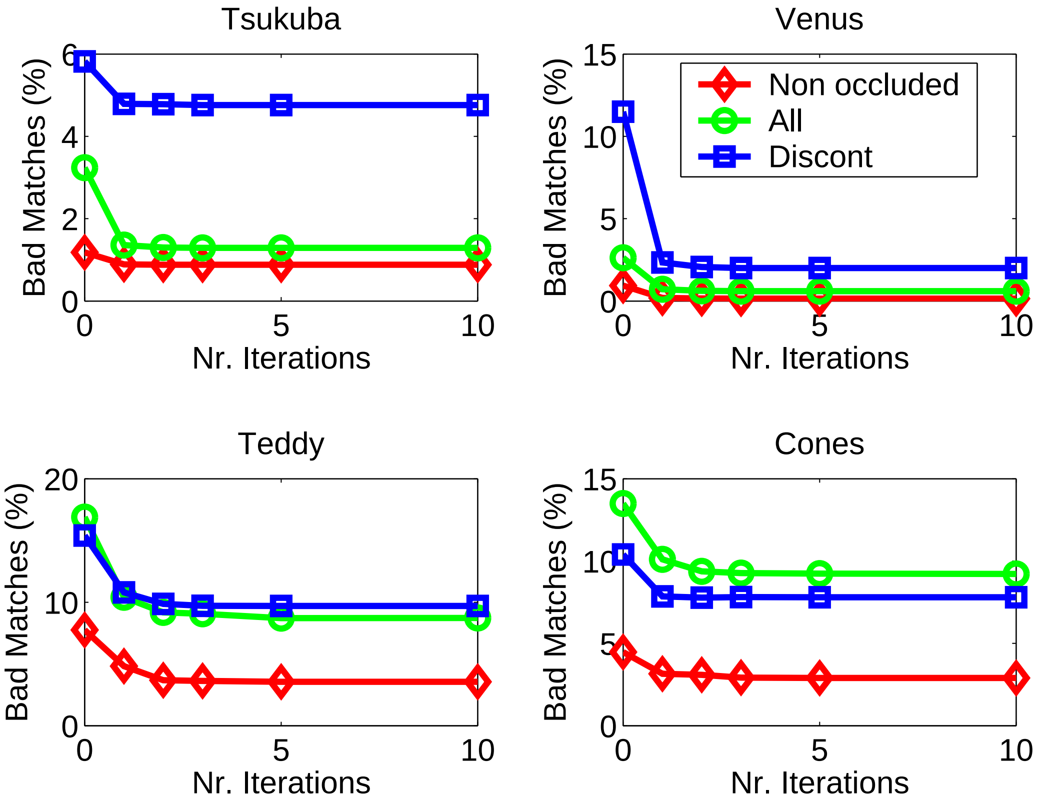


Figure 2.4: The iterated computations of $E_D^{(i+1)}$ improves the result. In most cases one iteration is enough for convergence. After five iterations, the result has always converged.

map. Based on this we chose to use five iterations in our method.

2.5 Conclusions

In this chapter, a stereo model based on energy minimization, color segmentation, plane fitting, and repeated application of hierarchical belief propagation was proposed. Typically, one application of the hierarchical belief propagation brings the error down close to its final value, so that the algorithm could perhaps be used as a two step approach, where occlusions and untextured areas are first detected and then filled in from neighboring areas.

The parameters provided constitute a good setting for the algorithm, but are not entirely optimized. More studies are needed to fully understand the behavior of our algorithm. Our algorithm is outperforming all other algorithms on the Middlebury data set, but there is space left for improvement. For instance, in our algorithm, we only refined the disparity map for the reference image, but [47] suggests that by generating a good disparity map for the right image, the occlusion constraints can be extracted more accurately. Another question that was left for further study is how to use

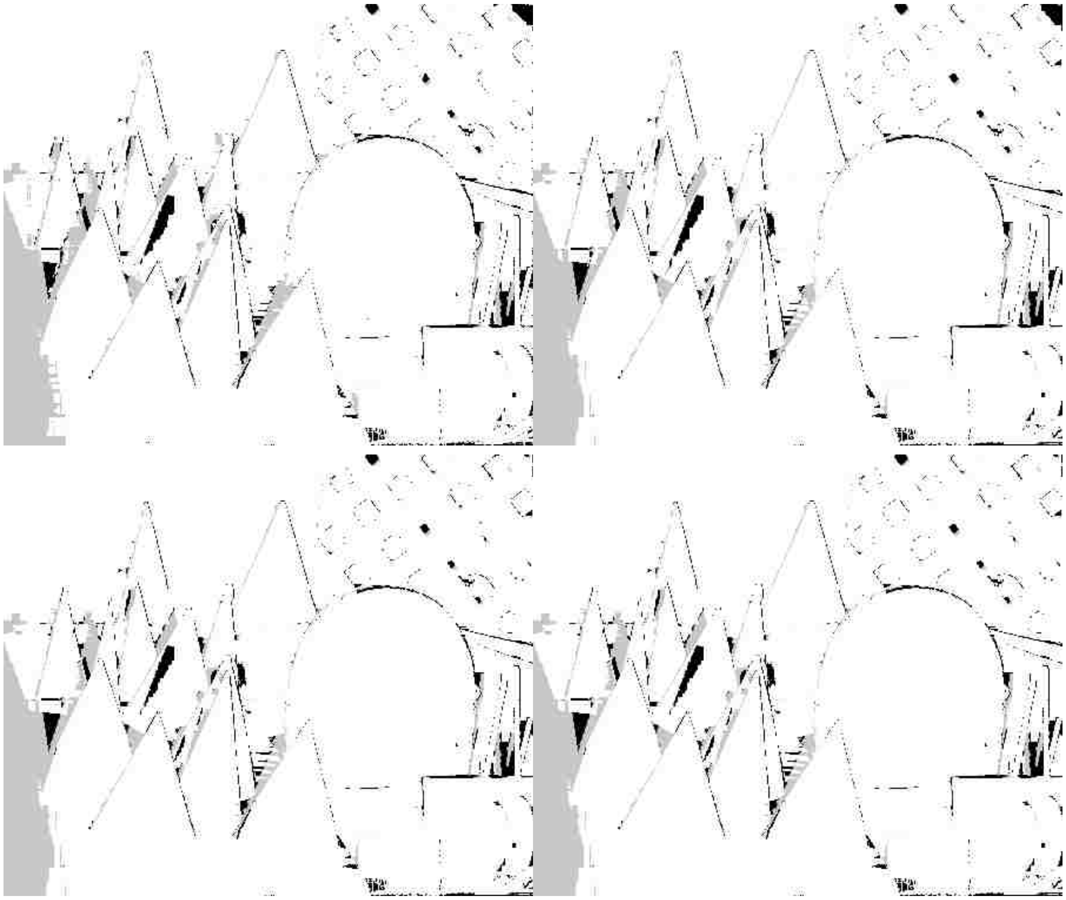
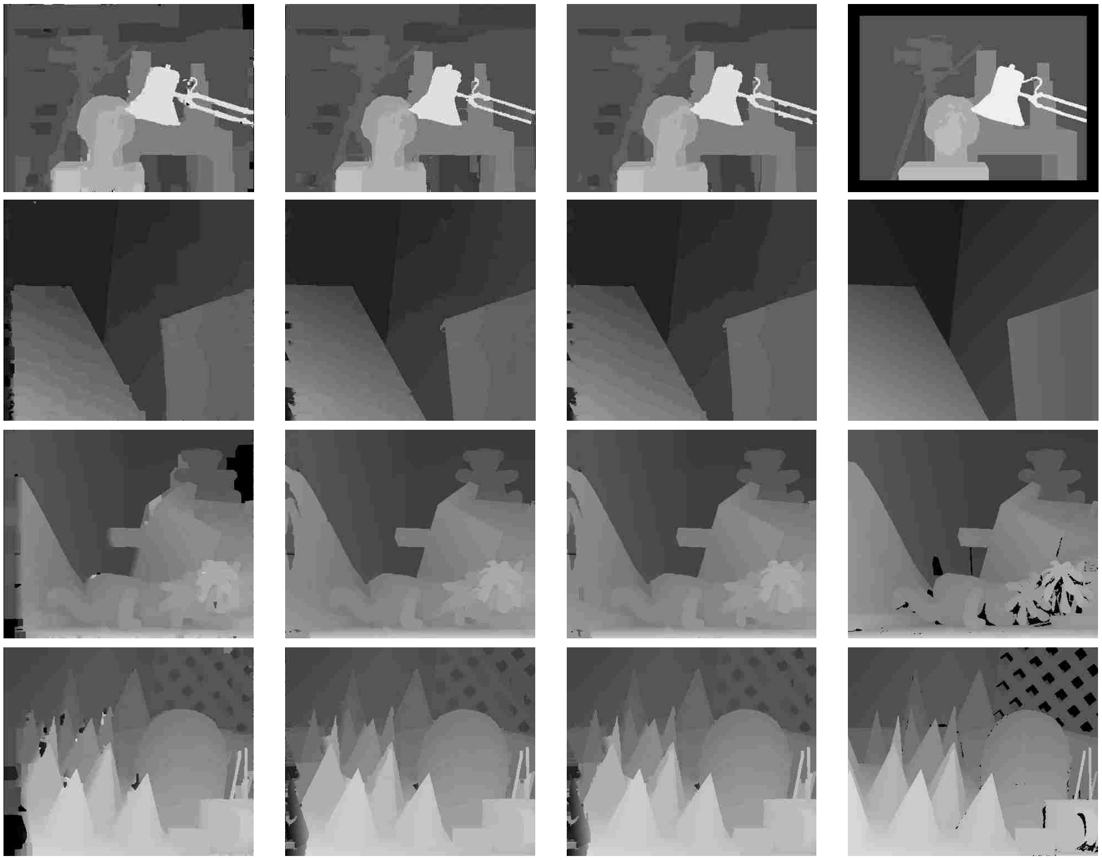


Figure 2.5: Pixels with incorrect disparity for the "Cones" data set. On the first row the results after 1 and 2 iterations are shown and on the second row the results after 3 and 5 iterations are shown.

the algorithm in a multi-view setting.



Initial Disp. ($D_L^{(0)}$) Plane Fitting ($D_{pf}^{(4)}$) Final Disp. ($D_L^{(5)}$) Ground Truth

Figure 2.6: Intermediate results from our algorithm for the four different test sets compared to the ground truth. In the first column the output of the initial BP is shown. This result is denoted $D_L^{(0)}$ in Figure 2.2 and Figure 2.3. In the second column the results after fitting planes to the regions from the color segmentation are shown. These are denoted by $D_{pf}^{(4)}$ in Figure 2.3. In the third column the final result of our algorithm is shown. These results are denoted by $D_L^{(5)}$ in Figure 2.3.

	Avg.	Tsukuba			Venus			Teddy			Cones		
Algorithm	Rank	nonocc	all	disc	nonocc	all	disc	nonocc	all	disc	nonocc	all	disc
$D_L^{(0)}$	5.7	1.18 2	3.24 8	5.82 2	0.94 4	2.63 10	11.5 10	7.75 4	16.9 7	15.4 2	4.47 6	13.5 9	10.4 4
$D_{pf}^{(4)}$	3.2	2.60 9	2.98 8	7.31 6	**0.13** 1	0.46 2	**1.79** 1	**3.93** 1	8.92 2	**9.97** 1	3.50 2	9.41 2	9.07 3
$D_L^{(5)}$	**1.3**	**0.88** 1	**1.29** 1	**4.76** 1	**0.14** 1	0.60 2	**2.00** 1	**3.55** 1	8.71 2	**9.70** 1	**2.90** 1	9.24 2	**7.80** 1
$D_L^{(5)}$ SPECIAL	**1.3**	**0.88** 1	**1.30** 1	**4.77** 1	**0.14** 1	0.60 2	**1.95** 1	**3.71** 1	9.20 2	**10.3** 1	**3.07** 2	9.33 2	**8.17** 1

Table 2.3: Intermediate results from our algorithm verified with the ground truth. The first three rows in the table corresponds to the first three rows in the above figure. The last row is the same as $D_L^{(5)}$ except that we do not use the colors of the reference image to define the smoothness cost.

Chapter 3

Two-View Stereo: Real-time Global Stereo Matching Using Hierarchical Belief Propagation

3.1 Introduction

Stereo vision has traditionally been one of the most extensively investigated topics in computer vision, and is still attracting the attention of many researchers. As a consequence, a variety of approaches have been proposed and an excellent survey of stereo algorithms can be found in [43].

In general, stereo algorithms can be categorized into two major classes: local methods and global methods. Local algorithms, which are based on correlation can have very efficient implementation that are suitable for real-time application [24, 52]. The central problem of local window-based algorithms is how to determine the size and shape of the aggregation window. That is, a window must be large enough to cover sufficient intensity variation while small enough to avoid crossing depth discontinuities for reliable estimation. This inherent ambiguity causes problems such as noisy disparities in textureless region and blurred object boundaries.

Global methods make explicit smoothness assumptions of the disparity map and minimize some cost function. A classic category of global methods is Dynamic Programming (DP) based [38, 32]. DP technique can offer optimized solution for independent scanlines in an efficient manner. Due to DP's one dimensional optimization solution and efficient performance, it is the algorithm of choice for many real-time stereo applications [54, 23, 21]. The major problem of DP is that inter-scanline consistency cannot be well enforced, leading to the well-known ”streaking” artifacts. Although new algorithms [32, 53] have been proposed to reduce the effect, it can hardly be eliminated. Recently, new global optimization methods such as Belief Propagation (BP) and Graph cut (GC) have attracted much attention. Unlike DP, these methods [48, 47, 34, 10] enforce the optimization in two dimensions, i.e. the entire image. Although some of the most impressive stereo results are obtained, both BP and GC are typically computionally expensive and therefore real-time performance has never been achieved. Recently, Felzenszwalb et al. proposed an efficient BP algorithm [18] uses a hierarchical approach for reducing the complexity. However, it still requires about one second to compute a small image (i.e. 384 × 288) and cannot achieve real-time performance yet.

In this chapter, we propose a real-time belief propagation stereo approach. This algorithm is based on a global energy-minimization framework which contains two terms, the data term and smoothness term. Thus our method can be treated as a two-step algorithm: the construction of the data term and the iterative optimization of the smoothness term. The second step is the essential part of BP, while at the same time, is commonly believed to be the bottleneck of the practical use of the algorithm. Hence, the main contributions of this chapter are: first, providing a good accelerator for all the BP based algorithms; second, providing a high quality real-time stereo matching approach.

The rest of this chapter is organized as follows. Section 2 gives a description of our stereo matching approach. In Section 3, we propose a fast-converging BP algorithm that greatly reduce the complexity when a large number of iterations are used. Section 4 details how our algorithm is implemented in GPU to gain performance improvement. Our experimental results are presented in Section 5 and in Section 6 we conclude.

3.2 Approach Description

The algorithm can be partitioned into two blocks: correlation volume computation and hierarchical BP implementation, which correspond to the two terms of the global energy: the data term E_D and the smoothness term E_S respectively.

$$E(d) = E_D(d) + E_S(d); \tag{3.1}$$

The correlation volume module constructs the data term, and the hierarchical BP module iteratively updates the smoothness term to minimize the global energy.

3.2.1 Correlation Volume Computation

In the correlation volume computation module, we compute the matching cost in a similar way as the Birchfield and Tomasi's pixel dissimilarity [8], that is for each disparity value, five matching costs are compute. In order to reduce noise, the matching cost is passing through a gaussian filter with σ one pixel. The minimum of the matching costs is selected and compared with a threshold T, multiply the smaller one with a weighting parameter η to get the data term and send it into the hierarchial BP module.

3.2.2 Hierarchical BP

The basic idea of loopy belief propagation algorithm [48] is first gathering information from a pixel's neighbors and incorporate the information to update the smoothness term between the current pixel and its neighboring pixels, and iteratively optimizing the smoothness term to achieve global energy minimization. This is different from the scanline optimization algorithms which only enforce the smoothness along each scanline, because in these algorithms, the smoothness cost information propagates only along the scanline, while in global algorithms like loopy believe propagation and graph cuts algorithms, the smoothness cost information is propagating across the whole image.

