IMPROVED DEPTH MAP UPSAMPLING USING ITERATIVE JOINT TRILATERAL FILTER

by

Yuksel Karahan

A thesis submitted to the Faculty of the University of Delaware in partial fulfillment of the requirements for the degree of Master of Science in Electrical & Computer Engineering

Winter 2018

IMPROVED DEPTH MAP UPSAMPLING

USING ITERATIVE JOINT TRILATERAL FILTER

by

Yuksel Karahan

Approved: _____

Gonzalo R. Arce, Ph.D.
Professor in charge of thesis on behalf of the Advisory Committee

Approved: _____

Kenneth E. Barner, Ph.D.
Chair of the Department of Electrical and Computer Engineering

Approved: _____

Babatunde A. Ogunnaike, Ph.D.
Dean of the College Engineering

Approved: _____

Ann L. Ardis, Ph.D.
Senior Vice Provost for Graduate and Professional Education

ACKNOWLEDGMENTS

I would like to express my gratitude to my advisor, Dr. Gonzalo R. Arce. I want to express appreciation for giving me the opportunity to be part of his research group. Special thanks to the Turkish Educational Ministry that has been supporting me during the study. Also, thanks to Juan Florez for taking the UD depth dataset measurements used in this research and for his helpful comments. Cenk Demir, thanks for helping me run the experiment. Carlos Mendoza, thanks for your advice and for acting as an instructor to me. Thanks also to Ria and Sara for their proofreading as well. Thanks to Hoover, Claudia, Abdullah, Mariano, Edgar, Xiao, Chen, Juan Becerra, Alejandro, Angela, Karelia, and Laura, a group of amazing people.

Last but not the least, my thanks also go to my mother, father, sisters, and brothers. Without them, I would not have come this far. Thanks go out to Gina Puentes and her family for their moral support. The unique appreciation goes to my princess, little angel Sara Lorin who is my source of motivation. The more can be said for my friends Mehmet Emin Akdogan and Jan Sher Khan who have supported and motivated me all the way. Finally, I would like to give my sincerest apologies for all those helping hands, whose names I may have missed unintentionally.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

Typical depth sensor cameras have a lower resolution compared to RGB cameras. In this work, inspired by the Joint Trilateral Filter (JTF), an iterative JTF (JTFite) approach is presented to improve the visual quality of the depth discontinuity regions. Our framework takes into consideration the utilization of the difference of refined pixels' values and the canny edge of the JTF, to obtain sharper edges. The recent work on JTF is revisited by setting varying parameters which employs local gradients information to avoid texture artifacts in a structure-aware onion filtering. We extend the JTF by using traits of its local filtering technique to improve further the edges of the objects in the depth map by post-filtering the updated region. The proposed method enhances details of the textures in the depth map by emphasizing on gradients of the drastic changes of the pixels values. The primary goal of this work is to increase the depth resolution by a high-resolution RGB image to preserve sharp edges and details of the textures. Visual and quantitative experimental results show that the proposed method outperforms the JTF for depth map upsampling. We evaluate the upsampled and filtered low-resolution (LR) depth maps with their associated ground truth (GT) images.

Chapter 1

INTRODUCTION

Depth maps have been playing an important task in entertainment, scientific research, and medical imaging areas, and so on. Time-of-flight (ToF) camera, an active-light range sensor, is the most well-known, which performs considerably well and measures the depth at a high frame rate. ToF cameras (e.g., Microsoft Kinect) use active sensing to measure the phase delay of the reflected near-infrared wave from a scene to its light source to calculate the distance. The ToF depth camera, however, is sensitive to illumination and noise; hence, it performs deficiently on rich-textured scenes. Additionally, the typical sensor size of depth camera is limited compared to an RGB camera; therefore, the captured data has low-resolution (LR) which leads to poor compositions at objects' boundaries. Thereupon, by taking into consideration the depth map sensors and their application areas, researchers seek to improve the visual quality of depth images.

The problem addressed above can be solved by upsampling the given low-resolution map or by employing a high-resolution (HR) color image using its property to meet particular applications. The most popular upsampling techniques increase the depth maps' resolution, such as, for example, bicubic or bilinear interpolation; however, while the interpolated version is a good estimator of super-resolution (SR), those upsampling methods inescapably smooth the textures and blur the edges.

Recently, some proposals have focused mainly on image filtering techniques for depth map upsampling, such as the Bilateral Filter (BF) or variations. BF is a non-linear filtering method proposed by Tomasi and Madchuni [2], which smooths the images while preserving edges. One of the most famous fast filtering methods proposed by Kopf et al. [8], is the Joint Bilateral Filter (JBU), which similarly averages

the samples at LR image into HR depth maps to obtain improvement on the edges. Equivalently, the Joint Bilateral Filter (JBF) proposed by Eisemann et al. [4], with the support of a high resolution guided flash image, assumes the scene is motionless. These approaches utilize LR depth input with a guidance color image and expect that the HR image is registered and both depth and guided images possess similar discontinuities. However, related work [2] fails when the guidance image is dark or noisy, and texture artifacts appear in [8], [4] when the same depth plane exhibits rich color texture or different depth regions show similar colors.

Lo et al. [1] proposed the Joint Trilateral Filter (JTF) to address texture artifacts by employing local gradients on depth maps. Lo applied a binary indicator to a bicubic interpolated depth map to identify depth discontinuities. Later, they applied JTF to the defined discontinuities in a structure-aware onion peel filtering to refine object edges. Although adopting a registered HR guided color image with a local gradient kernel can increase the depth details, the result suffers from distortion when objects in the depth map exhibit sharp corners, similar range in depth border, and rich clustered textures.

In this work, we present an iterative JTF (JTFite) to obtain sharper edges by post-filtering the JTF result. We iterate the refined edges using the union of the Canny edge of the JTF and the gradient of refined pixels' differences. The proposed method is inspired by the traits of the local structure of the JTF to solve distortion of defined edges. We observe the pixels' values after each iteration to track changes before and after refining. The changes, naturally, appear in depth discontinuity regions. Since the JTF does not update the pixels equally, some filtered regions exhibit drastic changes. We applied a local gradient to the refined pixels' differences which reveals a slope of every pixel. The most considerable changes of pixels assign the direction of the gradient. Henceforth, we unite gradient-applied regions with the Canny edge of JTF to constrain the update range. JTFite reconstructs depth map edges considerably well when two adjacent pixels exhibit similar depth but different color, where JTF fails. Similarly, JTFite outperforms JTF when two adjacent pixels show similar color but

different depth planes.

This work shows that the proposed framework preserves details in the depth map and outperforms JTF to some extent. The goal of this paper is to produce quality HR depth maps out from the LR depth map using local filtering features highlighted in the JTF technique. The remainder of the paper is organized as follows: Section II briefly describes related algorithms of the bilateral filter and extensive details about JTF. Section III features the framework of binary region identification and structure-aware onion peel filtering. In section IV, the proposed method JTFite is presented. Section V is the experimental results and evaluations. Section VI is the conclusion of the work.

Chapter 2

BILATERAL FILTERING TECHNIQUE AND ITS EXTENSIONS

The Bilateral Filter is an advanced smoothing and edge-preserving image filtering technique introduced by Tomasi and Madchuni [2] in 1998. The bilateral filter is widely used in image processing, and it has been improved in many of its extensions such as the Joint Bilateral Filter, Joint Bilateral Upsampling, and Joint Trilateral Filter. All of the extensions are well developed for specific imaging filtering tasks. A few of them and their theoretical and mathematical explanations are presented here.

2.1 Bilateral Filter

The Bilateral Filter, introduced by Tomasi and Madchuni [2], is a widely used technique that smooths an image while preserving edges, and the filter is broadly applied in images to reduce the noise. Since it is not an iterative filtering method, BF contains nonlinear weights of nearby pixel values. Those summations of weights are delivered as a filtered result. The BF can be expressed as follows:

$$\tilde{I}_p = \frac{\sum_{q \in \Omega} f(||p-q||).g(||I_p - I_q||).I_q}{\sum_{q \in \Omega} f(||p-q||).g(||I_p - I_q||)} \tag{2.1}$$

In equation (2.1) p and q represent the pixel indices where p is the center and q indicates its neighbor pixel under the spatial support Ω. A window w appoints the spatial support by its size and restricts q in this frame. $\tilde{I}_p$ is the filtered output at the pixel p. $f(||p-q||)$ and $g(||I_p - I_q||)$ are two Gaussian kernels where f(.) is a spatial and g(.) is a range kernel. The spatial kernel is applied to measure the Euclidean distance from the center pixel, and the range is applied to describe the difference of intensity

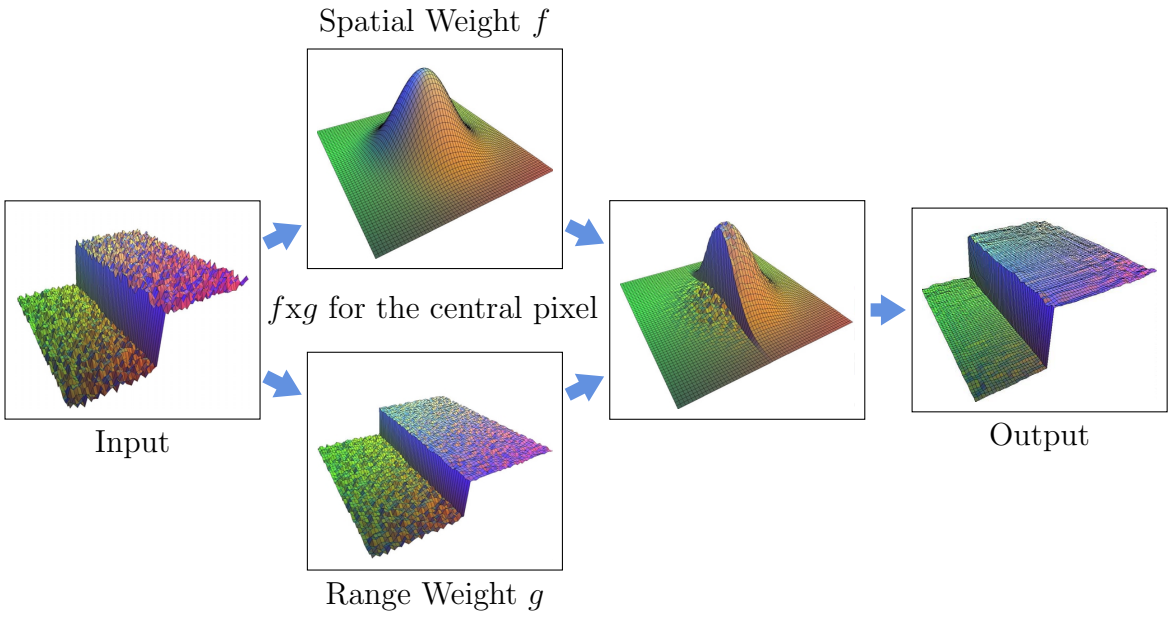


Figure 2.1: The bilateral filter smooths the input image while preserving its edges. Each pixel is updated by an average weighted of its adjacent pixels. Weights are calculated by a spatial and range kernel under a spatial support. The figure is reproduced from [23]. HTTP://doi.acm.org/10.1145/566570.566574

values between p and neighboring pixels q. Hence, the weights are not only assigned by the geometric distance but also with the intensity closeness. The f(.) and g(.) in equation (2.1) defined in a Gaussian form

$$f(||p-q||) = e^{-(||p-q||)^2/2\sigma_f^2} \quad \text{and} \quad g(||I_p - I_q||) = e^{-(||I_p-I_q||)^2/2\sigma_g^2},$$

where σ_f and σ_g are spatial domain and intensity domain standard deviations respectively. These parameters are dominating the fall-off of weights in the spectrum and characterizing their adjacencies.