Figure 3.1 provides an example of how to update the smoothness term $E_{S,\mathbf{X}}^{(4)}$ between pixel $\mathbf{X}$ and one of its neighbors $\mathbf{Y}_4$. The first step is using Equation 3.2 to incorporate the data term of $\mathbf{X}$ ($E_{D,\mathbf{X}}$) with the smoothness term of its other neighboring pixels to generate a new jump cost $M_{S,\mathbf{X}}^{(4)}$. In this chapter, we define this cost as **multi-pass jump cost**. The new smoothness term $E_{S,\mathbf{X}}^{(4),new}(d)$ between $\mathbf{X}$ and its neighbor $\mathbf{Y}_4$ is then updated by computing the smallest jump cost

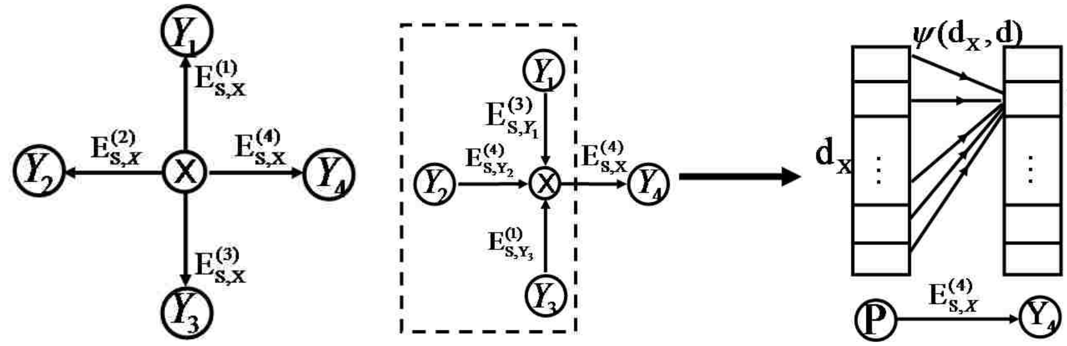


Figure 3.1: Belief propagation algorithm. In the left figure, pixel $\mathbf{X}$ has four neighbors $\mathbf{Y}_1$, $\mathbf{Y}_2$, $\mathbf{Y}_3$, $\mathbf{Y}_4$, there's a smoothness term between pixel $\mathbf{X}$ and each of its neighboring pixels. The right figure provides an example showing how to update the smoothness term $E^{(4)}_{S,\mathbf{X}}$ between pixel $\mathbf{X}$ and one of its neighbors $\mathbf{Y}_4$.

using Equation 3.3.

$$\begin{aligned} M^{(4)}_{S,\mathbf{X}}(d_{\mathbf{x}}) &= E_{D,\mathbf{X}}(d_{\mathbf{x}}) + E^{(3),old}_{S,\mathbf{Y_1}}(d_{\mathbf{x}}) + E^{(4),old}_{S,\mathbf{Y_2}}(d_{\mathbf{x}}) + E^{(1),old}_{S,\mathbf{Y_3}}(d_{\mathbf{x}}), & (3.2) \\ E^{(4),new}_{S,\mathbf{X}}(d) &= arg\min_{d_{\mathbf{x}}}(M^{(4)}_{S,\mathbf{X}}(d_{\mathbf{x}}) + \Psi(d_{\mathbf{x}}, d)), & (3.3) \end{aligned}$$

we define $\Psi(d_{\mathbf{x}}, d)$ as **single-pass jump cost** between two neighboring pixels, it is linear to the absolute difference of the disparities of pixel $\mathbf{X}$ and $\mathbf{Y}_4$. However, in order to increase the robustness to outliers, a threshold λ is added as shown in Equation 3.4.

$$\Psi(d_{\mathbf{x}}, d) = \mathbf{min}(\lambda, \rho \mid \mathbf{d_x} - \mathbf{d} \mid), \tag{3.4}$$

The smoothness term (E_S) is iteratively updated which results in the minimization of the global energy E:

$$E(d) = \sum_{\mathbf{X}} E_{\mathbf{X}}(d) = \sum_{\mathbf{X}} E_{D,\mathbf{X}}(d) + E^{(3)}_{S,\mathbf{Y_1}}(d) + E^{(4)}_{S,\mathbf{Y_2}}(d) + E^{(1)}_{S,\mathbf{Y_3}}(d) + E^{(2)}_{S,\mathbf{Y_4}}(d), \tag{3.5}$$

The global energy converges after a certain number of iterations. The disparity value on each pixel $\mathbf{X}$ is calculated as following:

$$D_{\mathbf{X}} = arg\min_{d}(E_{\mathbf{X}}(d)), \tag{3.6}$$

The general loopy belief propagation algorithm is too slow to be practically used while achieving very good result, not only because the algorithm itself is complicated, but also because a certain number of iterations are required before the algorithm converges. Felzenszwalb [18] provides a hierarchical algorithm which runs much faster than the previous algorithms while yielding comparable accuracy. The main difference between the hierarchical BP and general BP is that hierarchical BP works in a coarse-to-fine manner, first performing BP at the coarsest scale, then using the output from the coarser scale to initialize the input for the next scale.

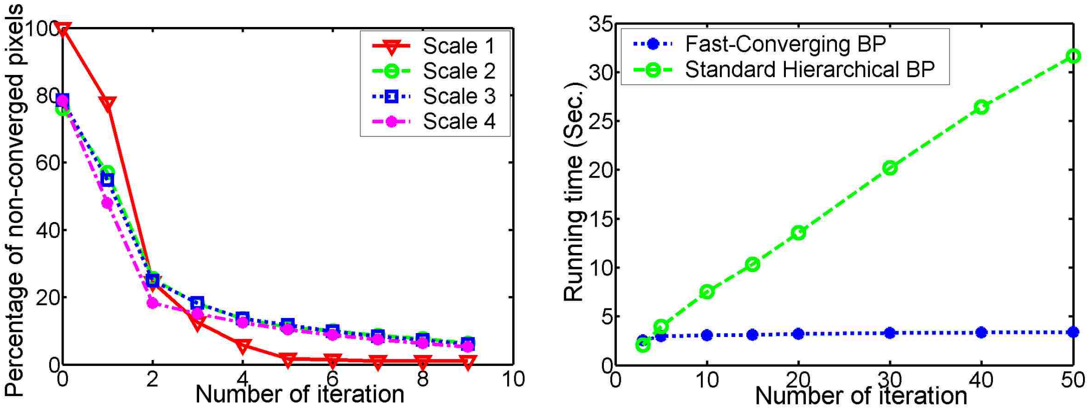


Figure 3.2: Comparison of fast-converging BP and standard BP. The left figure provides the percentage of the non-converged pixels after every iteration, and the right figure provides the comparison of the running time of fast-converging BP and Standard hierarchical BP algorithms. Both algorithms are run on Tsukuba data set with the same number of iterations on all the four scales.

3.3 Fast-Converging BP

For standard BP algorithms, in order to achieve the best stereo results, a large number of iterations are required to guarantee the convergence. However, since the running time is linear to the number of iterations, large number of iterations will greatly hurt the practical application of BP algorithms.

Actually, there are lots of redundant computations involved in standard BP. In essence, by only updating pixels that have not yet converged, fast-converging BP removes those redundant computations while achieving the same accuracy as standard BP.

In detail, the new smoothness term of one of the pixels in the graph is updated according to its own data term ($E_{D,\mathbf{X}}$), the previous smoothness term of its four neighboring pixels ($E_{S,\mathbf{Y}_i}^{(i),old}$), and the **single-pass jump cost** function Ψ. Since the data term and the **single-pass jump cost** stay unchanged, they can be treated as fixed parameters, and the updated smoothness term ($E_{S,\mathbf{X}}^{(i),new}$) of a pixel $\mathbf{X}$ in the graph thus becomes a function of variables containing only the previous smoothness term of its four neighboring pixels, for instance:

$$E_{S,\mathbf{X}}^{(4),new} = f(E_{S,\mathbf{Y}_1}^{(3),old}(d_{\mathbf{x}}), E_{S,\mathbf{Y}_2}^{(4),old}(d_{\mathbf{x}}), E_{S,\mathbf{Y}_3}^{(1),old}(d_{\mathbf{x}})), \tag{3.7}$$

As a result, before updating the smoothness term of a pixel $\mathbf{X}$ at iteration i, check whether the smoothness term of its four neighboring pixels at iteration $i-1$ and at iteration $i-2$ are equivalent or not. If the smoothness terms are the same, it is not necessary to update the smoothness term of pixel $\mathbf{X}$.

Figure 3.2 shows that after several numbers of iterations, most of the pixels on the graph converge. The fast-converging BP algorithm thus ignores these pixels, the updating scheme is only applied to the non-converged pixels, which greatly decrease the running time of BP approaches with large number of iterations. Figure 3.2 also shows that the running time of the standard BP is linear to the number of iterations while the running time of the fast-converging BP dramatically converges.

We have successfully implemented this fast-converging BP approach on CPU, and are looking forward to implementing it on GPU. The experiment results provided in Figure 3.2 are based on CPU implementation.

3.4 GPU Implementation

We have implemented both the first (correlation volume computation) and second step (hierarchical BP) on graphics hardware to facilitate real-time computation. The GPU implementation of the first step is very simple, so we only focus on the second step.

In our implementation, there are four scales in the hierarchical BP, and the main process for each scale are the same. The updating scheme for each scale are summarized in Algorithm 3.4. For each scale, eight textures are used to store the smoothness term. The old smoothness term generated in the previous iteration is stored in four of the textures ($E_{S,old}^{(i)}$, $i = 1, 2, 3, 4$), the other four textures are used to store the updated smoothness term ($E_{S,new}^{(i)}$). For the coarsest scale, before the iteration begins, initialize $E_{S,old}^{(i)}$ with all zeros, as to the other scales, $E_{S,old}^{(i)}$ is initialized as Algorithm 3.4 describes. At the beginning of each iteration, compute **multi-pass jump cost** $M_S^{(i)}$ from the data term and the previous smoothness term as described in the Approach Section. The next step is to update the smoothness term using Equation 3.3. The complexity of this problem is $O(NR_{disp}^2)$ (NR_{disp} is the number of disparity levels), but the updating scheme provided in Algorithm 3.4 reduces the complexity to $O(N)$. Finally, normalize $E_{S,new}^{(i)}$, such that $\sum_{d=0}^{N-1} E_{S,new}^{(i)}(d) = 0$.

When the iteration is completed in the fine scale, uses Equation 3.5 and 3.6 to create the disparity map.

3.5 Experimental Results

We tested our **real-time BP** algorithm on a 3 GHz PC running Direct3D 9.0. The GPU is a Geforce 7900 GTX graphics card with 512M video memory from NVIDIA. All shaders are implemented using HLSL and complied using pixel shader 3.0 profile. The following experiments are conducted to evaluate both the quality and efficiency performance of our algorithm. The same parameter settings were used throughout the experiments. Two parameters $T = 30$ and $\eta = 0.15$ are used in the correlation volume computation module, another two parameters are involved in the calculation of the **single-pass jump cost**: ρ and λ. ρ is set to 1.0, and λ is determined by the the number of disparity levels (NR_{disp}) of the input data set: $\lambda = (2.0 \times NR_{disp})/16$. 16 is the number of disparity levels of the Tsukuba data set. In this chapter, we implement hierarchical BP in GPU with four scales, and the typical iterations for each scale are $(5, 5, 10, 4)$, from coarse-to-fine scale.

Algorithm 1 Updating the Smoothness Term on GPU

```
Require: $E_{S,old}^{(i),coarse}$, $i = 1, 2, 3, 4$.
 1. Initialize $E_{S,old}^{(i),fine}$: $E_{S,old}^{(i),fine}(X) = E_{S,old}^{(i),coarse}(X/2)$;
 2. Initialize $N$ with the number of iterations of the current scale.
 3. repeat
 4.     -Compute $M_S^{(i)}$ according to Equation 3.2;
 5.     -Compute the minimun of $M_S^{(i)}$ for each pixels, plus it with the threshold $\lambda$
         which is provided in Equation 3.4, save it as $MIN_S$;
 6.     -Update $E_{S,new}^{(i)}$:
           for $d$ from 1 to $NR_{disp} - 1$:
               $M_S^{(i)}(d)$=min($M_S^{(i)}(d), M_S^{(i)}(d-1) + \rho$);
           for $d$ from $NR_{disp} - 2$ to 0:
               $M_S^{(i)}(d)$=min($M_S^{(i)}(d), M_S^{(i)}(d+1) + \rho, MIN_S$);
           $E_{S,new}^{(i)} = M_S^{(i)}$;
 7.     -$E_{S,old}^{(i)} = E_{S,new}^{(i)}$;
 8.     -Normalize $E_{S,new}^{(i)}$, such that $\sum_{d=0}^{N-1} E_{S,new}^{(i)}(d) = 0$
 9.     -Decrease $N$ by 1.
10. until $N \leq 0$.
```

3.5.1 Quality Evaluation with Ground Truth

We first evaluate the reconstruction quality of our approach using the benchmark middlebury stereo data set based on known ground truth. The new evaluation test data consists of four stereo pairs within which ”Tsukuba” and ”Venus” are standard stereo data with slanted surfaces and up to 20 disparity levels, ”Teddy” and ”Cones” are both new adopted image pairs with more complicated scene structure and much larger disparity ranges. We evaluate the numerical accuracy of the dense disparity maps generated by our algorithm using the online system at [42]. The results from all test images are shown in figure 3.3.

This quantitative evaluation confirms that, as demonstrated in Table 3.1, our **real-time BP** performs as well as other global optimization approaches. Generally speaking, the overall performance is ranked between the best belief propagation based algorithms which are the current

Algorithm	Avg. Rank	Tsukuba nonocc	 all	Venus nonocc	 all	Teddy nonocc	 all	Cones nonocc	 alk
DoubleBP [56]	2.3	0.88$_{1}$	1.29$_{1}$	0.14$_{2}$	0.60$_{5}$	3.55$_{1}$	8.71$_{3}$	2.90$_{3}$	9.24$_{4}$
SymBP+occ [47]	5.1	0.97$_{2}$	1.75$_{5}$	0.16$_{3}$	0.33$_{2}$	6.47$_{6}$	10.7$_{4}$	4.79$_{11}$	10.7$_{8}$
Our Algorithm	**10.4**	**1.49**$_{10}$	**3.40**$_{13}$	**0.77**$_{7}$	**1.90**$_{11}$	**8.72**$_{13}$	**13.2**$_{8}$	**4.61**$_{9}$	**11.6**$_{10}$
GC+occ [34]	11.5	1.19$_{4}$	2.01$_{9}$	1.64$_{14}$	2.19$_{13}$	11.2$_{16}$	17.4$_{16}$	5.36$_{13}$	12.4$_{13}$
MultiCamGC [35]	12.0	1.27$_{5}$	1.99$_{8}$	2.79$_{19}$	3.13$_{17}$	12.0$_{17}$	17.6$_{17}$	4.89$_{12}$	11.8$_{11}$
GC [43]	16.6	1.94$_{12}$	4.12$_{15}$	1.79$_{16}$	3.44$_{18}$	16.5$_{21}$	25.0$_{22}$	7.70$_{17}$	18.2$_{18}$
Belief prop. [48]	NA	1.15	NA	0.98	NA	NA	NA	NA	NA
HierarchicalBP [18]	NA	1.86	NA	0.96	NA	NA	NA	NA	NA

Table 3.1: Performance comparison of the proposed method with other high-quality global optimization approaches. This measure is computed for three subsets of the images,they are "nonocc": the subset of non-occluded pixels, "all": pixels that are either non-occluded or half-occluded. The subscript is the relevant rank of each item on the table. Note that since the old middlebury table which contains several bp based stereo methods is no longer functional, we have collected the non-occluded (overall) error rate of the shared test data 'Tsukuba' and 'Venus'. Those numbers that are not available due to this reason are labeled "NA".

state-of-the-art stereo algorithms and the Graph Cuts based algorithms. One thing worth noticing is that most of these methods, such as [56, 47, 34], integrate multiple low-level visual cues (e.g., segmentation, edges, visibility testing) as either soft or hard constraints to improve stereo matching while our approach works under a basic and clean probabilistic framework without any additional information incorporated. Moreover, the iteration numbers used across all experiments is only 4. Although increasing the number of iterations can produce stronger results simultaneously, we balance the quality and efficiency by not using too may iterations and our later experiments will show this compensation does not prevent us from achieving satisfying results.

In addition, in terms of accuracy, Table 3.2 shows that our **real-time BP** outperforms all the other methods that can achieve real-time or near real-time performance listed on the new middlebury evaluation table. Since the old evaluation table at middlebury, which contains some algorithms that aim real-time, is no longer functional, we collecte the non-occluded error percentage of the shared test data 'Tsukuba' and 'Venus' and provide results in Table 2 for reference.

Algorithm	Avg. Rank	Tsukuba nonocc	Venus nonocc	Teddy nonocc	Cones nonocc
Our Algorithm	**10.4**	**1.49** $_{10}$	**0.77** $_{7}$	**8.72** $_{13}$	**4.61** $_{9}$
RealTimeGPU [54]	14.3	2.05 $_{14}$	1.92 $_{17}$	7.23 $_{8}$	6.41 $_{15}$
ReliabilityDP [23]	15.6	1.36 $_{7}$	2.35 $_{18}$	9.82 $_{15}$	12.9 $_{22}$
Realtime [25]	NA	4.25 $_{\text{NA}}$	1.32 $_{\text{NA}}$	NA	NA
Realtime DP [21]	NA	2.85 $_{\text{NA}}$	6.25 $_{\text{NA}}$	NA	NA
Max. surf. [46]	NA	11.10$_{\text{NA}}$	5.51 $_{\text{NA}}$	NA	NA

Table 3.2: Performance comparison of the proposed method with other real-time approaches listed on the Middlebury evaluation tables.

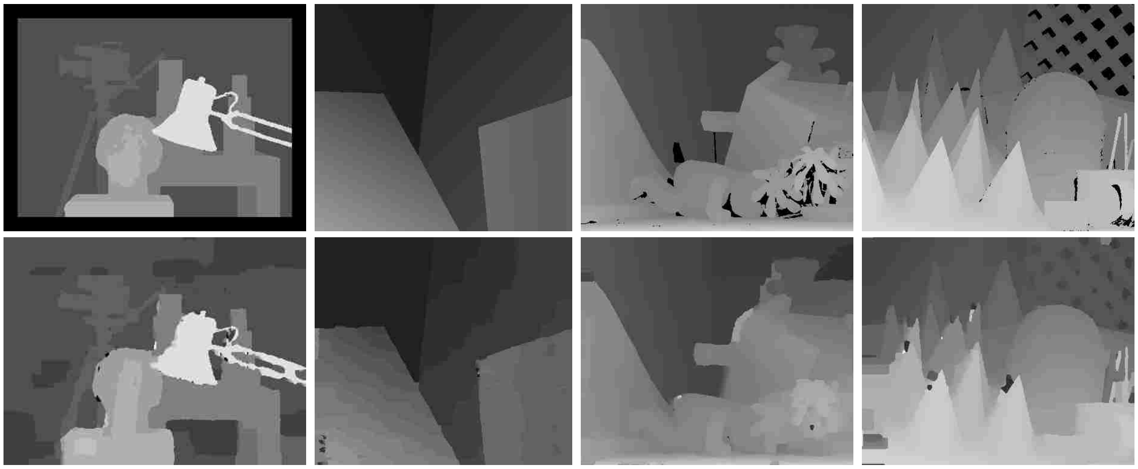


Figure 3.3: Resulting disparity maps from the middlebury stereo data set. (top row) Ground truth; (bottom row) Disparity maps generated from our method.

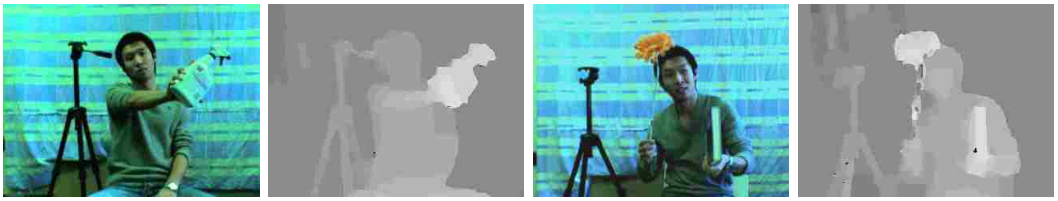


Figure 3.4: Two sample images and their corresponding disparity maps from our live system on a 3 GHz PC with a Geforce 7900 GTX graphics card from NVIDIA. Our system can reach 16 fps given 320 × 240 input images and 16 disparity levels.

3.5.2 Live System and Efficiency Performance

We integrated our algorithm into a real-time stereo system with live video input. The stereo pairs are rectified and with lens distortion removed. This pre-processing is implemented in the GPU using texture-mapping functions. Figure 3.4 shows the results of applying our **real-time BP** algorithm to some live images captured from our system. These real scene images are with resolution 320 × 240 and 16 disparity levels. Note that the scene structures and object borders have been well detected. The speed is about 16 fps for our live system.

To further evaluate the efficiency performance of our algorithm, we test our system against the four middlebury test data under different configurations and summarize the results in Table 3.3. Two characteristics of our **real-time BP** algorithm can be observed from the measurements. First, by utilizing graphics hardware acceleration, we can achieve a speedup factor up to 45 compared to its CPU counterpart. Second, the error percentage changed slightly with the increasing of iterations. Using a few iterations are able to produce strong results. These two characteristics cooperatively explain why our algorithm is very suitable for real-time application.

Iteration	MDE/Second		Error(%)				Avg.
(N)	CPU	GPU	Tsukuba	Venus	Teddy	Cones	Rank
2	0.49	22.2	1.59	0.90	8.89	4.73	11.0
4	0.39	17.0	1.49	0.77	8.72	4.61	10.4
6	0.31	13.7	1.47	0.67	8.68	4.57	9.8
10	0.23	10.1	1.47	0.60	9.09	4.54	9.7

Table 3.3: Running time evaluation on the four new Middlebury stereo data. The speed and overall error rate corresponding to different number of iterations are presented in the table. Speed performance is measured by million disparity estimations per second (MDE/s). Here both the CPU and GPU's MDE/s values are calculated based on Tsukuba data set. Clearly GPU acceleration can achieve a high speedup factor compared to the CPU implementation. In addition, iterations used for each scale are $(5, 5, 10, N)$, from coarse-to-fine scale. N is the variable provided in the table.

3.6 Discussion

In this chapter, a real-time stereo model based on hierarchical belief propagation was proposed, which demonstrates that global optimization based stereo matching is possible for real-time applications. The whole algorithm design in this chapter is very clean and results in very high quality stereo matching. We qualified the accuracy of the stereo results using the Middlebury benchmark, which shows that our algorithm outperforms all the other real-time stereo algorithms.

Looking into the future, both the quality and the speed of the proposed real-time BP approach can be improved. For the quality, more constraints and priors (e.g. edges, corners, junctions, segmentation, visibility) can be incorporated; for the speed, in Section 3.3, we have proposed an approach which allows large number of iterations to guarantee the convergence, for instance, the running time of fast-converging BP with 100 iterations is even less than the running time for standard BP with 5 iterations. We're planning to transfer it to GPU in the near future. In addition, [18] presented an approach which can decrease both the storage requirements and the running time by a factor of two. Because for a bipartite graph, all the nodes can be separated into two clusters, for each iteration, only one cluster's smoothness term needs to be updated. We have implemented this approach on CPU, and we would also like to implement this approach on GPU to improve the speed up to a factor of two in the near future.

Chapter 4

Multi-view Stereo: Near Real-time Robust Plane-fitting Stereo for Weakly-Textured Urban Imagery

4.1 Introduction

This chapter presents a robust method for correcting textureless areas in stereo depth maps using locally estimated planes. The approach is especially relevant to 3D urban reconstruction, for which many areas in an image may contain planar objects. We designed our system with an emphasis on performance in order to facilitate the computation of large reconstructions.

Urban mapping and reconstruction are topics extensively researched in many areas. Photogrammetrists have long been able to map urban terrain and buildings using aerial and satellite imagery, which is now widely available via commercial applications, such as Microsoft Virtual Earth and Google Earth. Many groups have successfully created ground-based reconstructions using laser range finders and cameras, see [22]. Within the scope of vision-only approaches, semi-manual reconstructions of facades can be created with a small number of images [16], while automatic reconstruction has been accomplished to a degree using longer video sequences [4].

When creating reconstructions from video, small subsequences can be reconstructed via stereo depth estimates, which are later combined into a larger reconstruction. While accurate depth estimates are important for generating usable reconstructions, the best current algorithms run offline. Since the number of depth maps necessary for reconstructing an urban area is typically high, allowing individual estimates to run until some optimal convergence or otherwise execute for an arbitrarily long time may not be feasible, so offline algorithms are not typically appropriate here. Several online approaches also exist, often executing on graphics hardware or on low resolution images. However, such approaches often fail in large textureless or weakly-textured regions, since stereo correspondence is uninformative. Unfortunately, urban areas tend to have many such regions. In this chapter, we propose a near real-time plane-fitting stereo pipeline and a BP-based plane-fitting stereo pipeline to deal with this problem.

The plane-fitting stereo pipeline contains three modules: window-based multi-view stereo matching, stereo fusion and plane-fitting refinement. To achieve high speed performance, a novel real-time color segmentation approach is proposed in the last module. The BP-based plane-fitting stereo pipeline contains three modules too, and the last two modules are the same as the plane-fitting stereo pipeline. The first module is different in that after window-based stereo matching, a plane-fitted depth map is created followed by a loopy belief propagation refinement. This helps correcting potential errors caused by plane-fitting due to non-robustness of color segmentation.

We discuss previous work in Section 4.2. Section 4.3 provides an overview of the algorithm that provides a basis for our approach. Sections 4.4 and 4.5 detail our improvements via plane fitting and belief propagation, respectively. Results are shown in Section 4.6, and Section 4.7 concludes.

4.2 Related Work

A survey of several classes of stereo algorithms is given in [43]. As stated in that paper, local stereo algorithms are dependent on their aggregation windows. If a local algorithm encounters a textureless area larger than the aggregation window, i.e. the depth estimate for a given pixel has no unique support within a local region, the algorithm is guaranteed to fail. Moreover, most real time algorithms are local, meaning that in weakly-textured environments, such algorithms will produce large amounts of error. Global algorithms, such as graph cuts [10] and belief propagation [33, 47, 48, 56, 57], have properties that can improve depth estimates in difficult environments. These algorithms rely on minimization of some global cost function. In textureless areas, the minimization tends to have a smoothing or blurring effect. Because of their iterative nature, they are typically too slow for limited time frame applications. Yang *etal* [57] uses a few iterations of hierarchical belief propagation in a real-time stereo implementation to smooth out the largest inconsistencies. We use a similar algorithm as an optional step in our approach. However, this method is still not sufficient for covering larger textureless regions.

Color segmentation provides a useful clue for related depth regions, as several authors [9, 33, 47, 50, 56] have noted. Yang *etal* [56] classify all the pixels in the reference image into three class: stable, unstable, and occluded pixels. For each color segment, only the stable pixels will be chosen to fit a 3D plane.

Instead of attempting to infer depth purely from stereo matching, several methods exploit the planarity implicit to urban environments. Baillard and Zisserman [5] use a 3D line and surrounding texture to hypothesize planes in an image. Similarly, Werner and Zisserman [55] automatically search for scene planes in a set of images using point and line correspondences. Since the authors focus on architectural scenes, they assume most of the reconstruction will be limited to a few dominant planes, and compensate for deviations from this assumption as a secondary step. Cornelis *etal* [15] describe a real-time method for creating simplified urban models that assume all surfaces are either on a ground plane or a plane orthogonal to it. In contrast to these methods, we place no initial assumption of planarity on the scene and use the planes as an error compensation method for depths we can not otherwise determine.

4.3 Window-based Stereo

Given images attached with camera poses the first step in a vision-based reconstruction pipeline is to calculate depth maps. Within the confines of real-time/near real-time vision-based large scale 3d reconstruction, current state-of-the-art is to use a multi-view stereo matcher that falls within the category of local methods, i.e. the depth selection for each pixel is only based on local data. A next logical step given a sequence of such consecutive depth maps is to enforce consistency among the depth maps using the large overlap existing in their viewpoints, hence being able to improve on their

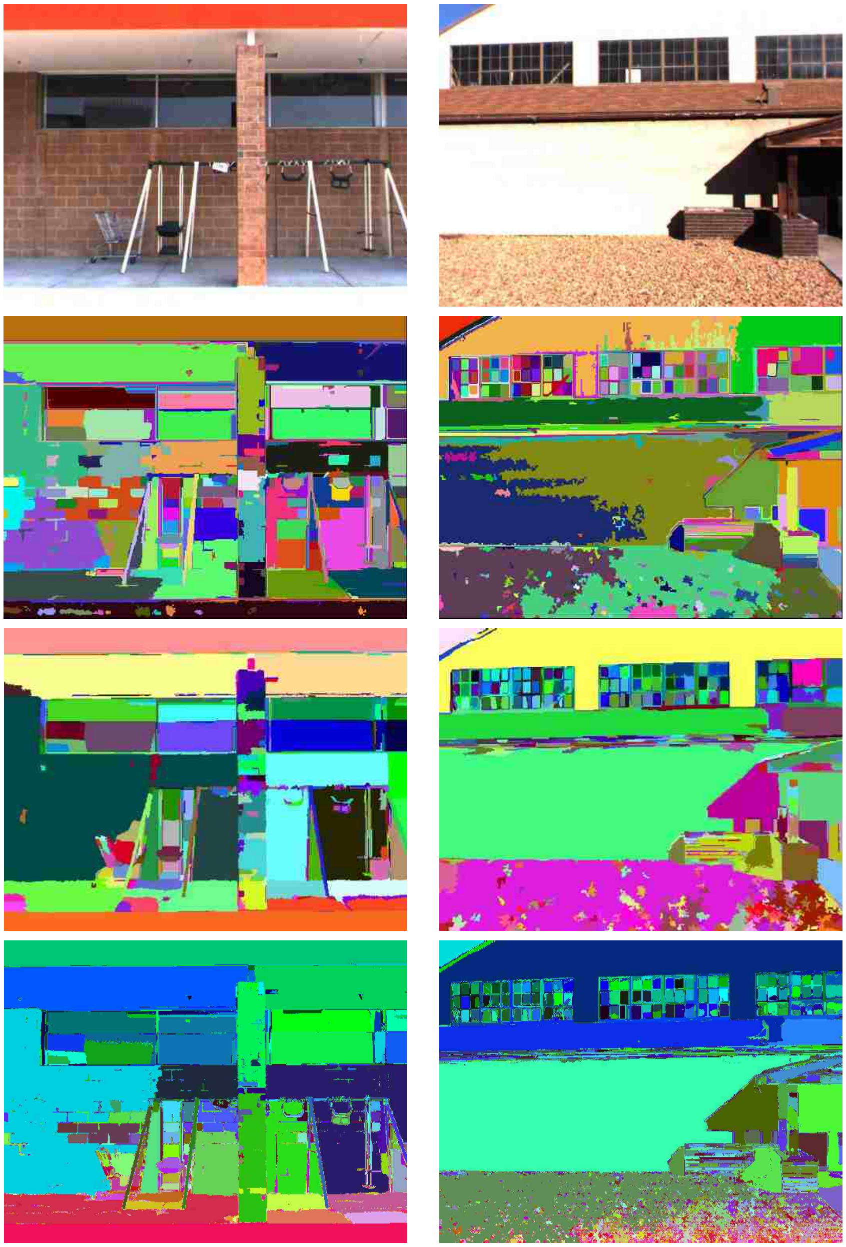


Figure 4.1: Visual comparison of the segmentation results using different color segmentation approaches. From top to bottom: Color images, Graph-Based approach [19], EDISON system [2] and our approach. Notice that our real-time approach produces similar results as the EDISON system, which takes around 15 seconds per image.

overall quality by removing outliers and noise. Such a system is described in [4], which we briefly review here.

The primary part of a multi-view stereo matching module is the plane-sweeping algorithm of [13, 58]. Given a sequence of consecutive images, say 7 images, the depth map is intended to be computed for the central image, denoted the reference image. A set of planes are swept through space at a number of hypothesized depths. Each plane defines a set of homographies with which all the non-reference images are warped onto the reference image. The absolute intensity difference is computed as the cost of the hypothesized depth. The set of images is divided in two halves, one preceding and one following the reference image. The costs are aggregated by a boxcar filter and the minimum of the two sums is the cost of the depth hypothesis [31] (this is an effective way of handling occlusions). By assuming that the cost function can be approximated by a quadratic function around the minimum, sub-pixel accuracy can be achieved by interpolation. The hypothesis having the lowest cost may not always be the true depth. This is in most cases due to lack of texture. Therefore a confidence map is needed to denote how certain we are about each chosen depth hypothesis.

The next step given the sequence of consecutive depth maps is to enforce consistency among these and output a new improved set of depth maps. This consistency fusion of the depth maps is very effective since the same surface points are visible in a large number of frames, which means that we have multiple depth estimates for each point. Hence using the most consistent estimate among these enables a large improvement removing outliers and noise. The module that performs these steps is named "Stereo fusion". However even after this step there will still be pixels for which the depth estimate is unlikely/wrong, so each new depth map is again associated with a confidence map.

4.4 Plane-fitting Stereo

The Plane-fitting stereo pipeline contains three modules, window-based stereo matching, stereo consistency fusion and plane-fitting refinement. The first two modules are the same as the window-based stereo pipeline described in Section 4.3. For the last added module a novel real-time color segmentation approach is used where within each large segment, a plane is fitted in order to obtain correct depth values for the weakly-textured regions.