The denominator of (2.1) is called the normalization factor where it normalizes the appointed weights accordingly, defined as follows:

$$k_p = \sum_{q \in \Omega} f(||p-q||).g(||I_p - I_q||) \tag{2.2}$$

The weights that contributed to all the pixels are displayed as normalized constant k_p.

• BF is nonlinear, and its formulation is simple [3]. The averaged weight of its neighbors replaces the emphasized pixel.

• The basic idea behind its simplicity is that there are only two parameters to preserve the features: size and intensity.

• BF is a standard domain filtering, not an iterative method; therefore, it is easy to set the parameters.

• It is efficient and fast when the computational time is crucial.

2.2 Joint Bilateral Filter

The Joint Bilateral Filter is an advanced edge-preserving image filtering technique introduced by Eisemann [4] and Petschnigg [6] that prevents image smoothing. It may be known as the Cross Bilateral Filter because it mainly decouples edge preserving and image smoothing. The technique is using a reference image $\hat{I}$ instead of input image I. The reference input $\tilde{I}$ is an image that is captured from a different source. The JBF, typically, assigns weights respect to pixel location and its color texture. Another way of saying this is that neighbor pixels get lower weight with greater geometric distance and higher photometric difference. The JBF can be expressed as follows:

$$\tilde{I}_p = \frac{\sum_{q \in \Omega} f(||p-q||).g(||\hat{I}_p - \hat{I}_q||).I_q}{\sum_{q \in \Omega} f(||p-q||).g(||\hat{I}_p - \hat{I}_q||)} \tag{2.3}$$

The new kernel $g(||\hat{I}_p - \hat{I}_q||)$ is utilized to improve the quality of input where $\hat{I}$ is an image of the scene well illuminated. In equation (2.3) p is the center of the kernel, and q is neighboring pixels under the spatial support Ω. The input image I and the geometric distance kernel $f(||p-q||)$ are the same that used in BF. $\hat{I}_p$ and $\hat{I}_q$ are the pixel intensities at the center pixel p and its neighbor q respectively where $\hat{I}$ is a

displacement of I. The denominator of (2.3) is normalization factor where the second kernel is utilized from the reference input and as follows:

$$k_p = \sum_{q \in \Omega} f(||p - q||).g(||\hat{I}_p - \hat{I}_q||).$$

It can be seen that the JBF technique depends on a high contrast and scene illumination. In the case of a dark and noisy scene, the contribution from the second kernel $g(||I_p - I_q||)$ to the BF is going to be inadequate due to the lack of intensity range. Fundamentally, the JBF technique [5] aims to associate two input images: one captured with a flashlight [6] and another one out of light, to obtain an HR output image.

•JBF performs well-removing noise [7]; therefore, the filter ensures better visual quality and outperforms Gaussian noise removal techniques regarding the PSNR comparison.

• The JBF performs considerably well when the noise is high.

2.3 Joint Bilateral Upsampling

The Joint Bilateral Upsampling algorithm is an extension of the Bilateral Filter introduced by Kopf et al. [8] that uses a low-resolution kernel instead of an upsampled version. JBU reconstructs an HR output where JBF upsamples the input instead. Given an HR color input, in-depth upsampling, JBU computes two Gaussian kernels where the spatial filter is a truncated Gaussian kernel and the range filter from HR image to upsample the LR depth input sparsely. Therefore, the technique is performing well at preserving high frequencies.

The upsampled image $\tilde{S}$ is computed as follows:

$$\tilde{S}_p = \frac{\sum_{q_\downarrow \in \Omega} f(||p_\downarrow - q_\downarrow||).g(||\hat{I}_p - \hat{I}_q||).S_{q_\downarrow}}{\sum_{q_\downarrow \in \Omega} f(||p_\downarrow - q_\downarrow||).g(||\hat{I}_p - \hat{I}_q||)} \tag{2.4}$$

In equation(2.4) p and q denote locations of pixels in HR guidance image $\hat{I}$, and $p_\downarrow$ and $q_\downarrow$ indicate the corresponding locations in the input LR image S. Although,

the above equation is almost same compared to BF, the geometric distance kernel $f(||p_{\downarrow} - q_{\downarrow}||)$ is set at LR input image S where the range kernel $g(||\hat{I}_p - \hat{I}_q||)$ is set at HR input image $\hat{I}$. Both kernels are Gaussian and can be expressed the same in the BF where the spatial distance kernel $f(.)$ is expressed respect to its pixel locations

$$f(||p_{\downarrow} - q_{\downarrow}||) = e^{-\left(||p_{\downarrow} - q_{\downarrow}||\right)^2 / 2\sigma_f^2}.$$

The denominator of (2.4) is a normalization factor where the first kernel is employed from LR input image. It can be written explicitly as follows:

$$k_p = \sum_{q_{\downarrow} \in \Omega} f(||p_{\downarrow} - q_{\downarrow}||).g(||\hat{I}_p - \hat{I}_q||).$$

JBU, a method inspired by BF, has a powerful ability to upsample hues for colorization and exposure map for tone mapping [9]. As BF does, the technique is similarly averaging the samples at LR image that the weights calculated by neighboring pixels where the contribution varies within the spatial distance and color difference. Kopf's JBU technique upsamples the LR image, considering depth discontinuities locally which makes it a fast filtering method.

• JBU obtains an HR $\tilde{S}$ depth result that can preserve edges that are consistent with the HR input image $\hat{I}$ [11]. In other words, the upsampled image $\tilde{S}$ is inclusive of the high-frequency components of the HR guidance image $\hat{I}$.

• The guidance image is sampled sparsely during the filtering process; therefore, it is independent of an upsampling factor.

• Since the filtering process takes into account a small spatial footprint; the JBU can be computed fast. It also applied to gigapixel images by Kopf et al. [16] to upsample tone mapping solutions.

2.4 The Trilateral Filter

The first Trilateral Filter introduced by P.Choudhury et al. [12] in 2003, is an extension of the Bilateral Filter for edge-preserving smoothing operations. The main aim is to combine a gradient and intensity BF with a pyramid-based method to limit

filter range. It starts with BF application on gradients of the input image to determine discontinued image regions. Authors define slopes on those discontinuities and tilt the BF before applying to the intensity of the input image. With the tilting process, the range kernel is remodeled as a spatial filter [9]. Conclusively, the output pixel's range is limited by the tilted BF where the filtering process focuses on a set of pixels that share similar gradient values. Overall, the trilateral filter addresses three contributions:

(i) **Tilting**

The filtering window is tilted by G_θ to address high-gradient regions. The gradients of the input can be formulated as follows:

$$G_\theta(x) = \frac{1}{k_\theta(x)} \int_{\infty}^{\infty} \bigtriangledown I_{in}(x+\zeta) c(\zeta) s(|| \bigtriangledown I_{in}(x+\zeta) - \bigtriangledown I_{in}(x)||)\, d\zeta \tag{2.5}$$

The normalization factor $k_\theta(x)$ is defined as

$$k_\theta(x) = \int_{\infty}^{\infty} c(\zeta) s(|| \bigtriangledown I_{in}(x+\zeta) - \bigtriangledown I_{in}(x)||)\, d\zeta. \tag{2.6}$$

The domain kernel c(.) is a Gaussian function, a location-dependent weight provider, and ζ is an offset vector determine the positions around x. I_{in} is the input data; the $G_\theta(x)$ is tilted vector output where s(.) is a range filter. To find the modified local values, authors define: $P(x,\zeta) = I_{in}(x) + G_\theta.\zeta$, and by substucting from I_{in}, they gets $I_\triangle(x,\zeta) = I_{in}(x+\zeta) - P(x,\zeta)$. Finally, adding s(.), c(.) and $I_\triangle(\zeta)$ together, the output $I_{out}(x)$ can be expressed as follows:

$$I_{out}(x) = I_{in}(x) + \frac{1}{k_\triangle(x)} \int_{\infty}^{\infty} I_\triangle(x+\zeta) c(\zeta) s(I_\triangle(x,\zeta)) f_\theta(x,\zeta)\, d\zeta. \tag{2.7}$$

$f_\theta(x,\zeta) = [0,1]$ and local weights are normalized by $k_\triangle(x)$:

$$k_\triangle(x) = \int_{\infty}^{\infty} c(\zeta) s(I_\triangle(x,\zeta)) f_\theta(x,\zeta)\, d\zeta.$$

(ii) **Automatic f_θ:Adaptive Neighborhood**

By applying a threshold R to obtain a binary form, the authors aim to limit the smoothed adjacent points connected to x by f_θ.

$$f_\theta(x,\zeta) = \begin{cases} 1 & if||G_\theta(x+\zeta) - G_\theta(x)|| < R \\ 0 & \text{otherwise} \end{cases} \tag{2.8}$$

(iii) **Self-Adjusting Parameters**

By the third contribution, the authors aim to avoid hand-tuned parameters to improve reconstruction by refraining generalization of parameters. Put another way, the user sets an initial parameter $\sigma_{c\theta}$ where the neighborhood size is assigned automatically to smooth the regions in the output image. This parameter controls seven other internal ones

$$\sigma_{s\theta} = \beta(||max(G_{avg}(x) - min(G_{avg}(x))||), \tag{2.9}$$

and β is set empirically.

- The highlighted features of Trilateral Filter is that the method smooths the image while preserving the edges which allows a fragmentary gradient approximation [12].
- Self-Adjusting Parameters allows one to use only one parameter, which makes the method almost a parameter-free technique.
- The trilateral filter has the ability to abstract noise from image details.

2.5 Joint Trilateral Upsampling

Inspired by Joint Bilateral Filter, Li et al.[11] further presented the Joint Trilateral Upsampling, which takes into account an LR input depth image, a corresponding HR color image, and an HR depth guidance map that are generated by the LR depth map. Li et al. proposed the JTU algorithm that works iteratively to overcome texture copying artifacts that are caused by high-intensity differences between neighboring pixels. If there is a high weight set to a neighbor pixel, then one can conclude that the

intensity difference will be high, likewise, if there is low weight set to a neighbor, the filtering process will generate a smooth result.

The upsampled image $\tilde{S}$ is computed as follows:

$$\tilde{S_p} = \frac{\sum\limits_{q_\downarrow \in \Omega} f(||p_\downarrow - q_\downarrow||).g(||\hat{I}_p - \hat{I}_q||).h(||\tilde{I}_p - \tilde{I}_q||).S_{q_\downarrow}}{\sum\limits_{q_\downarrow \in \Omega} f(||p_\downarrow - q_\downarrow||).g(||\hat{I}_p - \hat{I}_q||).h(||\tilde{I}_p - \tilde{I}_q||)} \tag{2.10}$$

The new kernel $h(||\tilde{I}_p - \tilde{I}_q||)$ is utilized to improve the quality of output to preserve the edges and avoiding texture copying artifacts, where $\tilde{I}$ is an HR depth map, generated from the input LR image to preserve edge information. $\hat{I}$ and S denote the intensity of HR color guidance image and LR input depth image respectively, where $\tilde{S}$ is the JTU output. p, q, $p_\downarrow$ and $q_\downarrow$ denote the locations of the pixels where they have the same usage as so in the JBU.

Although the equation is almost the same compared to JBU, the addition Gaussian depth filter kernel h(.) is set at HR input depth map, and defined as:

$$h(||\tilde{I}_p - \tilde{I}_q||) = e^{-\left(||\tilde{I}_p - \tilde{I}_q||\right)^2 / 2\sigma_h^2}. \tag{2.11}$$

The denominator of (2.10) is normalization factor where the third kernel h(.) is employed on HR depth input. It can be decomposed discretely as follows:

$$k_p = \sum_{q_\downarrow \in \Omega} f(||p_\downarrow - q_\downarrow||).g(||\hat{I}_p - \hat{I}_q||).h(||\tilde{I}_p - \tilde{I}_q||)$$

Authors aim to take advantage of using an additional kernel to avoid a direct use of LR depth in the upsampling process, where the kernel filter is generated on HR depth, utilizing the edge contents as well as reducing the noise. The function of the third kernel is preserving the edge information that HR guidance image $\hat{I}$ contains.