4.4.1 Real-time Color-weighted Color Segmentation

Today, color segmentation is becoming more and more important in stereo computation. For instance, the top six stereo algorithms [9, 27, 33, 36, 47, 56] reported to the Middlebury benchmark [42] use color segmentation as a step in their computations. However color segmentation is known to be non-robust and most of the state-of-the-art segmentation algorithms that perform reasonably well are very slow. Hence, the color segmentation becomes the bottleneck of the speed performance of most of the state-of-the-art stereo algorithms.

There exist lots of different color segmentation algorithms, such as watershed segmentation [44], graph-based segmentation [19], and the mean shift segmentation [14]. One of the current state-of-the-art is the Edge Detection and Image Segmentation (EDISON) System [2] which integrates confidence based edge detection [37] and mean shift based image segmentation [14]. Our goal is to achieve similar segmentation quality as the EDISON system while running at real-time. We separate our approach into two steps: Image smoothing and region linking.

To well preserve the edges, we use a color-weighted filter [56, 59] to smoothen the image. The support from the neighboring pixel q to the pixel under consideration p is weighted as

$$w(p,q) = exp(-(\frac{\triangle c_{pq}}{\gamma_c} + \frac{\triangle s_{pq}}{\gamma_s})), \tag{4.1}$$

where $\triangle c_{pq}$ is the maximal color difference between p and q measured in each channel of the CIELUV color space and $\triangle s_{pq}$ is the distance between p and q in the image domain. γ_c and γ_s are the parameters that control weights. To achieve real-time performance, the color space transformation and the smoothing steps are done in GPU (Graphics Processing Unit), and the range of the three channels of CIELUV color space is confined in $[0, 255]$. In our implementation, the parameters are set experimentally to $\gamma_c = 2.0$ and $\gamma_s = 10$, and the filter is iteratively applied to the color image five times. The speed performance of the smoothing step is about 15 frame per second with image resolution 512×384 using NVIDIA GeForce 8800 GTX graphics card.

The second step is to link all the 8-connected pixels based on the color similarity. Assume the absolute color differences of two 8-connected pixels p, q of the smoothed image is $\{dL,\ dU,\ dV\}$, then if

$$\max(dL, dU, dV) < \gamma_c, \tag{4.2}$$

p, q will be joined together. The speed performance of the second step is about 33 frame per second. We adopted a parallellized approach by using the GPU to smoothen the color image and the CPU to perform region linking. This approach results in a speed performance of about 15 frame per second.

Figure 4.1 provides the visual comparison of our segmentation approach with two other approaches. The comparison shows that the graph-based approach is more vulnerable in preserving details. For instance, the ground on the left side of the column in the first dataset is joined with the wall if the graph-based approach is performed. However, both the EDISON system and the proposed color-weighted segmentation approach separated them clearly. And also, comparing with the proposed approach, the graph-based approach is more weakly in handling noises. The white ground in the first dataset and the white wall in the second dataset are grouped as single region using both the EDISON system and the proposed approach, however, the graph-based approach fails. By visual comparison, the main difference of the proposed color-weighted approach and EDISON system is that ours contains many very small segments. The reason is that we don't threshold on minimum segment size. Instead we later in our pipeline never perform plane-fitting for segments which are too small.

In general, comparing to the other approaches, our color segmentation approach works very well in terms of quality when run on our large experimental urban dataset. At the same time on image resolutions of 512×384, our approach runs at real-time while the graph-based approach needs about 0.5 seconds and the EDISON system takes about 15 seconds using default parameters.

4.4.2 Plane-fitting

The goal of plane-fitting is to correct depth values that are believed to be incorrect, for example depth estimates computed in weakly textured image regions. Such bad estimates are detected by defining the following confidence map C_c,

$$C_c(p) = C_f(p)h(\sum_{i=1}^{N} h(|D'_i(p) - D_{ref}(p)|, \sigma_c), \eta_c) \tag{4.3}$$

where

$$h(a,b) = \begin{cases} 1 & \text{if } a >= b \\ 0 & \text{else} \end{cases}$$

and C_f is the confidence map after the stereo fusion module. D'_i is one of the N neighbouring depth maps projected onto the reference depth map, D_{ref}. $\sigma_c = 0.2$ and $\eta_c = N - 1$ are thresholds for the consistency check.

All the pixels in the reference depth map are classified into stable and unstable pixels by setting a threshold for the confidence map, C_c. If $C_c(p)$ is higher than a threshold, p is classified as a stable pixel. For each selected segment S_j in the image of the reference depth map, a 3D plane is fitted robustly using RANSAC [20] on the depth values of the stable pixels only. All the stable pixels $p_k \in S_j$ are back projected as 3D points P_k. A set of hypothesis planes are generated by randomly selecting three points and computing the plane that intersects these. The vector defining the plane is then normalized and each plane is associated with the following error cost

$$E_\pi(S_j, \pi_j) = \sum_{p_k \in S_j} \min(P_k^T \pi_j, \eta_d), \tag{4.4}$$

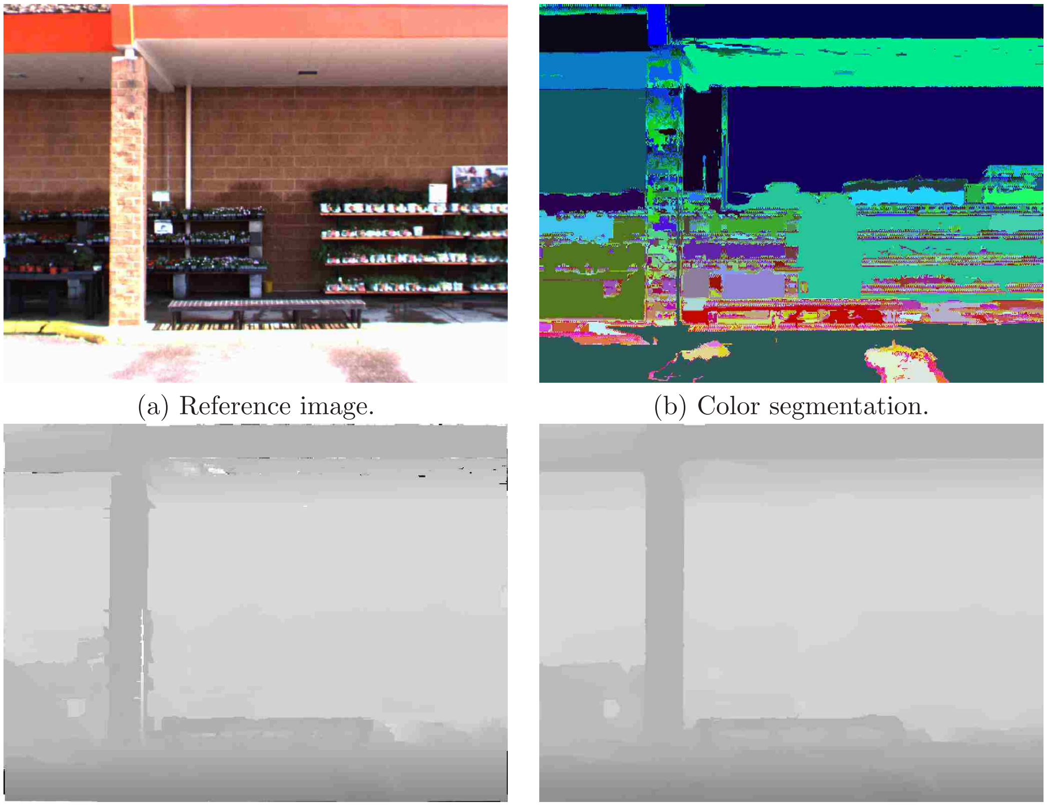
where η_d is a constant to increase robustness by rejecting potential outliers. Finally, the plane hypothesis with the minimum cost is selected and the depth values of *only the unstable pixels* are replaced with the plane-fitted depth values. What should be mentioned is that only large segments will be fitted with a plane, since a small segment suggests variations in the segments neighborhood, i.e high texture.

The plane-fitting approach may fail if the number of stable pixels in the segment is too small. In this case, a bounding box containing the segment is computed and all the stable pixels within the bounding box will instead be used for the RANSAC plane-fitting.

As a last step in order to remove small differences between plane fitted unstable pixels and the original stable pixels, an adaptive smoothing is done. The adaptive smoothing kernel is computed as

$$K(p+u) = \begin{cases} 1 & \text{if } |D_\Pi(p) - D_\Pi(p+u)| < \sigma_p \\ 0 & \text{else} \end{cases}$$

where D_Π is the plane-fitted depth map, and $\sigma_p = 0.5$ is a constant. The size of this kernel is usually set to something relatively small like 9×9.



(a) Reference image.

(b) Color segmentation.

(c) Plane-fitted depth map.

(d) BP-based depth map.

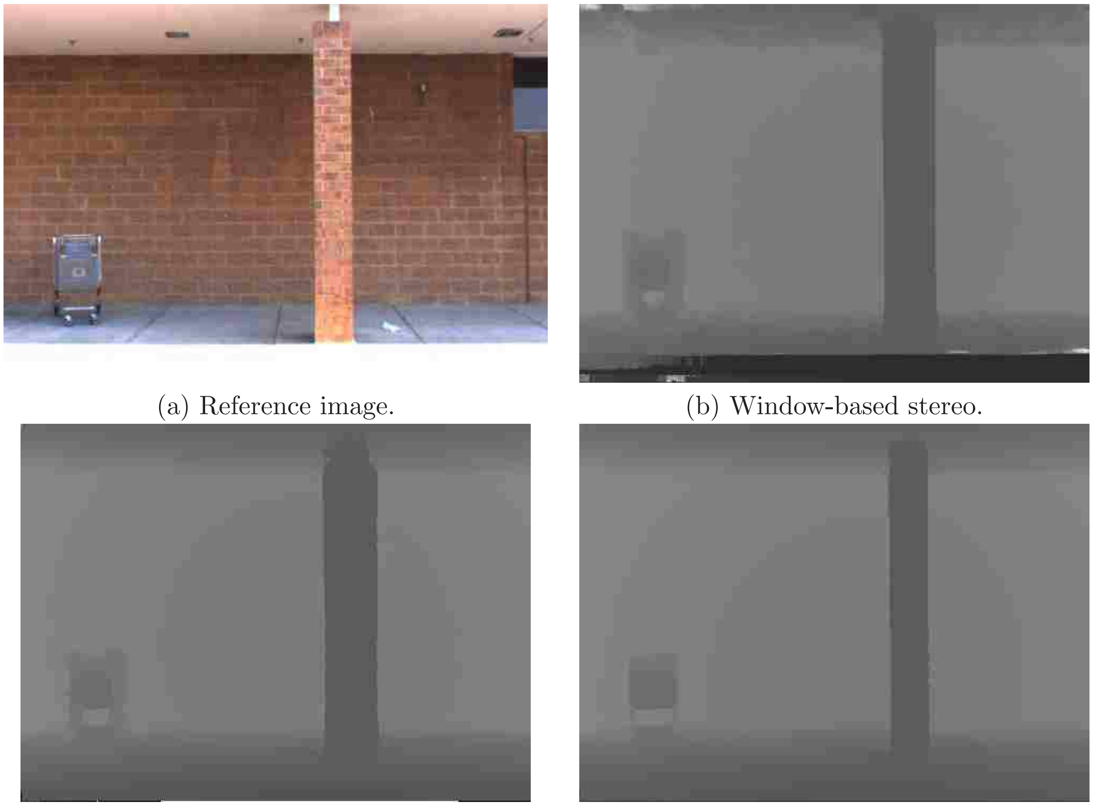
Figure 4.2: BP-based plane-fitting stereo. Due to strong sun shine, part of the column in (a) is joined with the ground in color segmentation as shown in (b). In this case, the plane-fitted stereo will fail. However, after BP refinement, the errors that appear in (c) are removed.

Finally, another pass of the consistency check is applied to the plane-fitted depth maps in order to update the confidence map. This confidence map is passed on to the mesh generation module where triangles only get created for depth values that have a high confidence.

4.5 BP-based Plane-fitting Stereo

The BP-based plane-fitting stereo pipeline contains three modules: BP-based stereo matching, stereo fusion and plane-fitted refinement. the last two modules are the same as the plane-fitting stereo pipeline described in Section 4.4. The first module is a bit different in that after window-based stereo matching, a plane-fitting depth map is calculated followed by a loopy belief propagation refinement, which helps correct the potential errors caused by plane-fitting due to non-robustness of the color segmentation.

After window-based stereo matching, The pixels are classified into stable pixels and unstable pixels based on the confidence map calculated from the correlation volume as it is described in



(a) Reference image.

(b) Window-based stereo.

(c) Plane-fitting stereo.

(d) BP-based stereo.

Figure 4.3: Visual comparison of depth maps. (b) is the depth map produced by a real-time local window-based stereo pipeline [4] (c) is the depth map produced by our near real-time plane-fitting stereo pipeline. Most of the errors in (b), such as the incorrect depth values in the textureless ground region are removed in (c). (d) is the depth map produced by our BP-based plane-fitting stereo pipeline. Clearly, the quality of (d) is higher than (c), notice for example how the thin parts of the shopping cart are preserved.

Section 4.3. For each large segment, the depth values of stable pixels are used to robustly fit a 3D plane to correct the depth values of the unstable pixels. However, due to the non-robustness of color segmentation, the plane-fitting may fail. To correct the potential errors, GPU hierarchical loopy belief propagation approach is implemented according to [57]. The new stereo problem is formulated as a Markov Random Fields (MRF) problem as follow:

$$P(D, L) = \frac{P(I|D, L)P(D, L)}{P(I)}, \tag{4.5}$$

where D is the depth map that is to be computed, L is the depth discontinuity, I refers to the input camera images. The occlusion issue is escaped and assumed to have been handled by multi-view stereo matching. Hierarchical loopy belief propagation algorithm is an efficient solution to the MRF problem, and [57] proved that by using GPU, the computation time of hierarchical BP can be greatly reduced while preserving comparable stereo quality. The loopy belief propagation state the stereo problem as follows

$$E(p,d) = E_D(p,d) + \sum_{q \in N_4(p)} E_S(q,d), \tag{4.6}$$

where p is a pixel, d is the depth hypothesis and $N_4(p)$ is the 4-connectivity set of p. The smoothness will be iteratively refined based on the assumption that the world surfaces are piecewise smooth.

$$E^{(i+1)}_{S,p\rightarrow t}(p,d) = \arg\min_{d_p}(E_D(p,d_p) + \sum_{q \in N_4(p), q \neq t} E^{(i)}_{S,q\rightarrow p}(q,d) + \Phi(d_p,d)), \tag{4.7}$$

$$\Phi(d_p,d) = \min(\lambda, |d_p - d|), \tag{4.8}$$

where $\lambda = 6.0$ is a constant used to control when the smoothness penalty should stop increasing. The data term E_D will stay unchanged and is defined as:

$$E_D(p,d) = C_m(p)\min(E_m(p,d), \eta_m) + (1 - C_m(p))\min(\beta(d - D_\pi(p))^2, \eta_\pi). \tag{4.9}$$

$C_m(p) \in [0,1]$ is the confidence map calculated based on the correlation volume. E_m is the correlation volume without boxcar aggregation, and D_π is the plane-fitted depth map. By integrating C_m, E_m and D_π, the data term E_D depends mostly on the plane-fitted depth map in the low confident areas, and depends on the correlation volume in the high confident areas. The constant $\eta_m = 50.0$ is used to reject outliers in the correlation volume. $\beta = 2.0$ is the rate of increase in the cost caused by the plane-fitted depth map D_π and $\eta_\pi = 50.0$ controls when the cost stops increasing. A good example about how BP correct the errors introduced by the plane-fitting stereo is shown in Figure 4.2. Due to the strong sun shine, part of the column in (a) is joined together with the ground if color segmentation is perform as it is shown in (b). In this case, the plane-fitted stereo will fail. However, after BP refinement, the errors that appear in (c) are removed.

To further improve the stereo quality, after BP refinement, the color-weighted filter designed in Section 4.4.1 is applied to E to help preserve the depth discontinuity under the assumption that color discontinuity also introduce the depth discontinuity. Note that after plane-fitting, the depth values of low confident areas has been corrected, thus the confidence map should be updated too. The following steps are the same as the plane-fitting stereo pipeline described in Section 4.4: stereo fusion and another pass of plane-fitting refinement. Figure 4.3 provides the final depth maps produced by different stereo pipelines for visual comparison. It shows that the BP-based plane-fitting stereo performs the best regarding both weakly-textured regions and the preservation of thin structures.