- The highlighted features of JTU is that the method obtains a Gaussian kernel h(.) that preserves edges consistent with the HR input image $\hat{I}$ [11]. In other words, the JTU is inclusive of the high-frequency components of the HR guidance image by converging at the edge iteratively.

- JTU avoids texture copying artifact that caused by intensity differences.

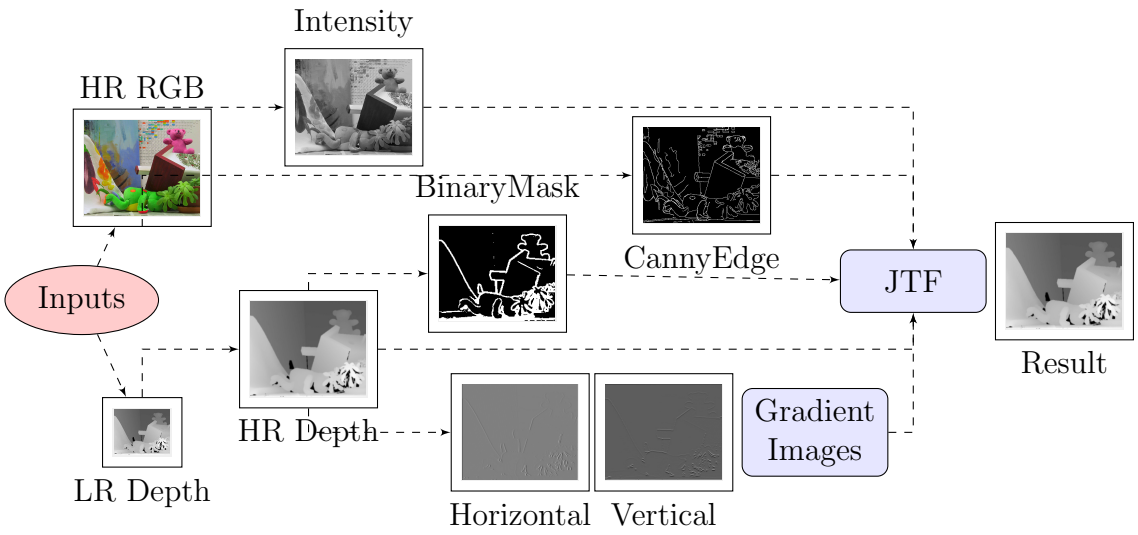


Figure 2.2: Flowchart of Joint Trilateral Filter [1]. The JTF smooths an input image while preserving its edges.

2.6 Joint Trilateral Filter

Inspired by Joint Bilateral Filter, Lo et al. [1] further introduced the Joint Trilateral Filter(JTF), which considers an LR input depth image and a registered HR color image. The authors' proposed method aims to overcome color inconsistency that occurs between HR color image and a depth map. The inconsistency can appear under two circumstances: when two adjacent pixels have the same color but different depth values and when those two neighbor pixels are in the same depth but have different colors. As mentioned earlier by Li et al. [11] these inconsistencies will lead the artifacts because of large intensity differences between neighboring pixels. The objective of Lo et al. [1] is to solve the second case inconsistency by the JTF algorithm that as follows:

$$\tilde{S_p} = \frac{\sum\limits_{q\in\Omega} f(||p\_q||).g(||\hat{I}_p - \hat{I}_q||).d(||G_p - G_q||).M(q).S_q}{\sum\limits_{q\in\Omega} f(||p-q||).g(||\hat{I}_p - \hat{I}_q||).d(||G_p - G_q||).M(q)} \tag{2.12}$$

Recall the JBF weighting; the JTF assigns weights respect to pixel location and its color texture. In other words, neighbor pixels get lower weight with greater geometric distances and larger intensity differences. JTF is an extension of the JBF to bypass the

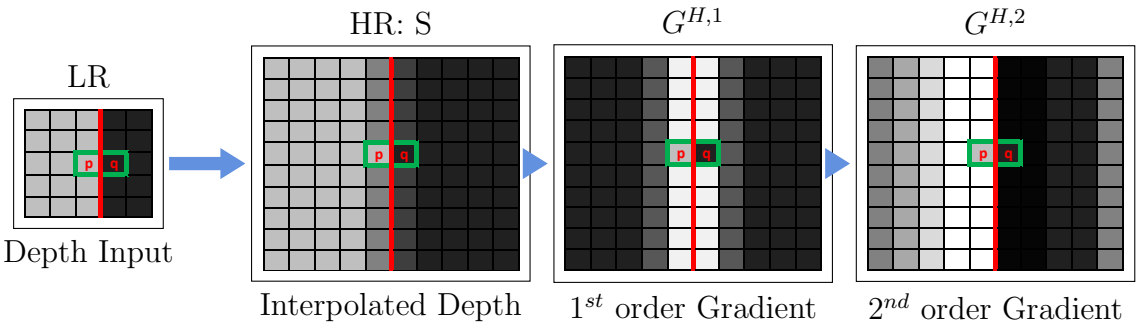


Figure 2.3: Even though, two adjacent pixels p and p have similar value at 1^{st} order gradient, they would show different values at 2^{nd} order gradient which allows us a better filtering. Adapted from [1].

gradient reversal artifacts occurring. The filtering process of JTF is firstly done under the guidance image $\hat{I}$, which is a color one, and the interpolation of LR input depth map S. The pixel locations that are denoted by p and q have the same usage as in the JBF, where f(.) and g(.) have been employed similarly. The last term M is a binary mask that generated by the *Identication of Depth Discontinuity Regions*(IDDR) algorithm. Identification of depth map region will be in the next chapter. Essentially, M decides if the neighbor pixel is reliable or not;

$$\text{where} \quad M(q) = 1 - P_{unreliable} \tag{2.13}$$

$\tilde{S}_p$ is the filter output, and the denominator of the JTF is a normalization factor where the third kernel d(.) and M are also employed to weight the local pixels. It can be decomposed discretely as follows:

$$k_p = \sum_{q \in \Omega} f(||p-q||).g(||\hat{I}_p - \hat{I}_q||).d(||G_p - G_q||).M(q).$$

The new term d(.) is used to avoid texture artifacts by performing at depth varying regions. The Gaussian kernel $d(||G_p - G_q||)$ is computing the absolute difference of second-order gradients on central pixel p and its neighboring pixels q. The process

is applied in both horizontal and vertical directions where the weights are assigned by choosing the maximum value. The filter kernel described as:

$$||G_p - G_q|| = max(|G_p^{V,2} - G_q^{V,2}|, |G_p^{H,2} - G_q^{H,2}|) \tag{2.14}$$

$G^{V,2}$ and $G^{H,2}$ are the second-order gradients along vertical and horizontal directions respectively. The process puts two inputs from both directions to choose the maximum value which assigns a weight that contributes filtering respect to p and q values. Particularly, a greater weight will be appointed from neighboring pixels as long as those adjacent pixels share similar values.

The signed second-order derivatives are derived from the first-order of depth map image. The first-order gradients are calculated on bicubic interpolated image S along both directions.

$$G^{V,1}(m,n) = \left| \frac{S(m+1,n) - S(m-1,n)}{2} \right|, \tag{2.15}$$

$$G^{H,1}(m,n) = \left| \frac{S(m,n+1) - S(m,n-1)}{2} \right|. \tag{2.16}$$

$G^{V,1}$ and $G^{H,1}$ denote the first-order gradients along vertical and horizontal directions respectively, and (m,n) denote pixel locations. Even though two adjacent pixels belong to a distinct depth region, they may carry similar gradient values since the first-order gradient is a directional change in the intensity. Therefore, the authors included the second-order one to address local variation in the first-order to reduce the texture copying artifacts effect. Finally, the second order gradients can be expressed respect to $G^{V,1}$ and $G^{H,1}$ as follows:

$$G^{V,2}(m,n) = \frac{G^{V,1}(m+1,n) - G^{V,1}(m-1,n)}{2}, \tag{2.17}$$

$$G^{H,2}(m,n) = \frac{G^{H,1}(m,n+1) - G^{H,1}(m,n-1)}{2}. \tag{2.18}$$

Thus, the authors desire to suppress the artifacts by generating weights from only the neighbor pixels that possess similar depth features, which allows preservation of the rich color texture and edge components.

Chapter 3

A TAXONOMY OF FRAMEWORK ALGORITHMS

With a given LR depth image, for an HR depth estimation, the LR depth map is interpolated via bicubic interpolation. The aim of the work is improving the edge reconstruction and the visual content. Lo et al. [1] proposed the identification of depth discontinuity regions which are based on dividing the image into sunblocks with a dynamic programming. The identified discontinuity regions proceed in an improved onion peel filtering called the Structure-Aware Onion Peel Algorithm.

3.1 Diagnosis of Depth Discontinuity Regions in SR Depth Image

The insufficiency of the sensors of the depth cameras and the environmental effects may cause incomplete scene capturing, which may result in blur on edges or the loss of information in a particular range. The resolution input depth image can be enhanced by upsampling; however, the interpolated image contains blurry edges and smoothed textural regions.

The process of Diagnosis of Depth Discontinuity Regions (DDDR) starts with an application of quadtree decomposition which divides an image into equal-sized sub-blocks; then the decomposition continues testing each macro-block to determine whether it meets the user-defined criteria or not. In our case, initially, the interpolated depth image is divided into non-overlapping 16x16 size sub-blocks. Thereupon, the size of the image should be adjusted by the power of 2 accordingly. The procedure carries forward by computing a calculation of the depth range for each sub-block. The extent is computed by taking the absolute value of the minimum and maximum differences for each block and repeated one by one. The next step is checking whether the range is greater than a predefined threshold τ or not. If the range exhibits a higher value

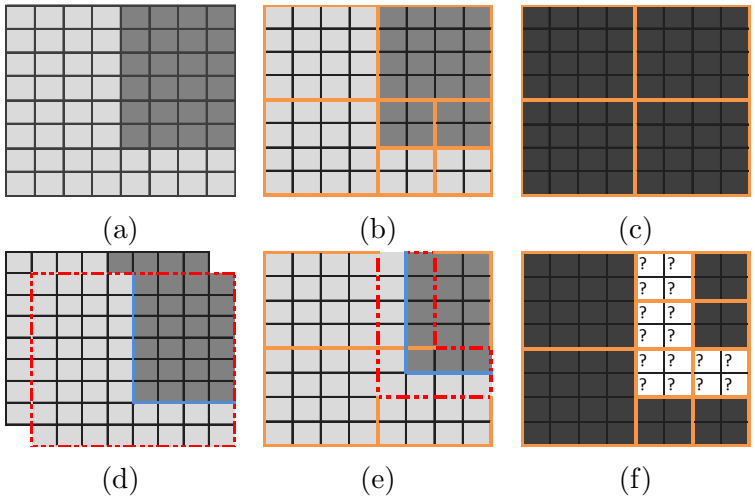


Figure 3.1: An example of the depth discontinuity identification by shifting one pixel. Orange lines show the original sub-block borders. Red dashed frame shows one pixel shifting. (a) The original depth map. (b) Sub-block lines for the original patch. (c)The end of identification for the original patch. (d) One pixel shifted. (e) Sub-block lines for one pixel shifted patch. (f) The identified region in the shifted patch.

than the threshold, the process will further split the sub-block into four macro-blocks with the size 8x8. The process will continue until the smallest block's size reaches the size 2x2. The implementation is executed via dynamic programming which allows one to break down the problem into subproblems. This quadtree decomposition will reveal the depth discontinuity by designating the smallest sub-blocks. If the range in the minimum size of 2x2 is higher than the threshold than the block will be set 1, otherwise 0. Therefore, the Binary Mask that is called *Pedge* constructed with zeros and ones, where ones depict the edges of depth planes.

However, the identification might fail when two adjacent pixels share the different depth values along an object's boundary. On that account, even though there is a distinct depth contrast, if the related pixels are sub-blocked in their depth plane then there is no edge detection(see Fig. 3.1(a, b, c)). In Fig. 3.1(b) and (c), distinct depth pixels do not intervene; therefore, no discontinuity is detected. To solve this problem, as shown in Fig. 3.1(d) and (e), Lo et al. shifted the interpolated depth image by 1 pixel in both vertical and horizontal directions. Similarly, the quadtree decomposition

is applied in both directions, and the identification successfully marked as shown in Fig. 3.1(f). Finally, the union of those masks is the desired one(see Fig. 3.2).

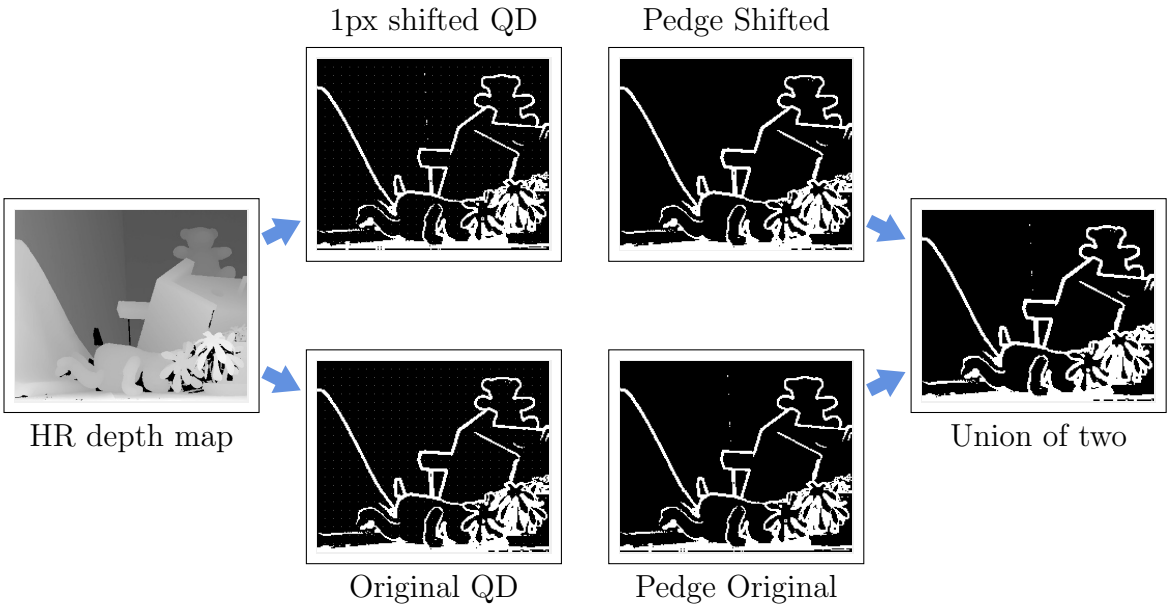


Figure 3.2: An example of the union of depth discontinuity regions. One pixel shifted-depth map is at top line, and the original one is at the bottom. The image on the right side called *Pedge* is the union of the two binary indicators.

3.2 Structure-Aware Onion-Peel Algorithm

An onion-peeling algorithm(OPA) is an approach that peels away a two-dimensional convex set S of n points. Consider $\tilde{S}$ to be the remaining points in the set. The algorithm aims to compute the $\tilde{S}$ and peel away all features until none is remaining. This recursive method is called the onion-peeling of S. Inspired by OPA, the authors present Structure-Aware Onion-Peel Algorithm(SAOPA) to refine the pixels that marked depth regions. SAOPA is an extension of current OPA where the algorithm is including the binary mask Pedge to update the edges of the upsampled depth image.

As explained in the previous section, Binary Mask, *Pedge* (see Fig. 3.2(a)) contains zeros and ones, and only those pixels with ones will be updated. Ones are labeled *unreliable* and zeros are labeled *reliable*.

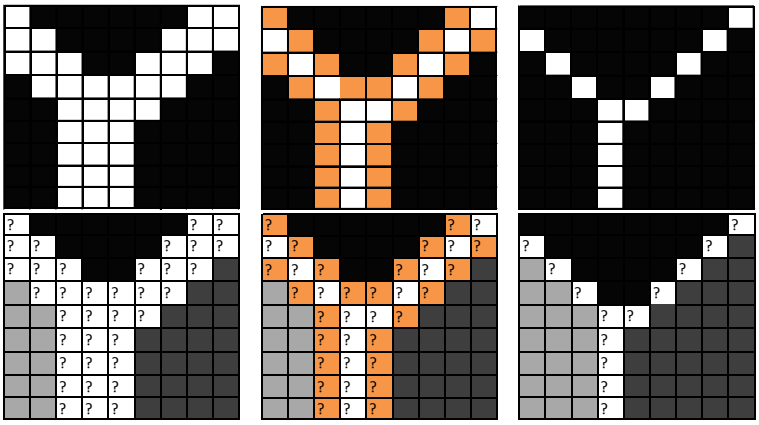
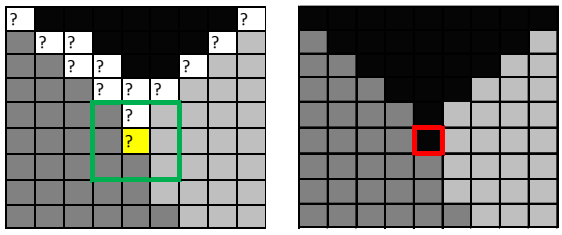


Figure 3.3: Illustration of OPA. The top figures are binary map images where bottom the images are corresponding depth map. The white pixels are depth discontinuity, and orange ones are marked to be filtered. The last column is the filtered ones.

SAOPA starts with initializing $Pedge = Punreliable$ and the JTF filtering will be applied to each pixel that identified as a discontinuity region. For pixel p, if $Punreliable$ =1, the eight neighbors of p will be checked; if there is at least one reliable pixel around p, then the filtering will proceed; otherwise, it will be skipped at this iteration. As a remark, the pixels marked 1 (depth discontinuity region) are unreliable, and the rest (zeros; belong to a distinct depth plane) are reliable.

As illustrated in Fig. 3.3 the identified region is white and filtering part marked(orange). Top figures indicate Binary Mask, Pedge, and bottom ones are the corresponding depth image. The process will scan the whole region; once it completed, the $Punreliable$ and upsampled depth image will be updated.



(a) (b)

Figure 3.4: An example of possible incorrect filtering. (a) The yellow pixel is labeled to be filtered. (b) The ground truth image. It can be seen that the filtering will assign incorrect depth value.

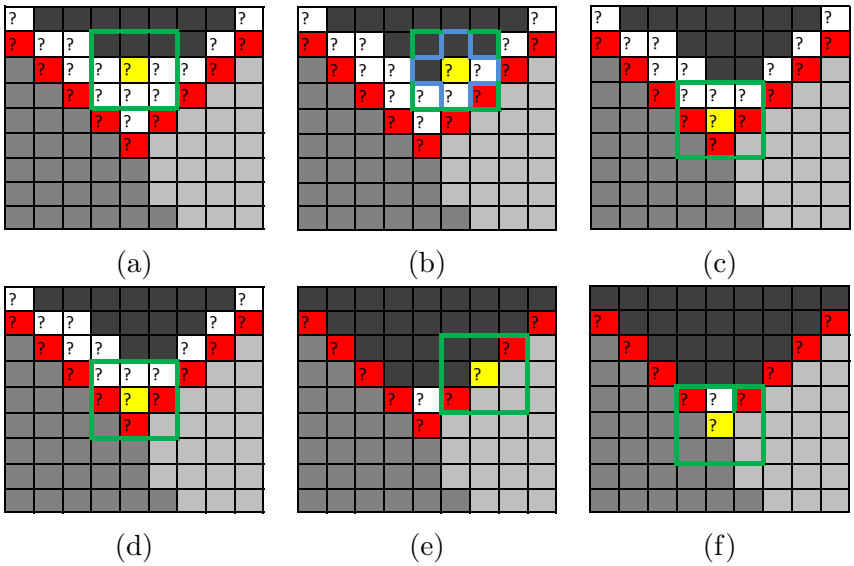


Figure 3.5: This is an illustration of the JTF filtering. Unreliable pixels are marked white, Canny is red, and the current pixel is presented in yellow color. (a) p does not connect to Canny. The filtering will proceed if at least one adjacent pixels is reliable. (b) p is connecting Canny; therefore, four adjacent pixels are checked(blue). (c) p is connecting Canny, but there is no reliable pixel in its neighborhood. (d) The filtering will be skipped at this iteration. (e) p is on the Canny edge. The filtering will proceed if all adjacent pixels are reliable except those are on the Canny edge. (f) The filtering will be skipped since there is an unreliable pixel around. Adapted from [1].

However, the OPA might fail when the refined pixel is part of a specific region, as shown in Fig.3.4. To avoid incorrect filtering, the authors applied a Canny edge detector to three channels of RGB image and then utilized the union of them into the process. An example of the process is illustrated in Fig. 3.5 where the filtering pixel p is shown is yellow color, green square is the window of its neighbor pixels. The unreliable pixels are labeled with a question mark, and the red pixels represent the Canny edge. The filtering process will be classified into three cases for a pixel p if there is at least one reliable pixel in its neighborhood.

- For an unreliable pixel that does not connect and is not on the Canny edge, as shown in Fig. 3.5(a)(the yellow pixel is the target), the filtering will proceed. The pixels without a question mark are reliable and belong to distinct depth region.
- For an unreliable pixel that has at least one reliable pixel around and that

is not on but connecting Canny edge as shown in Fig. 3.5(b) the filtering will have proceeded if one of its four neighbors is reliable, otherwise, it will be skipped(see Fig. 3.5(d)). The blue lines enclose the pixels of interest.

• Finally, an unreliable pixel that is on the Canny edge as shown in Fig. 3.5(e) will be filtered if all its four neighbors are reliable except those on Canny edge, otherwise, it will be skipped as shown in Fig 3.5(f).

The above classification will decrease the number of unreliable pixels significantly. However, the filtering fails when the Canny edge exhibits a loop that is caused by color texture(see Fig. 3.6(e)). The loop in Canny obstructs filtering to refine unreliable pixels. To detect and break the loop, the authors used the magnitude of the first-order gradient of the interpolated depth image and set the lowest point zero. After breaking the loop, as shown in Fig. 3.6(d), the filtering can be completed. The gradient magnitude is calculated as

$$\left|G^1(m,n)\right| = \sqrt{\left(G^{V,1}(m,n) - G^{H,1}(m,n)\right)}, \tag{3.1}$$

where (m,n) are pixel indices. As an example demonstrated in Fig.3.6, one can see that the SAOPA filters *Punreliable* and update all pixels until none is remaining.

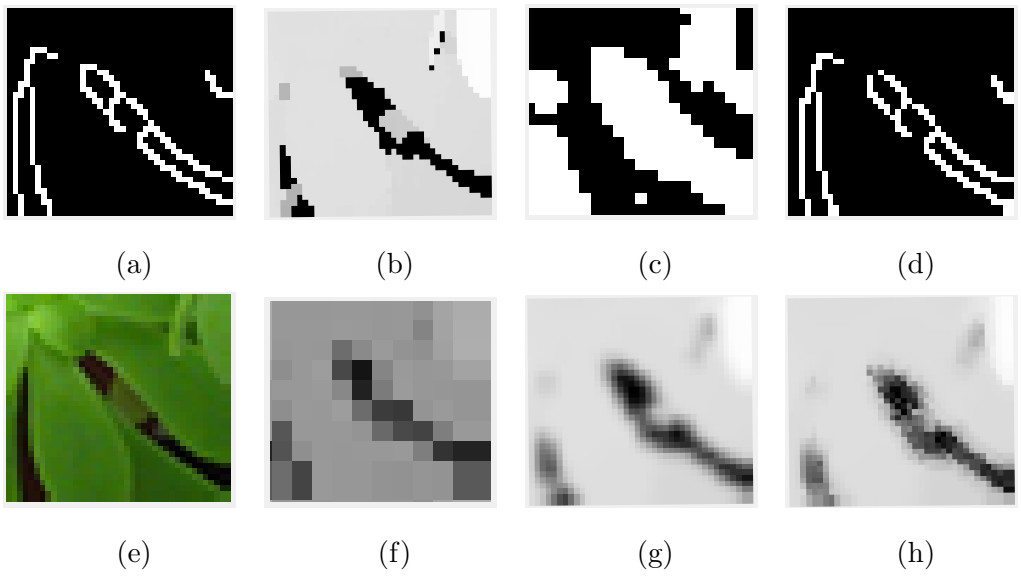


Figure 3.6: This is an example of the Canny edge suspension that prevents filtering. (a)Canny edge loop(Teddy). (b) The corresponding location at the ground truth image. (c)The Binary Mask where the white pixels are unreliable. (d) Breaking of the Canny edge. (e)(f)(g)(h) are corresponding locations of RGB, LR depth map, the Upsampled depth map, and JTF result respectively.