4.6 Experimental Results

For comparison, we run the three stereo pipelines on several urban datasets, and show the depth maps in Figure 4.4, 4.6 and the screenshots of their 3D models in Figure 4.5 and 4.7. The models are generated with the high confident pixels in the depth map. Hence if the confidence of a region is low, it will leave a hole in the 3D model. By visual comparison, the two proposed stereo pipelines outperform the window-based stereo pipeline a lot in the weakly-textured areas. For instance, part of the ground in Figure 4.5 is textureless, which leaves a lot of holes in the first row in Figure 4.5. However, the two proposed stereo pipelines successfully fill in most of the holes correctly. This is even more obvious in Figure 4.7, where there are lots of weakly-textured areas. However, there are not too much visual difference between the Plane-fitting Stereo pipeline and the BP-based Plane-fitting Stereo pipeline in the 3D models.

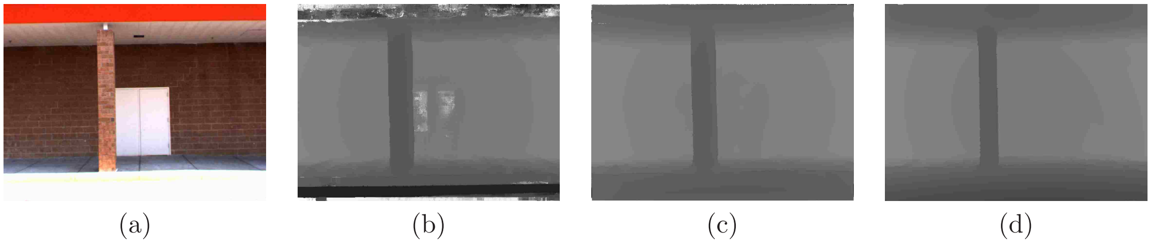


Figure 4.4: Visual comparison of depth maps associated with the 3D models shown below. From left to right: reference image, window-based stereo,plane-fitting stereo and BP-based plane-fittin. Notice how the depth estimates for the white textureless door are wrong in (b) and how that produces an incorrect surface in the corresponding 3D model below. Both (c) and (d) correctly handle the same door. Also notice that the very large depth errors found in the upper red and the bottom white textureless regions in (b) are discarded from the window-based 3D model since they otherwise would do much more harm than good, they are thresholded away based on the confidence map.

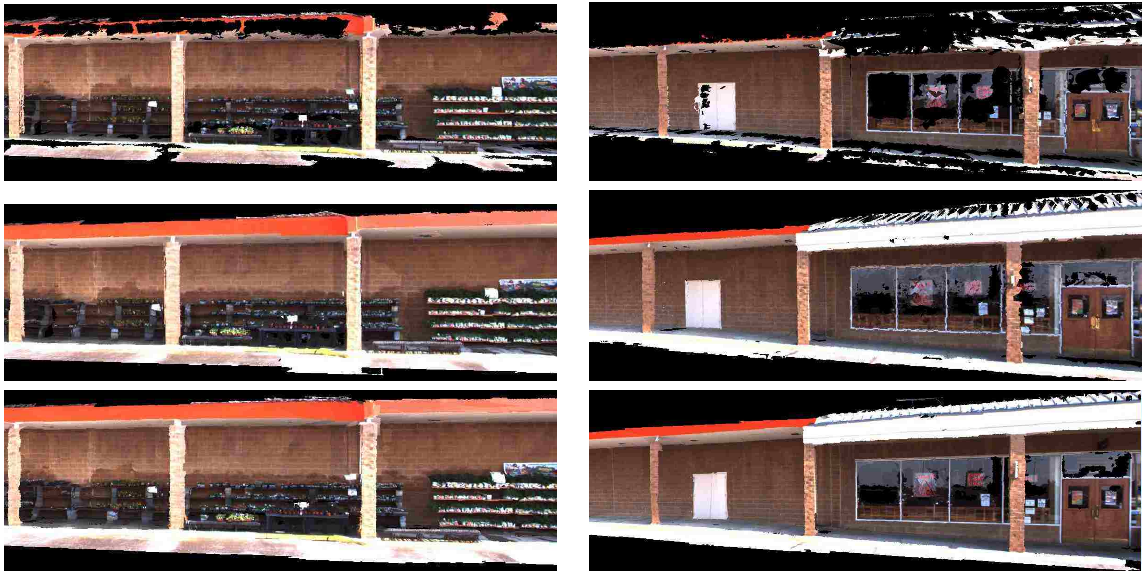


Figure 4.5: Screen shot of the 3D models. top: Window-based stereo. middle: Plane-fitting stereo. bottom: BP-based plane-fitting stereo. Notice how we obtain good surfaces for the textureless regions where the window-based stereo fails.

Sometimes the 3D model and the texture may be cheating the eyes. To present better comparison, the reference camera image and the depth maps produced at the end of the three stereo pipelines are shown in Figure 4.4 and 4.6. The depth map produced by the Window-based Stereo is fatten due to the boxcar aggregation, and the depth values of the weakly-textured areas such as part of the white door and part of the ground in Figure 4.4 are incorrect. The plane-fitting stereo helps correct the depth values in the weakly-textured areas but not the fattening errors. Also, in Figure 4.6, due to incorrect color segmentation, some of the errors still exist in the plane-fitted depth map. Clearly, all the problems are solved using BP-based plane-fitting stereo as it is shown in the last column of Figure 4.4 and 4.6, which proves that BP-based plane-fitting stereo has higher stereo quality than plane-fitting stereo.

In general, the two proposed stereo pipelines outperform the Window-based Stereo, while still providing good speed performance. With our settings the Window-based Stereo pipeline can process video data of resolution 512 × 384 and 48 depth hypotheses at about 18 frames per second using NVIDIA Geforce 8800 GTX graphics card and Intel Xeon (TM) 3.2GHz CPU, the Plane-fitting Stereo pipeline runs at about 8 frames per second, and the BP-based Plane-fitting Stereo pipeline runs at about 1 frame per second. Overall the Plane-fitting Stereo pipeline is the best due to its fast processing time and the ability to still produce good reconstruction accuracy.

4.7 Discussion

In this chapter, we focus on providing fast solutions to reconstruct the weakly-textured regions that are common in urban environments. The proposed solutions guarantees the local smoothness by using plane-based depth representation of the textureless segments. We demonstrate that the proposed solutions outperform the current-state-of-the-art in the weakly-textured areas by showing reconstructed 3D models and depth maps. We also compare the speed performance and show that our approach can process video sequences at near real-time speed. Considering both the reconstruction accuracy and the speed performance we provide good solutions for a large scale urban reconstruction system.

In our current near real-time plane-fitting stereo pipeline, we don't consider any smoothness cost across the neighboring segments. Although we provide an efficient solution with the BP-based plane-fitting stereo pipeline, it is time consuming. In the future, we are planning to re-formulate this plane-fitting problem as an energy minimization problem which includes both data term and smoothness term. The data term associated with a 3D plane hypothesis will be the sum of all the euclidean distance from the 3D points to the plane, and the smoothness term will be a function measuring the similarity of the plane hypothesis in the current segment and the plane hypotheses in all its neighboring segments. Some stereo algorithms [33, 36] are proved to be very good at solving this energy minimization problem. These methods are far away from being real-time because they are using mean-shift color segmentation which is time consuming, however this is not an issue using

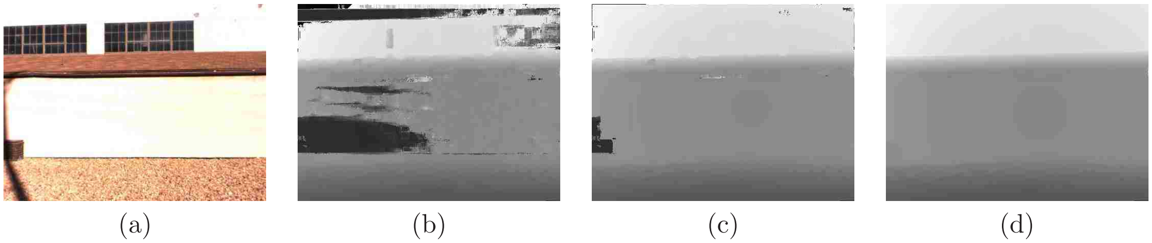

(a) (b) (c) (d)

Figure 4.6: Visual comparison of depth maps associated with the 3D models shown below. From left to right: reference image, window-based stereo,plane-fitting stereo and BP-based plane-fittin. Notice how the depth estimates for the white textureless wall above the windows and the wall bellow the roof are wrong in (b) and how that produces an incorrect surface in the corresponding 3D model below. (c) corrects most of errors in (b). However, due to bad color segmentation, some of the errors still exist, while (d) removes all the errors completely.

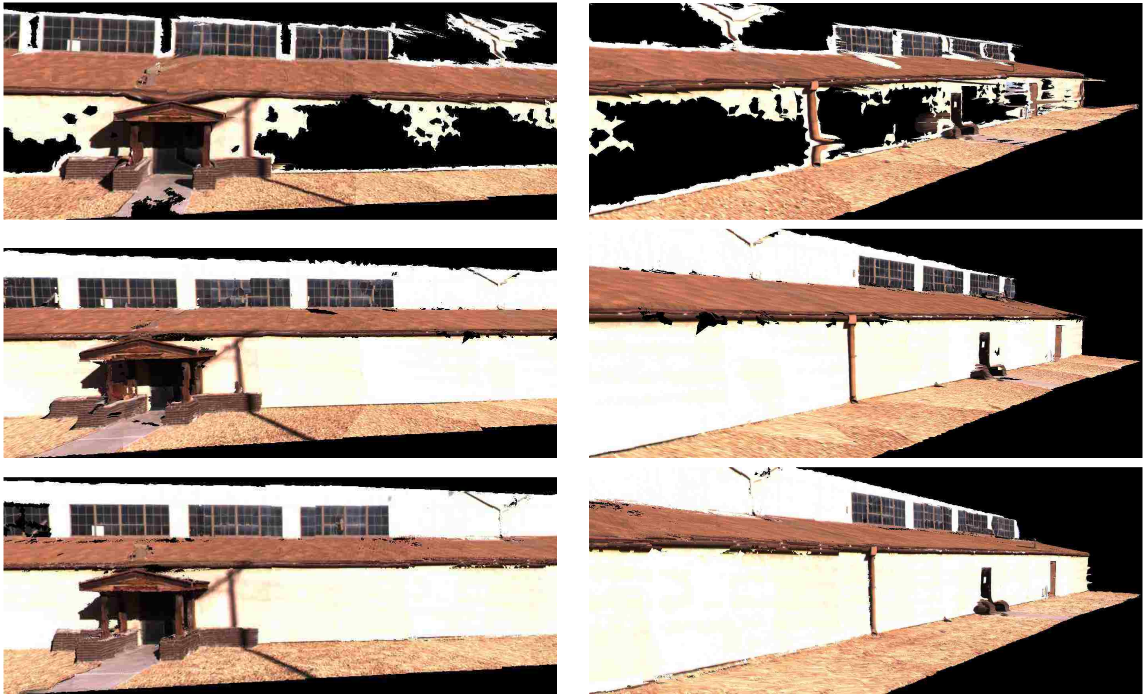


Figure 4.7: Screen shot of the 3D models. top: Window-based stereo. middle: Plane-fitting stereo. bottom: BP-based plane-fitting stereo.

our real-time segmentation method.

Chapter 5

Reconstruction from Range images: Spatial-Depth Super Resolution for Range Images

5.1 Introduction

There exists a variety of range measuring technologies to acquire 3D information about our world. For example, laser range scanners can provide extremely accurate and dense 3D measurement over a large working volume [6, 7, 29, 39, 41, 45]. However, most of these high-quality scanners measure a single point at a time, limiting their applications to static environments only. The options to capture depth at video rate are rather limited: the main contender–stereo vision–is known to be quite fragile in practice.

Recently new sensors [1, 3, 49] have been developed to overcome this limitation. By using extremely faster shutter (on the order of nanosecond), these sensors measure time delay between transmission of a light pulse and detection of the reflected signal on an entire frame at once. While the technology is promising, in the current generation, these sensors are either very expensive or very limited in terms of resolution. For example the Canesta EP DevKit sensors can provide range images only up to 64×64. Their applications are therefore limited to background segmentation and user interface control.

In this chapter we present a framework to substantially enhance the spatial and depth resolution of low-quality and highly quantized range maps, e.g., those from stereo vision or the Canesta sensor. Our approach takes advantage of the fact that a registered high-quality texture image can provide significant information to enhance the raw range map.

Most related to our work is a range-enhanced method by Diebel and Thrun [17], in which a Markov Random Field (MRF) is first designed based on the low resolution depth maps and high resolution camera images. The MRF is then solve with the well-known conjugate gradient (CG) algorithm [40]. This method gives promising spatial resolution enhancement up to 10×. Our formulation has demonstrated spatial resolution enhancement up to 100×.

Key to our success is the use of an adaptive filter, inspired by several state-of-the-art stereo algorithms [54, 56, 59]. In essence, we consider that the input range map provides a probabilistic distribution of depth, from which we can construct a 3D volume of depth probability, typically referred to as the *cost volume* in the stereo vision literature. Then we iteratively apply a color-weighted filter to the cost volume. The output high-resolution range image is produced by taking the winner-takes-all approach on the weighted cost volume and a sub-pixel refinement afterward.

This simple formulation turns out to be very effective. As demonstrated with a variety of real-world objects, it can provide not only visually compelling range images up to 100× resolution, but also a numerically more accurate depth estimate. We have applied our framework to *all* the algorithms reported on the middlebury stereo benchmark site [42]. Our depth-enhanced disparity maps, when compared to their original counter parts, are *superior in overall ranking for each and every algorithm listed*, including those already having sub-pixel disparity refinement.

The chapter is organized as follows: Section 5.2 presents an overview of our super resolution framework and the details about spatial resolution enhancement using an adaptive color-weighted filter and depth resolution enhancement by quadric polynomial interpolation. In Section 5.3 we then discuss how to enhance the depth resolution for general two-view stereo vision problems through a sub-pixel refinement step. The experimental results are reported in Section 5.4, followed by a conclusion in Section 5.5.

5.2 Approach

An overview of the framework of the approach is provided in Figure 5.1. First, up-sample the low-resolution depth map from the range image to the same size as the high-resolution camera image, save it as $D_{(0)}$. Then follows an iterative refinement module. A cost volume C_i is built based on the current depth map $D_{(i)}$, then a color-weighted aggregation is performed throughout each slice of the cost volume to produce the new cost volume $C_{(i)}^{CW}$. The refined depth map $D_{(i+1)}$ is generated based on this cost volume by first selecting the depth hypothesis with the minimal cost and a sub-pixel estimation afterwards.

5.2.1 Construction and refinement of the cost volume

At first, a coarse cost volume is built based on the current depth map. In order to allow large depth variations, as the current depth values are not guaranteed to be correct, the cost function should become constant as the differences become large. One such common function is the truncated quadric model, where the cost increases quadratically based on the distance between the potential depth candidate d and the currently selected depth $D_{(i)}(\mathbf{y},\mathbf{x})$

$$C_{(i)}(\mathbf{y},\mathbf{x},d) = \min(\eta * L, (d - D_{(i)}(\mathbf{y},\mathbf{x}))^2) \tag{5.1}$$

L is the search range, η is a constant. The square difference is selected as the cost function since we will use quadratic polynomial interpolation for sub-pixel estimation later. This cost function can help to preserve the sub-pixel accuracy of the input depth map.

A color-weighted aggregation is then applied to each slice of the cost volume based on the following prior assumptions:

1. World surfaces are piecewise smooth.
2. The pixels with similar colors around a region are likely to have similar depth.