Algorithm 1 Structure-Aware Onion-Peel Filtering [1]

Require: Input: S :interpolated depth map; I: color image; $Punreliable$: binary mask of discontinuity region
Output: $\tilde{S}$: refined depth map
1: Compute 1st-order gradient $G^{V,1}$, $G^{H,1}$ on S Eq. (8)(9)
2: Compute gradient magnitude $G^1(x, y)$ Eq. (6)
3: Compute 2nd-order gradient $G^{V,2}$, $G^{H,2}$ Eq. (10)(11)
4: Apply Canny detector on I to obtain $Canny$ edge-constrained binary map
Ensure: Initialize: $\tilde{S} \leftarrow S$; $\tilde{P}unreliable \leftarrow Punreliable$
while number of unreliable pixel $\neq 0$ **do**
$Punreliable = 1$
reset $count \leftarrow 0$ the number of unreliable pixels refined each iteration
for $i = 1; i \leq imgHeight; i++$ **do**
for $j = 1; j \leq imgWidth; j++$ **do**
if $Punreliable(i, j) = 1$ **then**
if (i, j) is not adjacent to Canny **then**
$S(i, j) \leftarrow$ apply JTF to $S(i, j)$ Eq. (7)
$\tilde{P}unreliable(i, j) \leftarrow 0$; $count++$
else if (i, j) is adjacent to Canny **then**
if at least one of 4-neighbors is reliable **then**
$S(i, j) \Leftarrow$ apply JTF to $S(i, j)$ Eq. (7)
$\tilde{P}unreliable(i, j) \leftarrow 0$; $count++$
else
Continue
end if
else if (i, j) is on Canny **then**
if all eight neighbors are reliable except those on Canny **then**
$S(i, j) \Leftarrow$ apply JTF to $S(i, j)$ Eq. (7)
$\tilde{P}unreliable(i, j) \leftarrow 0$; $count++$
else
Continue
end if
end if
end if
end for
end for
if $count = 0$ **then**
{no refined pixel at this iteration} find the $\min(G^1(x, y))$ in G^1 and set Canny zero at this location
end if
end while

The end of the pseudo-code of structure-aware onion-peel filtering.

Chapter 4

ITERATIVE JOINT TRILATERAL FILTER

Inspired by JTF, an iterative edge-improving Joint Trilateral Filter (JTEite) implemented in this work (see Fig. 4.1). At the expense of computational time, the new approach is re-filtering the output with the union of the Canny edge of the JTF result and the gradient of the difference of updated regions (see Fig. 4.2). In other words, after the first filtering, the difference of refined region in depth map is calculated and saved as $Diff$ (see Fig. 4.2(b)). Naturally, the $Diff$ image will look like the depth discontinuity map. Afterward, the application of gradients on $Diff$ will reveal a slope which is a direction assigned by the refined pixels values. We applied a binary operation on $Diff$ thenceforth to determine the only values that are greater than the default threshold [22]. The filtering concentrates on the more significant changes hereby to

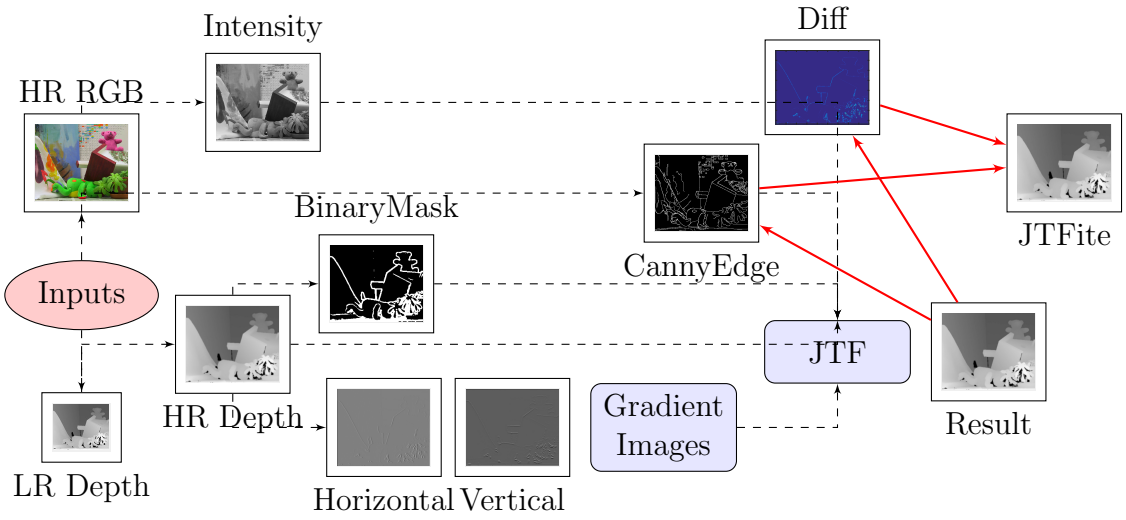


Figure 4.1: Flowchart of Iterative Joint Trilateral Filter. JTFite aims to improve the visual quality by re-updating the refined pixels with respect to their value charges. The iterative process is shown in red arrows.

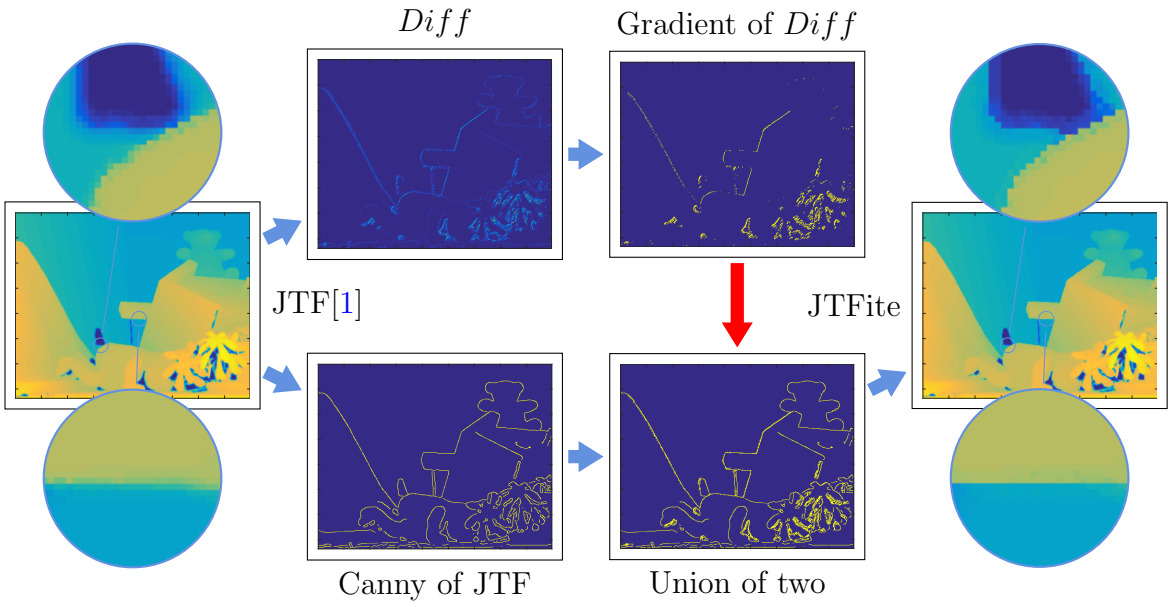


Figure 4.2: Illustration of the Iterative Joint Trilateral Filter(JTFite). It is reconstructed from union of gradient of $Diff$ and Canny edge of JTF result.

assures the JTFite filters a precise interval at the border of depth maps. The $Diff$ Fig. 4.1 displays the flowchart of the JTFite technique for depth map upsampling, and the details of each component will be introduced in this chapter.

Recall that the interpolated depth map contains blur at the edges and has poor textural details. The filtering process will not uniformly update all-region, therefore, at each iteration, the weights will not be ideally assigned because of blur and artifacts. Ideally, along with a depth border, a high contrast is expected at the adjacent pixels which lie on sides of the edge. However, JTF results fail to get a sharp contrast on the edge because the interpolation smooths the border. The transition between depth planes is considerably soft and has a broader length compared to the window size w. To solve this problem, we unite a precise Canny edge of filtered JTF with a gradient of $Diff$ to constrain discontinuity region and to obtain sharper edges.

Fig. 4.4 shows the depth discontinuity regions after each iteration. The JTFite automatically computes the $Diff$ after each iteration to filter remaining regions. It can be seen that the change of pixel values is decreasing by increasing the number of

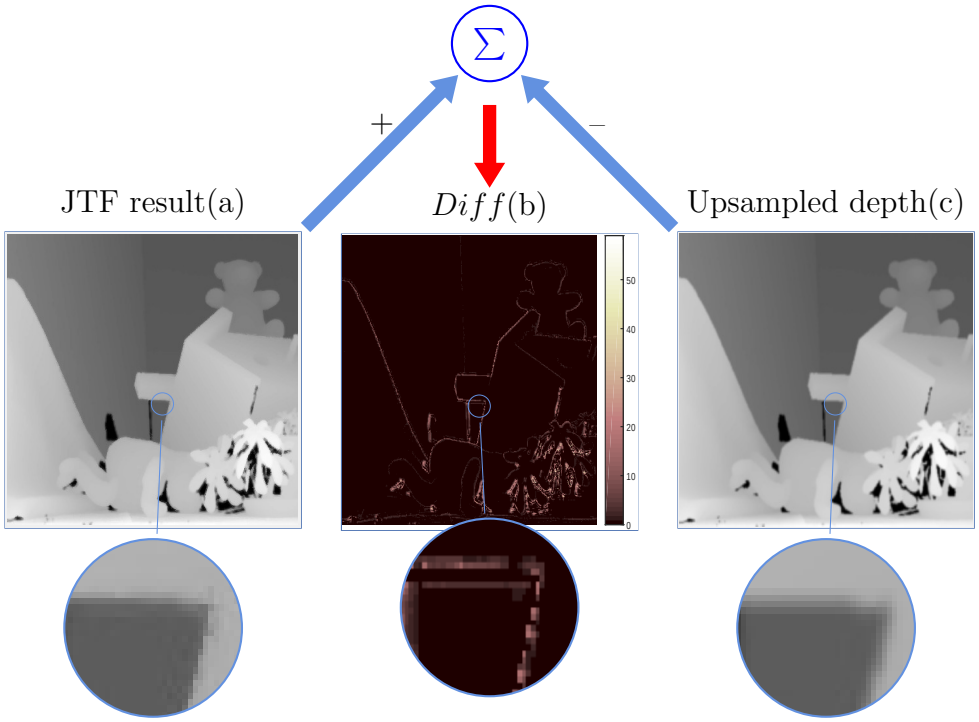


Figure 4.3: Illustration of the Difference of Refined Region($Diff$). We subtract interpolated HR depth from JTF result. Lighter color depicts drastic changes.

iteration. In this experiment, we iterate ten times the JTF result to assure there is a distinct transition between depth planes.