The color-weighted filter was first presented in [59], and then integrated into the stereo algorithm proposed in [56] which is one of the middlebury top algorithms. The experimental results in

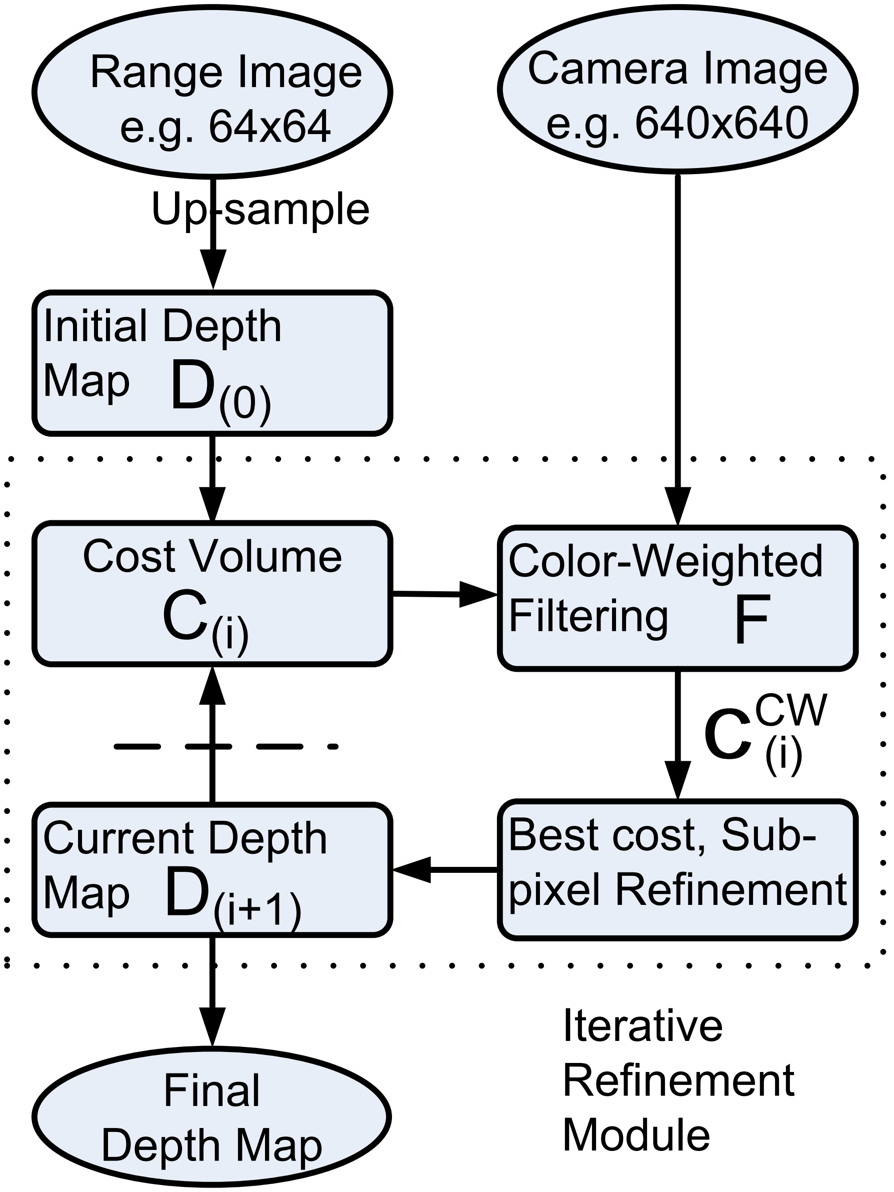

Range Image
e.g. 64x64
Camera Image
e.g. 640x640
Up-sample
Initial Depth Map $D_{(0)}$
Cost Volume $C_{(i)}$
Color-Weighted Filtering F
$C^{CW}_{(i)}$
Best cost, Sub-pixel Refinement
Current Depth Map $D_{(i+1)}$
Final Depth Map
Iterative Refinement Module

Figure 5.1: Framework of our post-processing approach. The range image is up-sampled to the same size as the camera image, and serves as the initial depth map hypothesis. The following is an iterative refinement process. A cost volume is built according to the current depth map hypothesis. A color-weighted filter is then applied to the cost volume to handle the fattening problem near depth discontinuities. A winner-take-all and sub-pixel estimation procedure is used to produce a new depth map hypothesis, which is fed back into the process.

both papers show that the color-weighted filter works very well near discontinuities, which is the main challenge for spatial super-resolution discussed in this chapter. For the smooth areas, after up-sampling, all the missed sampling areas are filled in correctly by interpolation. However, this is generally not true for the discontinuous areas. The missed sampling areas are blurred after up-sampling. But by using the color information provided by the registered camera images, we demonstrate that it is possible to get sharp/true depth edges for stereo spatial super resolution. This is the central theme of the chapter.

The color-weighted filter is designed as following:

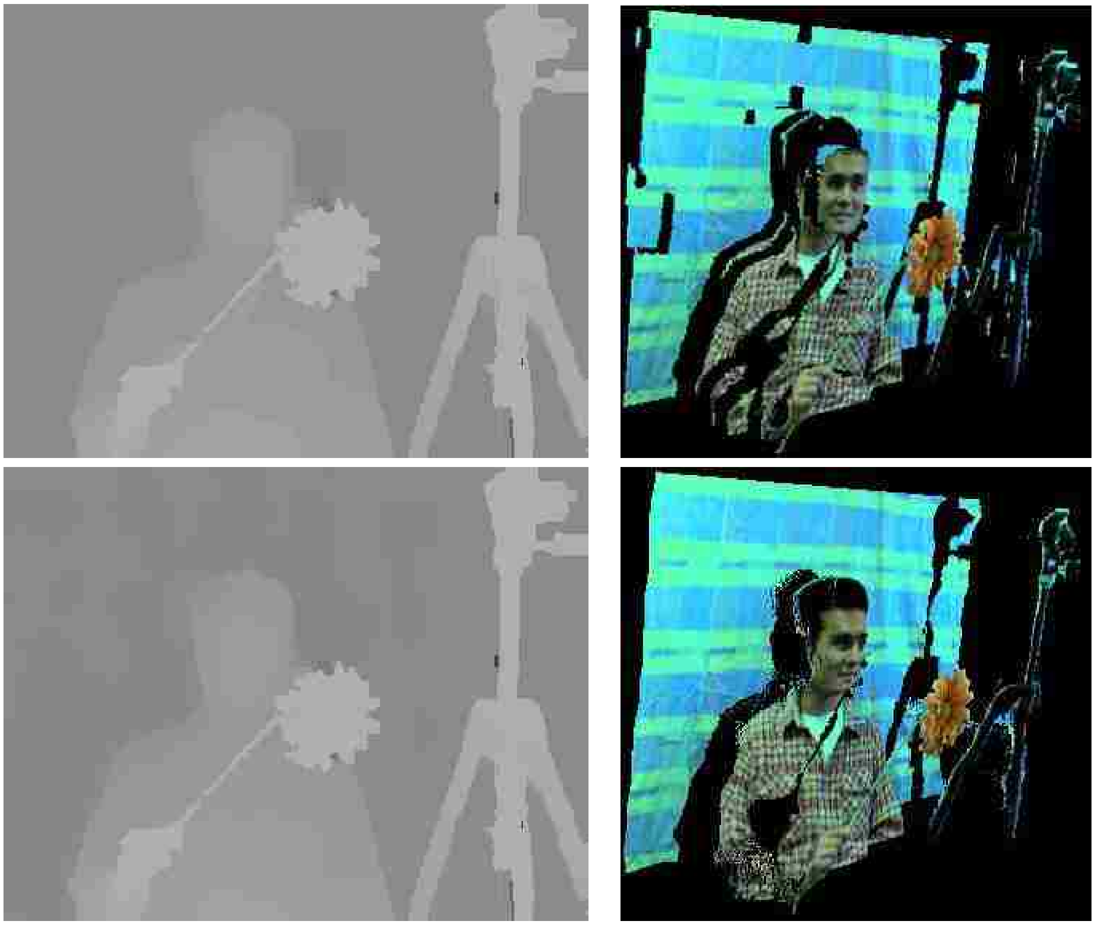
$$\begin{aligned}
F(\mathbf{y}+u, \mathbf{x}+v) &= f_c(W_c(\mathbf{y},\mathbf{x},u,v))f_s(W_s(u,v)), \qquad (5.2)\\
f_c(x) &= exp(-\frac{|x|}{\gamma_c}),\\
f_s(x) &= exp(-\frac{|x|}{\gamma_s}),\\
W_c(\mathbf{y},\mathbf{x},u,v) &= \frac{1}{3}(|R(\mathbf{y}+u, \mathbf{x}+v) - R(\mathbf{y},\mathbf{x})|\\
&+ |G(\mathbf{y}+u, \mathbf{x}+v) - G(\mathbf{y},\mathbf{x})|\\
&+ |B(\mathbf{y}+u, \mathbf{x}+v) - B(\mathbf{y},\mathbf{x})|),\\
W_s(u, v) &= \sqrt{u^2 + v^2}.
\end{aligned}$$

$\mathbf{y}$, $\mathbf{x}$ are the indices of the current pixel in the camera image, and u, v are two variables. R, G, B are the RGB channels of the camera image. γ_c and γ_s are two constants used as the thresholds of the color difference and the filter size. The color-weighted filter works as soft color segmentation in the super resolution framework, which aggregates the probabilities of each depth candidates of the pixels around a region based on the color similarity of the central pixel and its neighbors.

As it is shown in Figure 5.1, the color-weighted filter is iteratively applied to the current cost volume to construct the color-weighted cost volume, then we search through all the depth hypotheses and select the one with the minimal cost. Finally, sub-pixel estimation is performed based on the color-weighted cost volume and the depth hypotheses with the minimal cost.

5.2.2 Sub-pixel Estimation

To reduce the discontinuities caused by the quantization in the depth hypothesis selection process, a sub-pixel estimation algorithm is proposed based on quadratic polynomial interpolation. If the cost function is continuous, the depth with the minimum matching cost can be found. However, the cost function is discrete in practice. The search range is limited, which results in discontinuous depth maps. In order to eliminate this effect, we use quadratic polynomial interpolation to approximate the cost function between three discrete depth candidates: d, d_- and d_+. d is the discrete depth with the minimal cost, $d_- = d - 1$, and $d_+ = d + 1$.



(a) Depth maps.

(b) Synthesized views.

Figure 5.2: Experimental results with and without sub-pixel refinement. (a) Depth maps generated with the **DoubleBP** algorithm [56] reported on the middlebury website. (b) Synthesized views using (a). First row shows results without sub-pixel refinement, second row shows results with sub-pixel refinement. Notice that the quantization effect on the man's face and the background on the synthesized view before sub-pixel is removed after sub-pixel estimation.

$$\begin{aligned} f(x) &= ax^2 + bx + c, && (5.3) \\ x_{min} &= \frac{-b}{2a}, && (5.4) \end{aligned}$$

$f(x_{min})$ is the minimum of function $f(x)$. Thus, given d, $f(d)$, $f(d_-)$ and $f(d_+)$, the parameters a and b of the continuous cost function can be calculated. Thus:

$$x_{min} = d - \frac{f(d_+) - f(d_-)}{2(f(d_+) + f(d_-) - 2f(d))}, \tag{5.5}$$

x_{min} is the depth with the minimum of the quadric cost function $f(x)$. Figure 5.2 provides a visual comparison of the depth maps and their synthesized views before and after sub-pixel estimation.

Notice that the quantization effect on the man's face and the background on the synthesized view is removed after sub-pixel estimation.

5.3 Extended depth super resolution with two views

The main difference between one-view super resolution and two-view super resolution is the construction of the cost volume. In two view case, general stereo matching algorithm can be performed, together with the range image, to provide a more accurate cost volume. At first, The cost volume is initialized with zero. Then three depth candidates d, d_{-}, d_{+} are selected according to the input depth map. d is extracted from the input depth map, $d_{-} = d - 1$ and $d_{+} = d + 1$. To perform depth enhancement with two views, three slices of matching cost based on each of the three depth candidates are calculated. The calculation of matching cost is implemented according to the symmetric color-weighted correlation approach presented in [56]. First, project the pixel in the reference view to the other view using the depth candidates calculated from the input depth map, and the matching cost is the pixel dissimilarity of the corresponding pixels. To reduce the noise, Birchfield and Tomasi's pixel dissimilarity [8] is used. Second, a symmetric color-weighted filtering is applied to the cost slices:

$$F_{symm}(\mathbf{y}+u, \mathbf{x}+v) = F(\mathbf{y}+u, \mathbf{x}+v) F(\mathbf{y}'+u, \mathbf{x}'+v), \tag{5.6}$$

$F(\mathbf{y} + u, \mathbf{x} + v)$ is the filter defined in Equation 5.3, $\mathbf{y}$, $\mathbf{x}$ is the index of the current pixel in the reference view, and $\mathbf{y}'$, $\mathbf{x}'$ is the index of the corresponding pixel in the other view.

The sub-pixel depth enhancement is performed by a quadratic polynomial interpolation with the symmetric color-weighted cost volume as it is described in Section 5.2.2. Finally, a box-car filter (G) is applied to smoothen the depth map with a hit-or-miss scheme:

$$G(\mathbf{y} + u, \mathbf{x} + v) = \begin{cases} 1.0 & \text{if } |D_0(\mathbf{y}, \mathbf{x}) - D_0(\mathbf{y} + u, \mathbf{x} + v)| < 1 \\ 0 & \text{else} \end{cases}$$

D_0 is the input depth map. The size of the box-car used is relatively small (9x9).

To validate our sub-pixel refinement approach, an off-line stereo benchmark that has the same scoring scheme as the middlebury benchmark [42] is built. Every result reported on the middlebury website is used with our sub-pixel refinement approach, and evaluated on the sub-pixel benchmark. Figure 5.3 shows that our approach is very robust, it works for all the algorithms, even for those originally having sub-pixel refinement. The completed version of Figure 5.3 is provided in the supplemental materials, which gives more details about the ranks on different datasets and error thresholds.

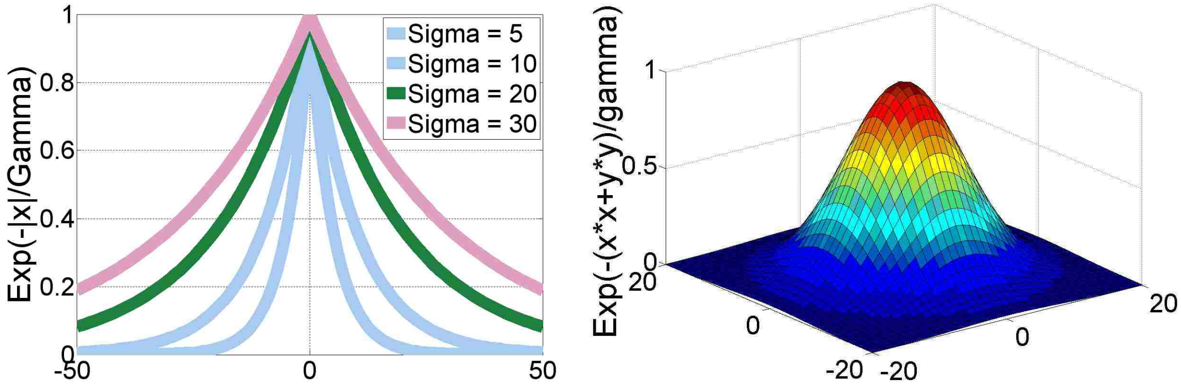
Algorithms	**Average Rank**			
	Two Views		**Single View**	
	Before	After	Before	After
DoubleBP	21.5	**5.75**	18.42	**11.33**
AdaptingBP	15.33	**6.92**	**12.58**	**9.17**
C-SemiGlob	**11.75**	**7.25**	**7.75**	**5.25**
Segm+visib	16.08	**9**	**14.67**	**11.17**
SymBP+occ	25	**12**	22.42	**14.92**
SemiGlob	15	**12.33**	**12**	**9.83**
AdaptWeight	29.25	**12.42**	25.58	14.92
RegionTreeDP	32.75	**14**	29.75	18.17
GC+occ	25.33	**14.42**	24	19.08
TensorVoting	24.83	17.67	22.08	16.33
MultiCamGC	28.17	17.83	27	23.08
Layered	34.83	18.08	32.25	25
SegTreeDP	28.33	18.17	26.33	17.42
RealtimeBP	34	18.92	31.42	21.92
CostRelax	23.58	20.17	21.58	21.17
GenModel	22.17	20.83	20.25	17.58
ReliabilityDP	40.17	23.5	37.5	29.5
RealTimeGPU	38.42	24	36.58	24.67
GC	33.67	24.5	32.42	29.25
TreeDP	44	30.5	42.5	36.67
DP	43.92	31.33	42.17	31.17
SSD+MF	46.08	34.75	45.17	41.75
STICA	44.67	35.25	44	35.58
SO	45.17	37.83	43.42	36.75
Infection	44.83	38.75	43.42	38.08

Figure 5.3: Sub-pixel estimation evaluation. The scores on the last four columns are the average ranks with error threshold 0.5. The scores with bold font are among the top 10 performers. The entries with blue highlighting are stereo algorithms originally without sub-pixel estimation, the others are algorithms originally having sub-pixel estimation. The scoring scheme is the same as the middlebury benchmark [42].

5.4 Experimental Results

Our experimental system consists of a Canesta EP DevKit camera [1] and a FLEA digital camera at [12]. The EP DevKit camera can produce range images with size up to 64 × 64 of the objects in its view, and the FLEA camera can produce color images with resolution up to 1024 × 768. These two cameras are placed very close to each other and image registration is achieved by a 3 × 3 homographic warp. The warping function is dependent on the average range to the object. A better setup would be to use an beam-splitter to align the optical axes of both sensors to guarantee image alignment.