JTF performs well on avoiding texture artifacts by employing local gradients on depth maps. Although their algorithm focuses particularly on the case when two

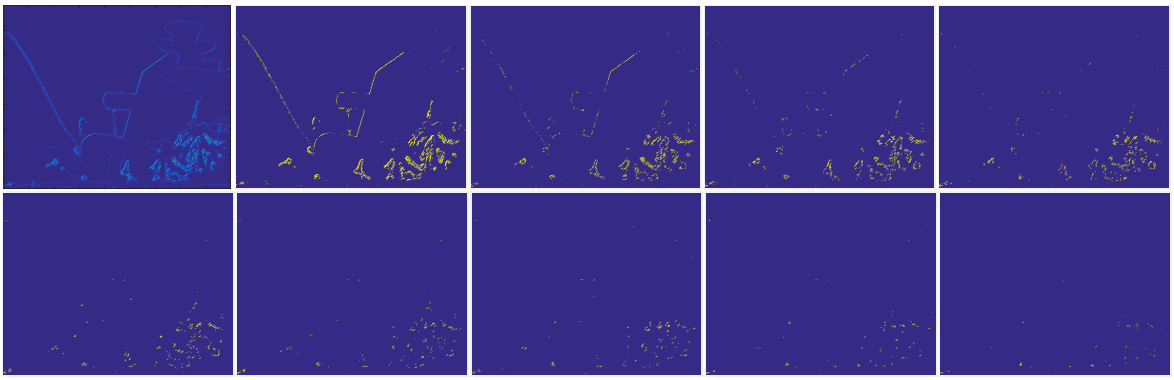


Figure 4.4: Illustration of the of Refined Region After each Iteration. At each iteration the JTFite focuses on the changes of pixel values. The iteration will continue until the generated values are similar to previous input depth map.

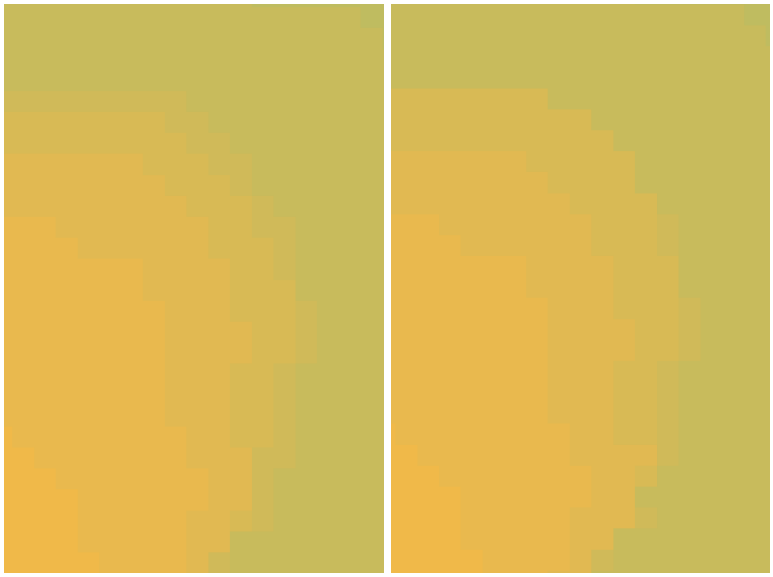


Figure 4.5: Illustration of Filtered Two Near Depth Planes. The left figure is the output of the JTF, and the right one is the result of the JTFite. It can be seen that JTFite pronounces the edge of two near depth planes even they exhibit an approximate depth level.

adjacent pixels present the same color and, different depth plane, JTFite derives higher contrast and precise border at the depth maps. Additionally, JTFite performs better when two adjacent pixels exhibit different colors but similar depth planes. In the experimental section, we also show that JTFite refines the depth planes successfully even when the planes show similar depth values. In other words, our approach obtains sharper edges of similar color depths even though those depth levels are quite near to each other (see Fig. 4.5).

It can be seen that our proposed iterative JTFite obtains sharper edges by post-filtering the JTF result. This work shows that the proposed framework preserves details in the depth map and outperforms JTF to some extent.

Chapter 5

EXPERIMENTAL SETUP, RESULTS, AND DISCUSSION

To evaluate the performance of our upsampling method, we implemented the algorithm on eleven datasets from three sources. In the following sections, the datasets, quality metrics, and visual and quantitative experimental results are presented.

5.1 Data Sets

The proposed algorithm JTFite and JTF are implemented on eleven datasets from Middlebury [14], NYU Depth V2 [20], and UD depth dataset. Images from Middlebury are synthetic where NYU and UD provide real-world data with corresponding color images. The resolutions of experimental unsigned integer 255 type depth images of Middlebury, NYU, and UD, are 368448, 480640 and 236x316 pixels, respectively. In the experiment, NYU and Middlebury HR depth images are down-sampled and than upsampled using two upsampling factor 4X and 8X. However, the UD dataset is only upsampled to its corresponding RGB resolution which is 944x1264. For a fair comparison, same parameters are applied for each dataset. The binary identification region threshold τ is set 8, standard deviations σ_f for f(.), σ_g for g(.), and σ_d for d(.), are set 0.5, 0.3 and 0.5 respectively, also the Canny edge detector's standard deviation σ_{canny} and the threshold are set 1.5 and 0.35.

5.2 Performance Evaluation Metrics

To evaluate the performance of the JTFite and JTF algorithms and the effects of differing its parameters, some quantitative metrics are applied to determine their performance. In this implementation, the following are the quality measures computed dependent upon the known ground truth image.

5.2.1 Bad Pixel Percentage

The Bad Pixel Percentage (BPP) measures the total number of pixels that the absolute value of reconstructed and GT image differences that is greater than a predefined threshold. In other words, any difference greater than the threshold between GT filtered output is considered bad pixels.

$$\text{BPP} = \frac{1}{N} \sum_{x,y} (|d_{RC}(x,y) - d_{GT}(x,y)| > \delta_d) \tag{5.1}$$

N is the total number of pixels where x, y, and δ_d are pixel indices and the threshold respectively. d_{RC} and d_{GT} are JTF result and Ground truth images. In this experiment; $\delta_d = 1.0$, same value used in [1] and [15]. The optimal value of BPP is 0.

5.2.2 MSE

MSE is a cumulative squared error between the reconstructed data and the reference one. This enables us to analyze mathematically which method grants better results.

$$MSE = \frac{1}{m\,n} \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} [d_{RC}(i,j) - d_{GT}(i,j)]^2 \tag{5.2}$$

$m\,n$ are the size of image, height, and width respectively. The optimal value of MSE is 0.

5.2.3 PSNR

PSNR is characterized as the ratio of the image's peak value (signal energy) to the MSE observed between the filtered image and the ground truth. Essentially, it is the assessment of the quality of the images by evaluating maximum possible data value over its cumulative squared error noise that affects the quality of its representation.

$$\text{PSNR} = 10 \log_{10} \left(\frac{(\max\{I\})^2}{\text{MSE}} \right) \tag{5.3}$$

$max\{I\}^2$ is the maximum possible pixel value of the image, where it is equal to 255 for 8 bit image. The higher PSNR, the better the result.

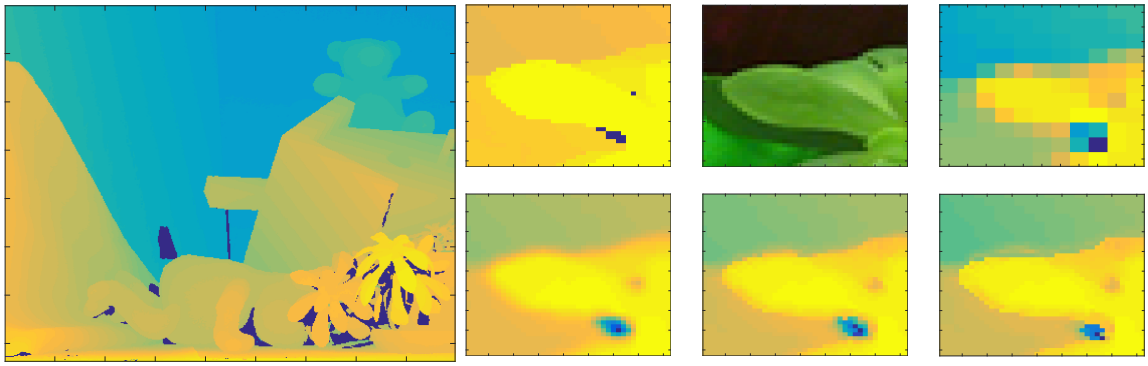


Figure 5.1: This is an example of Teddy where the left image is the ground truth, and right small figures are the zoom version of the corresponding images. The top small figures are GT, RGB, and LR depth map; the small bottom figures are interpolated depth map, JTF result, and JTFite result respectively.

5.2.4 SSIM

The Structural Similarity (SSIM) index is a technique for measuring the similarity between two images [13]. The SSIM index can be described as a measure of one of the images being filtered, and an available image is inferred as a reference. The SSIM is defined by the following formula as proposed by Wang et. al [13],

$$\mathrm{SSIM}(x, y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)}, \tag{5.4}$$

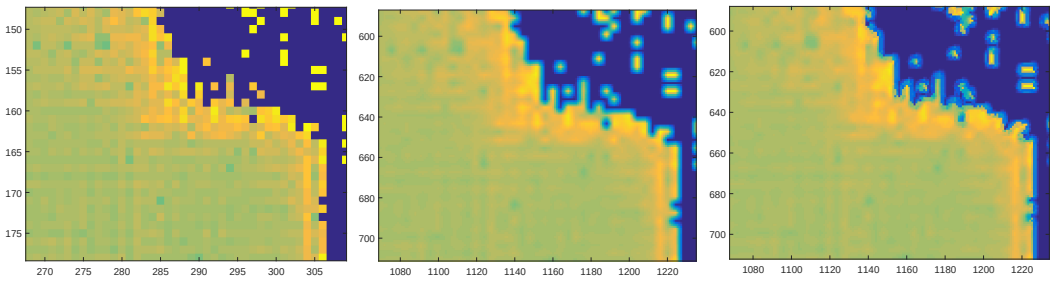
where μ_x , μ_y and σ_x, σ_y are averages and the variances of x or y respectively. σ_{xy} is covariance of x and y, and C1, C2 are constant values to stabilize the division with the weak denominator. In this experiment, default parameters are used. The optimum value of SSIM is 1.

5.3 Experiment and Results

The first step to assess the result is the visual representation. The analysis has been done comparing the filtered output with GT, and its corresponding LR interpolated HR and RGB images. There are also zoom versions of images presented for reader evaluation on objects in the images.