Three main parameters are involved in the experiment, they are η, γ_c and γ_s. η is the constant used in Equation 5.1, it is set to 0.5 experimentally. To allow large depth variations, the cost function is truncated by $\eta \times L$, where L is the search range. Two parameters are involved in the color-weighted filter, they are γ_c and γ_s. In this paper, they are both set to 10. A visual explanation about how these parameters control the shape of the weighting functions in Equation 5.3 is provided in Figure 5.4. The experimental results show that γ_c is relatively sensitive, it should be decreased around the low texture area.



(a)

(b)

Figure 5.4: (a) $\gamma_c \in \{5, 10, 20, 30\}$. (b) $\gamma_s = 10$.

5.4.1 Spatial super resolution

To show the power of the iterative color-weighted filtering, a series of intermediate depth maps are provided in Figure 5.5 in a coarse-to-fine manner architecture.

In Figure 5.5, D_0 is the depth map after up-sampling from the low-resolution range image. The quality of D_0 is unacceptable. D_1 is the depth map after iteration 1. The quality has been improved a lot, but the areas around part of the discontinuities are incorrect. D_3 is the depth map after iteration 3, the discontinuities are well detected, and the algorithm has almost converged. D_{10} is the depth map after iteration 10. By visual comparison, the difference between D_{10} and D_3 is tiny. Others experimental results are shown in Figure 5.6. The input depth maps are up-sampled from

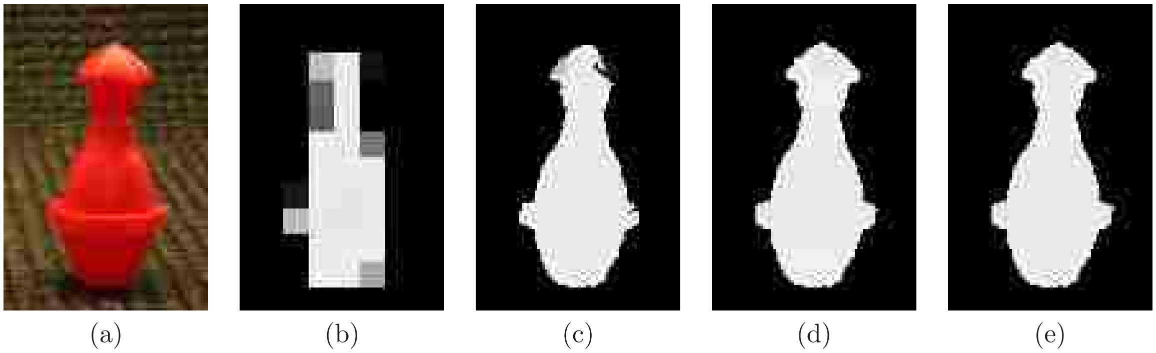


Figure 5.5: Intermediate results from iterative color-weighted refinement module. (a) Camera image. (b) The initial depth map (D_0). (c) Depth map after one iteration (D_1). (d) Depth map after three iterations (D_3). (e) Depth map after ten iterations (D_{10}).

the 64 × 64 range images, and the resolution of the output depth maps is 640 × 640.

Table 5.1 evaluates the performance of our approach and the MRF approach presented in [17] on the middlebury datasets on three different scales. On each scale, the depth image is down-sampled by a factor of 2 gradually. On scale 0, the depth image is the ground truth. By comparing the bad pixel percentages before and after color-weighted refinement, we show that our approach improves the stereo quality of all data sets. The MRF approach in [17] also improves the stereo quality, but the improvement is relatively small compared to our approach. A visual comparison of the depth maps of the middlebury datasets on Scale 3 are provided in Figure 5.7. Clearly, the results using our approach have more clean edges than the input depth maps and the results using MRF approach. For further comparison, Figure 5.8 provides the experimental results of the Cones data set from scale 1 to scale 4 using the MRF approach and our approach. By visual comparison, our approach outperforms the MRF approach as the resolution of the range sensor keeps on decreasing. On the last row in Figure 5.8, we show that even with tiny sensors (23 × 28), we can still produce decent high-resolution range images.

Algorithms	Tsukuba			Venus			Teddy			tsukuba		
	Scale			Scale			Scale			Scale		
	1	2	3	1	2	3	1	2	3	1	2	3
Before refinement	2.67	5.18	9.66	0.61	1.34	2.79	2.92	8.64	14.7	3.92	7.85	14.7
MRF approach [17]	2.51	5.12	9.68	0.57	1.24	2.69	2.78	8.33	14.5	3.55	7.52	14.4
Our approach	1.16	2.56	6.95	0.25	0.42	1.19	2.43	5.95	11.5	2.39	4.76	11.0

Table 5.1: Experimental results on the Middlebury datasets. The numbers in the last twelve columns are the percentages of the bad pixels with error threshold 1.

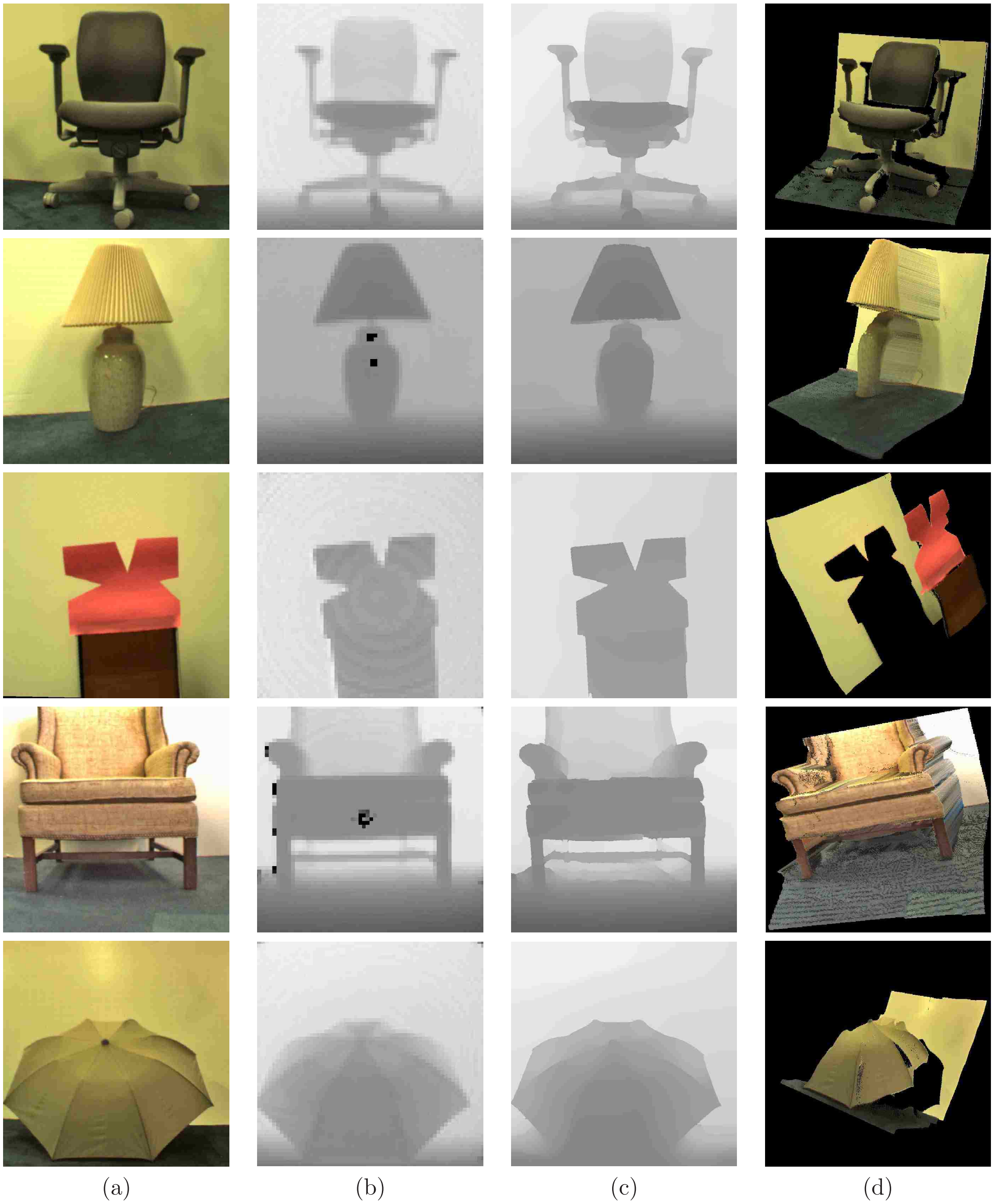


Figure 5.6: Experimental results of depth super resolution. (a) Camera images. (b) Input depth maps. (c) Refined depth maps. (d) Synthesized views by using (c). The input depth maps are up-sampled from range image with resolution 64 × 64, and resolution of the refined depth maps is 640 × 640. The spatial resolution is enhanced 100×.

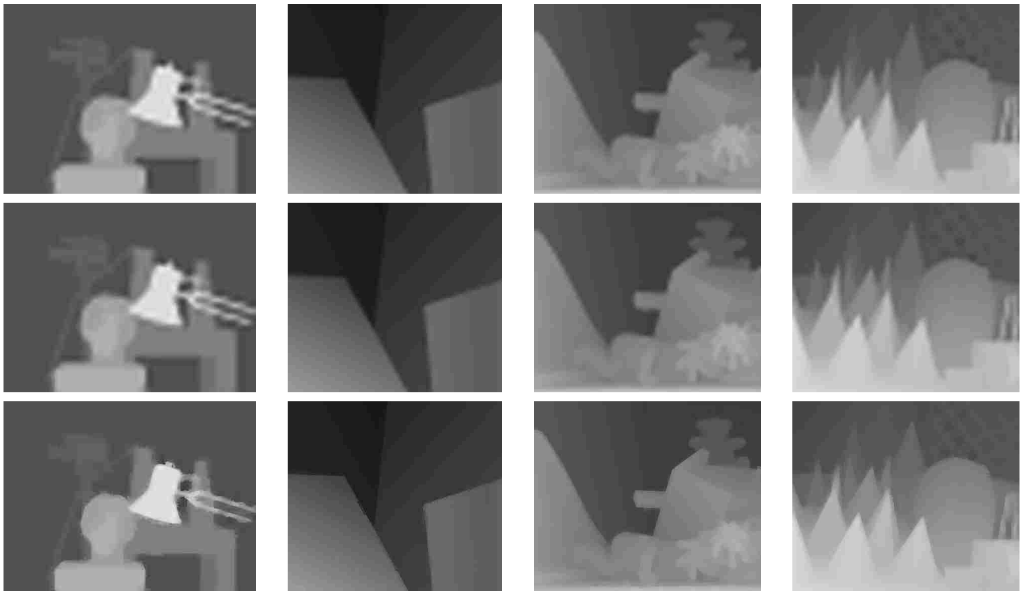
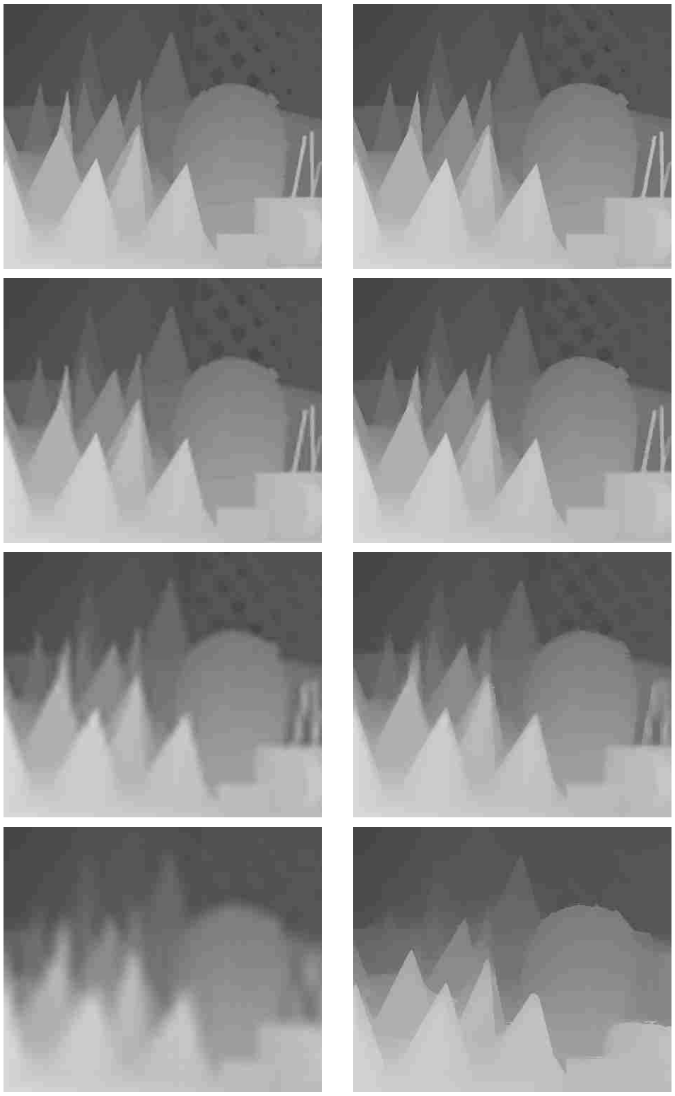


Figure 5.7: Super resolution on Middlebury datasets. From top to bottom: Before refinement, using MRF approach [17], using our approach.

5.4.2 Sub-pixel estimation with one or two reference image(s)

Besides the enhancement of the spatial resolution of range images, our approach also provides sub-pixel estimation for general stereo algorithms with either one or two camera image(s). To evaluate the performance of our sub-pixel estimation approach, we established an off-line stereo benchmark. The scoring scheme is the same as the middlebury benchmark. In our off-line benchmark, all the algorithms reported to the middlebury benchmark and their sub-pixel refinement results are evaluated, thus the total number of algorithms evaluated is twice the number on the middlebury benchmark [42]. Figure 5.3 provides the average ranks for all the algorithms. With either one or two view(s), we achieve across-the-board improvement for sub-pixel accuracy. The 10 entries with bold font are the top 10 performers. In two-view case, nine of them are the algorithms with our sub-pixel refinement approach. All the entries without blue highlighting in Figure 5.3 are average ranks of those algorithms using its own sub-pixel refinement techniques. The experimental results show that our sub-pixel estimation approach works for all of these algorithms, however the improvement is naturally a bit smaller than for the cases that originally don't have any kind of sub-pixel refinement. A set of synthesized views built from the **DoubleBP** algorithm [56] are shown in Figure 5.9, providing a visual comparison of the algorithms with and without sub-pixel refinement. The depth enhancement is obvious. The results shown in column (a) are quantized to discrete number of planes. After sub-pixel estimation, the quantization effect is removed, as it is shown in column (b).

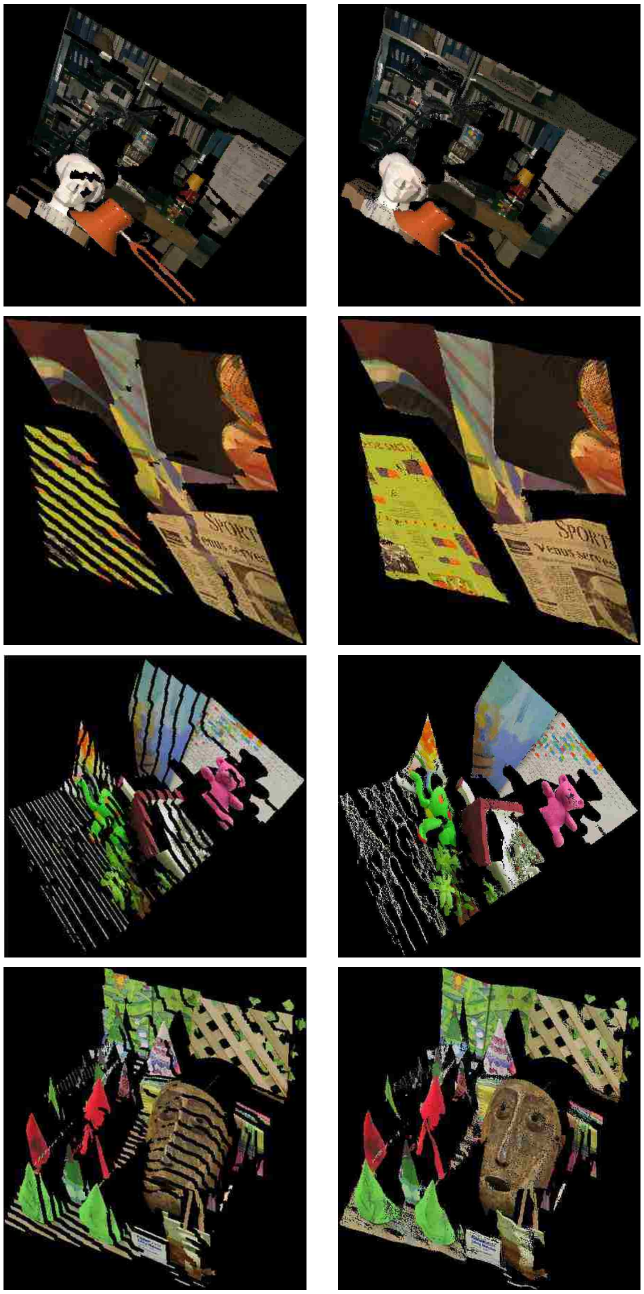


(a) Using MRF approach. (b) Using our approach.