Comparison of BP%, MSE, PSNR, and SSIM with upsampling factor 4X and 8X								
Datasets tho8w3x3	BP%	BP% Ite.	MSE	MSE Ite.	PSNR	PSNR Ite.	SSIM%	SSIM% Ite.
Teddy 4X	2.88	2.80	95.26	104.10	28.34	27.96	95.17	94.81
Cones	4.22	4.16	236.81	242.94	24.39	24.28	87.14	87.17
Venus	0.71	0.72	8.54	8.32	38.82	38.93	98.14	98.19
Dolls	5.15	4.88	37.49	39.14	32.39	32.20	95.67	95.55
Midd2	2.29	2.28	72.84	74.09	29.51	29.43	93.42	93.32
Moebius	4.01	3.84	44.88	47.51	31.61	31.36	95.95	95.72
Reindee	3.73	3.67	81.34	84.25	29.03	28.88	93.54	93.61
Kitchen1	4.94	4.90	3.15	4.28	43.15	41.82	99.02	98.90
Kitchen2	5.62	5.51	2.75	3.12	43.74	43.20	98.87	98.88
Store1	3.26	3.27	2.43	2.41	44.28	44.30	99.01	99.09
Teddy 8X	5.19	5.17	163.34	164.70	26.00	25.96	91.77	91.84
Cones	7.70	7.67	148.67	150.88	26.41	26.34	89.18	89.37
Venus	1.96	1.96	16.80	16.67	35.88	35.91	97.34	97.37
Dolls	9.39	9.37	54.28	53.90	30.78	30.82	93.13	93.34
Midd2	3.89	3.87	49.56	49.65	31.18	31.17	93.64	93.67
Moebius	6.97	6.96	70.30	70.73	29.66	29.63	93.15	93.28
Reindee	6.91	6.87	62.13	61.55	30.20	30.24	93.72	93.95
Kitchen1	13.17	13.15	8.70	9.38	38.73	38.41	97.85	97.78
Kitchen	12.68	12.57	7.86	7.86	39.18	39.18	97.50	97.53
Store1	6.31	6.28	7.70	6.90	39.27	39.74	97.65	97.81

Table 5.1: The experimental results of the JTFite and JTF are presented using the same parameters for $\tau = 8$. Addition to BP% and MSE, PSNR and SSIM of the filtered images are added for further evaluations.



(a) (b) (c)

Figure 5.2: Visual comparison on zoom versions of UD depth dataset. (a) Given LR depth map. (b) Interpolated HR depth map. (c) Proposed method JTFite result.

In Table 5.1 the numeric results of all datasets are presented with different

upsampling. Note that JTFite and JTF results are compared column by column. It can be seen that the JTF results on NYU datasets Kitchen1, Kitchen2 and Store1 are similar when compared the PSNR and SSIM where Middlebury results show JTFite is performing better. Similarly, JTEite performs slightly better on NYU datasets, and most of the Middlebury datasets when compared the BPP. However, implemented JTF and proposed JTFite give poor MSE because of the interpolation of rich texture images such as Cones. During the experiment, it was noted that when the object contains too many sharper corners, the JTF algorithm produces higher MSE. The experiment shows that even excluding refining regions, the calculated MSE is still high due to down-sampling and upsampling. Also, PSNR and SSIM are considerably good where JTF and JTFite give similar results.

In Fig. 5.2 and 5.3, the visual comparison of the UD depth dataset is presented, and it can be seen that the edges are sharper at the right images. Different parameters are used to evaluate the algorithm and found that τ =4 and window size 3x3 gives the best result (See Table 5.3).

Comparison of BPP on JTFite and JTF results for upsampling factor 4X										
4Xw3	Tdy	Cones	Venus	Dolls	Midd2	Mbs	Rdeer	Ktc1	Ktc2	Str1
4 JTF	2.66	4.12	0.71	4.73	2.33	3.72	3.65	5.01	5.45	3.15
6	2.78	4.16	0.70	4.91	2.32	3.81	3.66	4.97	5.59	3.22
7	2.85	4.21	0.70	4.98	2.31	3.86	3.72	4.96	5.62	3.24
8	2.88	4.24	0.71	5.04	2.29	3.90	3.73	4.96	5.62	3.25
10	2.92	4.25	0.72	5.12	2.28	3.95	3.75	4.94	5.63	3.26
4 JTFite	2.65	4.03	0.71	4.56	2.32	3.66	3.59	4.92	5.29	3.16
6	2.72	4.08	0.71	4.75	2.31	3.74	3.6	4.91	5.46	3.22
7	2.76	4.13	0.7	4.82	2.29	3.8	3.65	4.9	5.5	3.25
8	2.8	4.16	0.72	4.88	2.28	3.84	3.67	4.9	5.51	3.27
10	2.83	4.18	0.73	4.96	2.27	3.89	3.69	4.88	5.52	3.28

Table 5.2: The experimental results of the JTF and JTFite are presented along varying τ values where the window size is 3 by 3.

Further, the algorithm is run by using different values of parameters to evaluate their effect on refining the edges. In Table 5.2, 5.3, and 5.4, it can be seen that JTFite

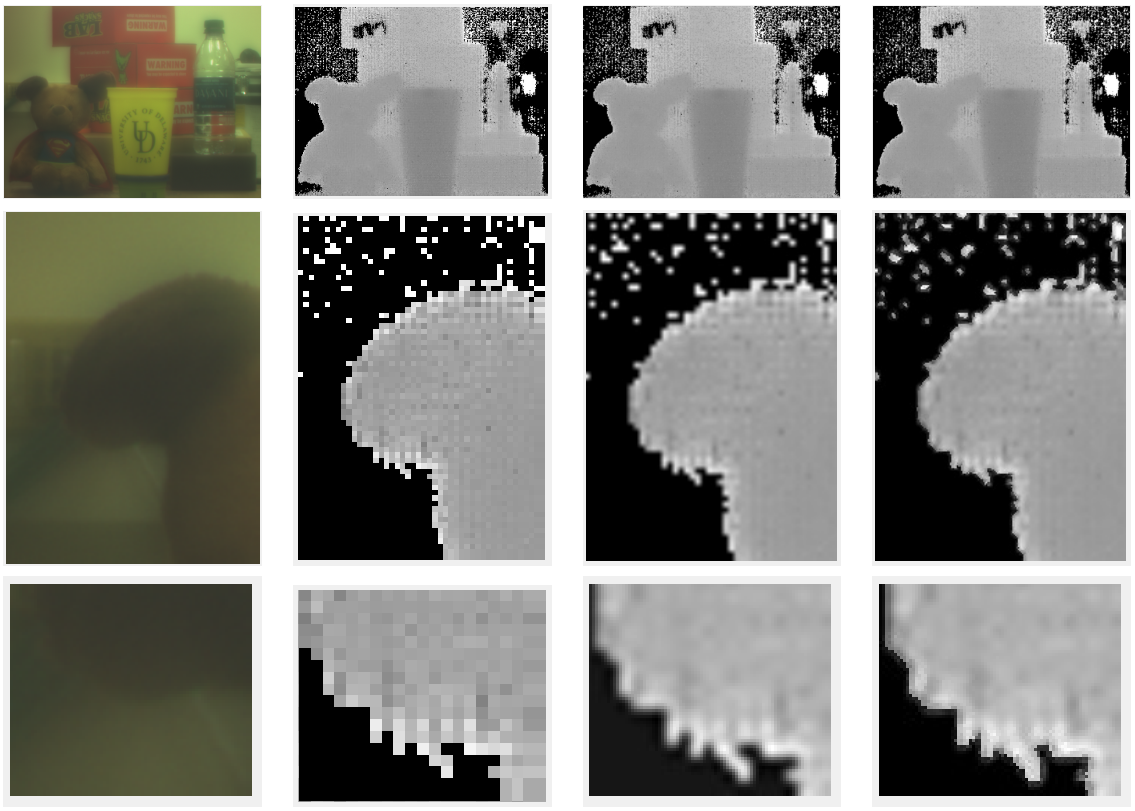


Figure 5.3: Example depth map filtering of UD depth dataset with one ROI. The top figures are HR RGB, LR depth, HR interpolated depth map, and JTFite. The bottom figures are the zoom version of the HR RGB, LR depth, HR interpolated depth map, and JTFite result respectively.

performs better than JTF itself in most cases when results are evaluated by BPP. The zoom versions of all datasets is displayed in Fig. 5.1 for Teddy, Fig.5.5, and Fig.5.6 are for the rest depth maps.

Comparison of BPP on JTFite and JTF results for upsampling factor 4X and 8X										
Datasets th4w3	BP%	BP% Ite.	MSE	MSE Ite.	PSNR	PSNR Ite.	SSIM%	SSIM% Ite.	CT(sc) BRI	CT(sc) SAOPF
Teddy 4X	2.66	2.65	95.62	103.75	28.33	27.97	95.15	94.78	0.88	0.39
Cones	4.12	4.03	235.47	241.27	24.41	24.31	87.30	87.32	0.73	0.26
Venus	0.71	0.71	8.42	8.14	38.88	39.03	98.19	98.26	0.39	0.31
Dolls	4.73	4.56	37.55	39.30	32.38	32.19	95.71	95.49	0.90	0.52
Midd2	2.33	2.32	72.91	73.84	29.50	29.45	93.42	93.32	0.56	0.17
Moebius	3.72	3.66	45.10	47.96	31.59	31.32	95.96	95.68	0.70	0.25
Reindeer	3.65	3.59	81.39	84.03	29.02	28.89	93.57	93.64	0.72	0.22
Kitchen1	5.01	4.92	3.30	4.52	42.95	41.58	99.01	98.86	1.39	0.58
Kitchen2	5.45	5.29	2.86	3.29	43.57	42.96	98.92	98.87	1.98	0.68
Store1	3.15	3.16	2.38	2.47	44.37	44.20	99.06	99.09	1.10	0.47
Teddy 8X	5.11	5.09	163.05	164.67	26.01	25.96	91.86	91.90	0.92	0.23
Cones	7.45	7.40	148.02	150.03	26.43	26.37	89.43	89.57	0.96	0.24
Venus	1.90	1.89	16.73	16.62	35.90	35.92	97.38	97.41	0.44	0.23
Dolls	9.15	9.14	54.04	53.81	30.80	30.82	93.32	93.42	1.05	0.29
Midd2	3.79	3.77	49.64	49.62	31.17	31.17	93.65	93.69	0.62	0.16
Moebius	6.82	6.81	70.06	70.84	29.68	29.63	93.38	93.40	0.78	0.23
Reindeer	6.77	6.74	62.37	62.03	30.18	30.20	93.81	93.97	0.81	0.20
Kitchen1	13.20	13.18	8.89	9.55	38.64	38.33	97.85	97.77	1.36	0.46
Kitchen2	12.63	12.52	8.15	8.20	39.02	38.99	97.53	97.56	1.27	0.37
Store1	6.31	6.29	7.28	6.48	39.51	40.01	97.77	97.92	1.05	0.42

Table 5.3: The experimental results of the JTF and JTFite are presented using the same parameters where τ is 4 and window size is 3. CT BRI and CT SAOPF are the computational time of JTFite for the Binary Region Identification and the Structure-Aware Onion Peel Filtering respectively.

Comparison of BP%, MSE, PSNR, and SSIM with JTFite and JTF results										
4Xth4	Tdy	Cones	Venus	Dolls	Midd2	Mbs	Rdeer	Ktc1	Ktc2	Str1
3x3 JTF	2.66	4.12	0.71	4.73	2.33	3.72	3.65	5.01	5.45	3.15
5x5	2.62	4.20	0.71	4.70	2.39	3.76	3.63	4.97	5.38	3.10
7x7	2.70	4.19	0.71	4.70	2.40	3.76	3.65	4.98	5.36	3.11
9x9	2.74	4.19	0.71	4.69	2.41	3.78	3.65	4.98	5.39	3.12
11x11	2.79	4.19	0.71	4.71	2.42	3.79	3.65	4.98	5.40	3.12
3x3 JTFite	2.65	4.03	0.71	4.56	2.32	3.66	3.59	4.92	5.29	3.16
5x5	2.65	4.12	0.69	4.52	2.37	3.69	3.55	4.83	5.16	3.09
7x7	2.74	4.11	0.69	4.53	2.37	3.71	3.6	4.85	5.15	3.13
9x9	2.82	4.11	0.69	4.53	2.38	3.72	3.59	4.85	5.17	3.15
11x11	2.86	4.10	0.69	4.55	2.38	3.73	3.6	4.85	5.19	3.15

Table 5.4: The experimental results of the JTF and JTFite are presented using the same parameters where τ is 4.