Figure 5.8: Super resolution on Cones datasets. From up to bottom: Experimental results on scale 1 (resolution: 187×225), Experimental results on scale 2 (resolution: 93×112), Experimental results on scale 3 (resolution: 46×56), Experimental results on scale 4 (resolution: 23×28). This figure shows that, by visual comparison, our approach performs better than the MRF approach as the resolution of the range sensor continues to drop.

5.5 Conclusion

In this chapter, we present a new post-processing step to enhance the spatial resolution of range images up to 100x with a registered and potentially high-resolution color image as reference. We have validated our approach on several real datasets, including the Middlebury data set, demonstrating that our approach gives clear improvements. In addition, the depth super resolution is extended



(a) Before.

(b) After.

Figure 5.9: Sub-pixel refinement on Middlebury datasets. (a) Synthesized views produced by the DoubleBP algorithm [56] reported on the Middlebury website. (b) Synthesized views after sub-pixel refinement. The results shown in column (a) are quantized to discrete number of planes, after sub-pixel estimation, the quantization effect is removed, as it is shown in column (b).

to two-view case. To evaluate the effectiveness of our depth-enhanced approach, we first built an off-line stereo benchmark that has the same scoring scheme as the Middlebury benchmark, then tried our approach on all the stereo algorithms reported to the Middlebury benchmark. Together with all the results submitted to Middlebury benchmark, we evaluated all the depth-enhanced results on the off-line benchmark with different error thresholds, and showed across-the-board improvement in sub-pixel accuracy. We are hoping to release an on-line sub-pixel benchmark in the near future.

Chapter 6

Conclusion

In this thesis, an important task in computer vision has been studied : 3D structure recovery from stereo/range images of a rigid 3D scene. The main contribution is several works on different algorithm in a general 3D reconstruction framework: from two-view to multi-view stereo, from off-line to on-line stereo, and from reconstruction from stereo images to reconstruction from low-resolution range image.

In the scope of reconstruction from stereo images, the algorithms generally perform (subsets of) the following four steps:

1. matching cost computation;
2. cost (support) aggregation;
3. disparity computation / optimization; and
4. disparity refinement.

The algorithms proposed/studied in this thesis are focused on the last three steps, however the quality of all of them depends on the how accurate the first step can achieve. The matching cost computation is the most fundamental step in this field. A lot of approaches can be used to improve the matching accuracy, and the most obvious ones are: increasing the range of the depth hypotheses, increasing the image resolution, and taking pictures under better lighting conditions. The first two approaches are easy to implement, but at the cost of some speed performance. The third approach involves human interaction, which is not central theme of the thesis.

In general, our methods are usually easy to use and quite efficient, compared with their counterparts in the literature. we proved that the proposed algorithms are the current state-of-the-art in the scope of either reconstruction accuracy or speed in the same condition. However, the reconstruction result produced with all the stereo algorithms studied in the current framework won't be able to compared against the ground truth, even the resolution of the image and the range of the depth hypotheses is larger enough. To break through the bottleneck, high level vision task, such as object recognition should be employed. For instance assume there is a window in the scene, we first recognize that it is a window, and then fit a plane to it; or say that there is a car, we first classify it as a car, and then recognize that it is a 2006 Nissan Versa car. We thus query the database to extract the data of this model, and the ground truth is approaching. The best paper [28] at $CVPR$2006 is working in a similar direction. Instead of integrating object recognition for better 3D reconstruction, this paper provides a framework of object detection with early estimation of 3D geometry.

Bibliography

[1] Canestavision^{TM} electronic perception development kit, canesta inc. ttp://www.canesta.com/html/$development_kits$.htm.

[2] Edge detection and image segmentation (edison) system. ttp://www.caip.rutgers.edu/riul/research/code/EDISON/.

[3] Z-cam, 3dv systems. ttp://www.3dvsystems.com/home/index.html.

[4] A. Akbarzadeh, J.-M. Frahm, P. Mordohai, B. Clipp, C. Engels, D. Gallup, P. Merrell, M. Phelps, S. Sinha, B. Talton, L. Wang, Q. Yang, H. Stewéius, R. Yang, G. Welch, H. Towles, D. Nisté, and M. Pollefeys. Towards urban 3d reconstruction from video. In *3DPVT*, 2006. Invited paper.

[5] C. Baillard and A. Zisserman. Automatic reconstruction of piecewise planar models from multiple views. *CVPR*, 02:2559, 1999.

[6] J. Batlle, E. Mouaddib, and J. Salvi. Recent progress in coded structured light as a technique to solve the correspondence problem: A survey. *Pattern Recognition*, 31(7):963–982, 1998.

[7] P. Besl. *Active Optical Range Imaging Sensors, in Advances in Machine Vision, chapter 1*, pages 1–63. 1989.

[8] S. Birchfield and C. Tomasi. A pixel dissimilarity measure that is insensitive to image sampling. *PAMI*, 20(4):401–406, 1998.

[9] M. Bleyer and M. Gelautz. A layered stereo algorithm using image segmentation and global visibility constraints. In *ICIP*, pages 2997–3000, 2004.

[10] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. *PAMI*, 23(11):1222–1239, 2001.

[11] Y. Boykov, O. Veksler, and R. Zabith. A variable window approach to early vision. *PAMI*, 20(12):1283–1294, 1998.

[12] Flea Camera. Point grey research. ttp://www.ptgrey.com/products/flea/index.asp.

[13] R.T. Collins. A space-sweep approach to true multi-image matching. In *CVPR*, pages 358–363, 1996.

[14] D. Comaniciu and P. Meer. Mean shift: A robust approach toward feature space analysis. *PAMI*, 24(5):603–619, 2002.

[15] N. Cornelis, K. Cornelis, and L. J. V. Gool. Fast compact city modeling for navigation pre-visualization. In *CVPR (2)*, pages 1339–1344, 2006.

[16] P. E. Debevec, C. J. Taylor, and J. Malik. Modeling and rendering architecture from photographs: A hybrid geometry- and image-based approach. *SIGGRAPH*, pages 11–20, 1996.

[17] J. Diebel and S. Thrun. An application of markov random fields to range sensing. In *NIPS*, 2005.

[18] P. F. Felzenszwalb and D. P. Huttenlocher. Efficient belief propagation for early vision. In *CVPR (1)*, pages 261–268, 2004.

[19] P. F. Felzenszwalb and D. P. Huttenlocher. Efficient graph-based image segmentation. *IJCV*, 59(2):167–181, 2004.

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

[21] S. Forstmann, Y. Kanou, J. Ohya, S. Thuering, and A. Schmitt. Real-time stereo by using dynamic programming. In *CVPRW(3)*, page 29, Washington, DC, USA, 2004. IEEE Computer Society.

[22] C. Früh and A. Zakhor. An automated method for large-scale, ground-based city model acquisition. *IJCV*, 60(1):5–24, 2004.

[23] M. Gong and Y.-H. Yang. Near real-time reliable stereo matching using programmable graphics hardware. In *CVPR(1)*, pages 924–931, Washington, DC, USA, 2005. IEEE Computer Society.

[24] H. Hirschm*ü*lle, P. R. Innocent, and J. Garibaldi. Real-time correlation-based stereo vision with reduced border errors. *IJCV*, 47(1-3):229–246, 2002.

[25] H. Hirschm*ü*ller. Improvements in real-time correlation-based stereo vision. In *SMBV*, page 141, Washington, DC, USA, 2001. IEEE Computer Society.

[26] H. Hirschmuller. Accurate and efficient stereo processing by semi-global matching and mutual information. In *CVPR(2)*, pages 807–814, Washington, DC, USA, 2005. IEEE Computer Society.

[27] H. Hirschmuller. Stereo vision in structured environments by consistent semi-global matching. In *CVPR*, pages 2386–2393, 2006.

[28] D. Hoiem, A. A. Efros, and M. Hebert. Putting objects in perspective. In *CVPR*, pages 2137–2144, Washington, DC, USA, 2006. IEEE Computer Society.

[29] R. Jarvis. A perspective on range finding techniques for computer vision. *PAMI*, 5(2):122–139, 1983.

[30] T. Kanade and M. Okutomi. A stereo matching algorithm with an adaptive window: Theory and experiment. *PAMI*, 16(9):920–932, 1994.

[31] S. Kang, R. Szeliski, and J. Chai. Handling occlusions in dense multiview stereo. In *CVPR*, 2001.

[32] J. C. Kim, K. M. Lee, B. T. Choi, and S. U. Lee. A dense stereo matching using two-pass dynamic programming with generalized ground control points. In *CVPR(2)*, pages 1075–1082, Washington, DC, USA, 2005. IEEE Computer Society.

[33] A. Klaus, M. Sormann, and K. Karner. Segment-based stereo matching using belief propagation and a self-adapting dissimilarity measure. In *ICPR*, pages 15–18, 2006.

[34] V. Kolmogorov and R. Zabih. Computing visual correspondence with occlusions using graph cuts. In *ICCV(1)*, pages 508–515, 2001.

[35] V. Kolmogorov and R. Zabih. Multi-camera scene reconstruction via graph cuts. In *ECCV(3)*, pages 82–96, London, UK, 2002. Springer-Verlag.

[36] C. Lei, J. Selzer, and Y.-H. Yang. Region-tree based stereo using dynamic programming optimization. In *CVPR*, pages 2378–2385, 2006.

[37] P. Meer and B. Georgescu. Edge detection with embedded confidence. *PAMI*, 23(12):1351–1365, 2001.

[38] Y. Ohta and T. Kanade. Stereo by intra- and inter-scanline search using dynamic programming. *PAMI*, 7(2):139–154, March 1985.

[39] D. Poussart and D. Laurendeau. *3-D Sensing for Industrial Computer Vision, in Advances in Machine Vision, chapter 3*, pages 122–159. 1989.

[40] W. H. Press. *Numerical recipes in C: the art of scientific computing.* Cambridge University Press, New York, 1988.

[41] J. Salvi, J. Pagès, and J. Batlle. Pattern codification strategies in structured light systems. *Pattern Recognition*, 37(4):827–849, 2004.

[42] D. Scharstein and R. Szelisk. Middlebury stereo vision research page. ttp://bj.middlebury.edu/ schar/stereo/newEval/php/results.php.

[43] D. Scharstein and R. Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *IJCV*, 47(1-3):7–42, 2002.

[44] L. Shafarenko, M. Petrou, and J. Kittler. Automatic watershed segmentation of randomly textured color images. *IEEE Trans. Image Processing*, 6(11):1530–1544, 1997.

[45] T. C. Strand. Optical three-dimensional sensing for machine vision. *Optical Engineering*, 24(1):33–40, 1985.

[46] C. Sun. Fast stereo matching using rectangular subregioning and 3d maximum-surface techniques. *IJCV*, 47(1-3):99–117, 2002.

[47] J. Sun, Y. Li, S. Kang, and H. Shum. Symmetric stereo matching for occlusion handling. In *CVPR(2)*, pages 399–406, 2005.

[48] J. Sun, N.-N. Zheng, and H.-Y. Shum. Stereo matching using belief propagation. *PAMI*, 25(7):787–800, 2003.

[49] CSEM SA Swiss Ranger SR-2. The swiss center for electronics and microtechnology. ttp://www.csem.ch/fs/imaging.htm.

[50] H. Tao and H. Sawhney. Global matching criterion and color segmentation based stereo.wacv, pages 246-253, 2000.

[51] O. Veksler. Stereo correspondence with compact windows via minimum ratio cycle. *PAMI*, 24(12):1654–1660, 2002.

[52] O. Veksler. Fast variable window for stereo correspondence using integral images. In *CVPR (1)*, pages 556–564, 2003.

[53] O. Veksler. Stereo correspondence by dynamic programming on a tree. In *CVPR(2)*, pages 384–390, Washington, DC, USA, 2005. IEEE Computer Society.

[54] L. Wang, M. Liao, M. Gong, R. Yang, and D. Nistér. High-quality real-time stereo using adaptive cost aggregation and dynamic programming. In *3DPVT*, 2006.

[55] T. Werner and A. Zisserman. New techniques for automated architecture reconstruction from photographs. In *ECCV*, volume 2, pages 541–555, 2002.

[56] Q. Yang, L. Wang, R. Yang, H. Stewénius, and D. Nistér. Stereo matching with color-weighted correlation, hierarchical belief propagation and occlusion handling. In *CVPR (2)*, pages 2347–2354, 2006.

[57] Q. Yang, L. Wang, R. Yang, S. Wang, M. Liao, and D. Nistér. Real-time global stereo matching using hierarchical belief propagation. In *BMVC*, pages 989–998, 2006.

[58] R. Yang and M. Pollefeys. A versatile stereo implementation on commodity graphics hardware. *Journal of Real-Time Imaging*, 11(1):7–18, 2005.

[59] K.-J. Yoon and I.-S. Kweon. Adaptive support-weight approach for correspondence search. *PAMI*, 28(4):650– 656, 2006.

Vita

Author's Name:	Qingxiong Yang
Date and Place of Birth:	Quanzhou, Fujian, China September 30, 1981
Education:	Bachelor of Engineering, special subject: Electrical Engineering University of Science and Technology of China, 2004
Professional Positions:	Graduate Research Assistant University of Kentucky Lexington, Kentucky, 2005–2007
Honors:	Research Assistantship University of Kentucky, 2005–2007

Professional Publications:

“**Q. Yang**, R. Yang, J. Davis, D. Nistér, Spatial-Depth Super Resolution for Range Images, ”In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2007 (acceptance rate: 27.5%).

“D. Gallup, J.-M Frahm, P. Mordohai, **Q. Yang** AND M. Pollefeys, Planesweeping Stereo with Multiple Sweeping Directions, ”In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2007 (acceptance rate: 27.5%).

“P. Mordohai, J.-M. Frahm, A. Akbarzadeh, B. Clipp, C. Engels, D. Gallup, P. Merrell, C. Salmi, S. Sinha, B. Talton, L. Wang, **Q. Yang**, H. Stewénius, H. Towles, G. Welch, R. Yang, M. Pollefeys and D. Nistér, Real-Time Video-Based Reconstruction of Urban Environments, ”In *ISPRS International Workshop ”3D-ARCH'07”* , 2007.

“S. J. Kim, D. Gallup, J.-M. Frahm, A. Akbarzadeh, **Q. Yang**, R. Yang, D. Nistér and M. Pollefeys, Gain Adaptive Real-Time Stereo Streaming, ”In *International Conference on Computer Vision Systems (ICVS)*, 2007.

“A. Akbarzadeh, J.-M. Frahm, P. Mordohai, B. Clipp, C. Engels, D. Gallup, P. Merrell, M. Phelps, S. Sinha, B. Talton, L. Wang, **Q. Yang**, H. Stewéius, R. Yang, G. Welch, H. Towles, D. Nisté and M. Pollefeys, Towards Urban 3D Reconstruction From Video, ”In *International Symposium on 3D Data Processing Visualization and Transmission (3DPVT)*, **Invited paper**, 2006.

“**Q. Yang**, L. Wang, R. Yang, S. Wang, M. Liao and D. Nistér, Real-time Global Stereo Matching Using Hierarchical Belief Propagation, ”In *British Machine Vision Conference (BMVC)*, 2006 (acceptance rate: 25%).

“**Q. Yang**, L. Wang, R. Yang, H. Stewénius and D. Nistér, Stereo Matching with Color-Weighted Correlation, Hierarchical Belief Propagation and Occlusion Handling, ”In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2006 (acceptance rate: 28.1%).

“**Q. Yang**, M. Steele, D. Nistér and C. Jaynes, Learning the Probability of Correspondences without Ground Truth,”In *IEEE International Conference on Computer Vision (ICCV)*, 2005 (acceptance rate: 20.3%).

(Qingxiong Yang)