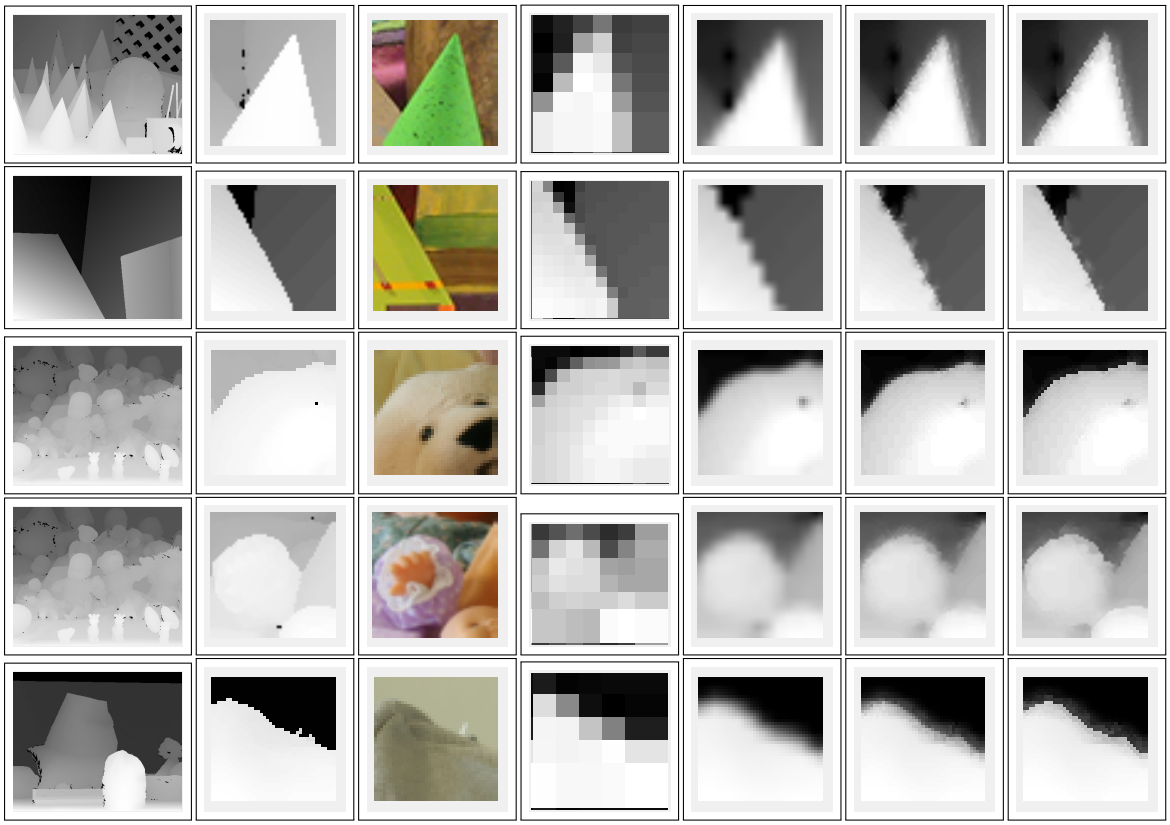


Figure 5.4: Example of Cones, Venus, Dolls(two ROIs) and Midd2 LR depth inputs compared to their corresponding GT, RGB, interpolated HR depth map and filtered outputs. The columns present GT images,and the zoom of RGB, LR input depth images, interpolated depth map, JTF result and JTF result with iteration respectively.

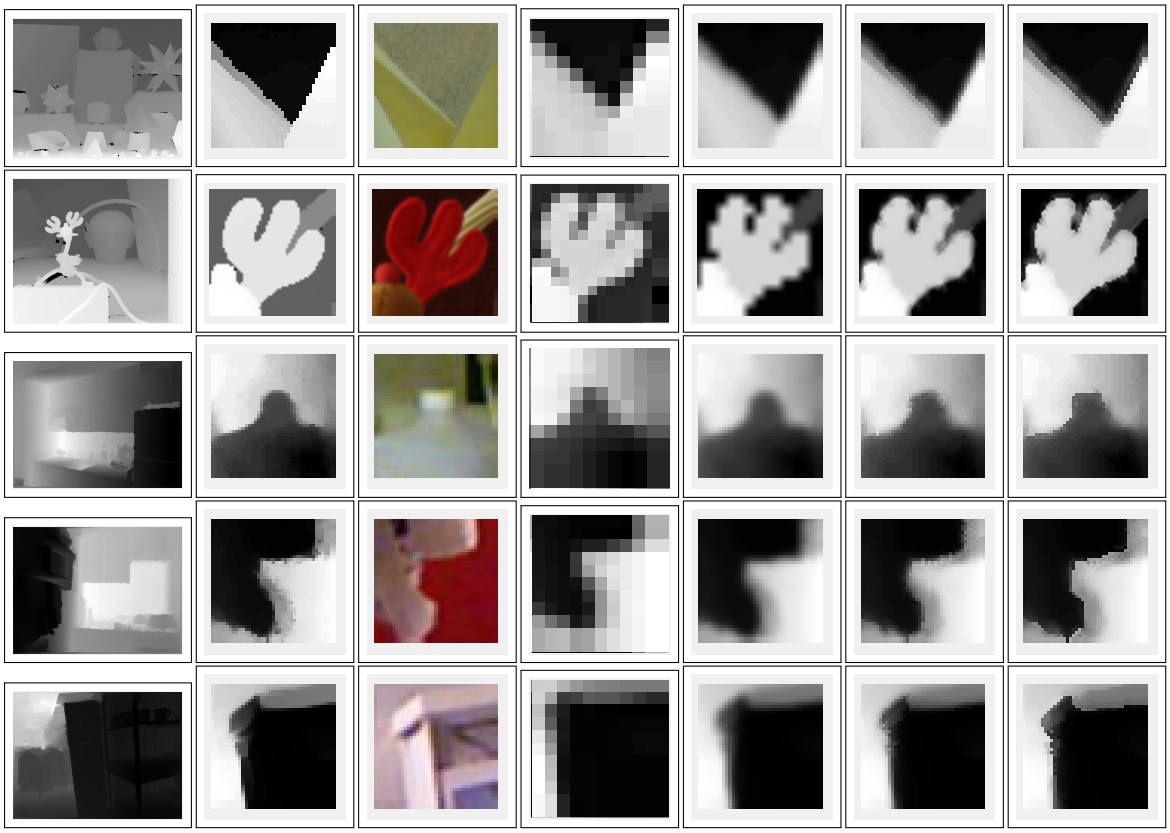


Figure 5.5: Visual Comparison of Moebius, Reindeer, Kitchen1, Kitchen2 and Store1 LR depth inputs compared to their corresponding GT, RGB, interpolated HR depth map and filtered outputs. The columns present GT images,and the zoom of RGB, LR input depth images, interpolated depth map, JTF result and JTF result with iteration respectively.

Chapter 6

CONCLUSION

In this paper, we present the filtering technique Iterative Joint Trilateral Filter (JTFite) for depth map upsampling. The proposed method employs spatial, range, and gradient kernels as the JTF does while post-filtering the output depth map iteratively. A given LR depth input is initially interpolated to have an SR estimator; however, the upsampling causes textural blur and loss of the details. Thus, the method incorporates a registered HR color image with an interpolated depth map to be able to be inclusive of the high-frequency components.

Primarily, JTFite applies a binary identification method to the upsampled depth map to identify depth discontinuities. Our method iterates the determined regions in a structure-aware onion peel algorithm instead of a single filtering as the previous work JTF does. The process divides the interpolated depth map into sub-blocks to apply a quadtree decomposition to identify the depth map edges along object borders at different depth planes. The defined region's pixels are marked as unreliable for further filtering. The filtering process is designed in a structure-aware onion peel filtering with an outside-inward refining procedure that updates all marked pixels by the Canny edge of the color image constraint, which is determined by the union of three channels. JTFite also uses local gradients to prevent textural artifacts which means the weights are assigned to adjacent pixels according to their gradient similarity. Our framework carries on the traits of previous works by employing gradients of the difference of the refined pixels. The gradients show how smooth the transition of the depth planes is and the value of the most changes pixels. Later we synthesize a Canny edge of the previously filtered output with those gradients to bound the next iteration.

Experimental results prove that the proposed method performs well reconstructing the edges by iterating. We repeat the iteration until the value of refinement pixels do not change significantly. At this point, we observe that the edges of the depth maps are recovered from the blur. Even the edge of two near depth planes can be observed distinctly after certain iterations.

Consequently, our framework is based on the application of the gradient on the refined pixels which belong to depth discontinuity regions. JTFite has been implemented to reconstruct edges of HR depth images iteratively where it uses the differences of refined pixel values. Visual and quantitative experimental results are displayed for different parameters. There are many other open research questions can be addressed. Which additional structures can be used to improve the depth map SR? What can be done to make it even faster for a real-time application? Implementing the new technique and adapting the previous algorithmic frameworks run in this work has given an insight that there are still much to do to have a better depth SR upsampling.

BIBLIOGRAPHY

[1] K. H. Lo, Y. C. F. Wang and K. L. Hua, "Edge-Preserving Depth Map Upsampling by Joint Trilateral Filter," in IEEE Transactions on Cybernetics, vol. PP, no. 99, pp. 1-14.

[2] C. Tomasi and R. Manduchi, Bilateral Filtering for Gray and Color Images, Proceedings of ICCV 1998, pp 839-846.

[3] G. Goyal, Impact and analysis of improved bilateral filter on TEM images in International journals of science and research, vol. 3, no. 6, (2014).

[4] Eisemann, E. and Durand, F., 2004. Flash photography enhancement via intrinsic relighting. ACM transactions on graphics (TOG), 23(3), pp.673-678.

[5] O. U. N. Jith and R. Venkatesh Babu Joint bilateral filtering based non local means image de-noising, in the proceedings of IEEE transactions on image processing, (2014).

[6] G. Petschnigg, R. Szeliski, M. Agrawala, M. Cohen, H. Hoppe, and K. Toyama, Digital photography with flash and no flash image pairs, in ACM SIGGRAPH , (2004), pp. 664-672.

[7] S. Deswal, S. Gupta and B. Bhushan, "A Survey of Various Bilateral Filtering Techniques," International Journal of Signal Processing, Image Processing and Pattern Recognition, vol. 8, no. 3, pp. 105-120, 2015.

[8] Johannes Kopf, Michael F. Cohen, Dani Lischinski, and Matt Uyttendaele. 2007. Joint bilateral upsampling. ACM Trans. Graph. 26, 3, Article 96 (July 2007).

[9] Sylvain Paris, Pierre Kornprobst, Jack Tumblin and Frdo Durand (2009), "Bilateral Filtering: Theory and Applications", Foundations and Trends in Computer Graphics and Vision: Vol. 4: No. 1, pp 1-73.

[10] Y. Song and L. Gong, "Analysis and improvement of joint bilateral upsampling for depth image super-resolution," 2016 8th International Conference on Wireless Communications & Signal Processing (WCSP), Yangzhou, 2016, pp. 1-5.

[11] Li, Yangguang et al. Depth map super-resolution via iterative joint-trilateral-upsampling. 2014 IEEE Visual Communications and Image Processing Conference (2014): 386-389.

[12] Prasun Choudhury and Jack Tumblin. 2003. The trilateral filter for high contrast images and meshes. In Proceedings of the 14th (EGRW '03). Eurographics Association, Aire-la-Ville, Switzerland, 186-196.

[13] Z. Wang, Image Quality Assessment: from Error Visibility to Structural Similarity," IEEE Transactions on Image Processing, Vol. 13, No. 4, pp. 600612, Apr. 2004

[14] Middlebury Stereo. Accessed on Oct. 20, 2017. [Online]. Available: http://vision.middlebury.edu/stereo/

[15] D. Scharstein and R. Szeliski, A taxonomy and evaluation of dense twoframe stereo correspondence algorithms, Int. J. Comput. Vis., vol. 47, nos. 13, pp. 742, 2001.

[16] Johannes Kopf, Matt Uyttendaele, Oliver Deussen, and Michael F. Cohen. 2007. Capturing and viewing gigapixel images. ACM Trans. Graph. 26, 3, Article 93 (July 2007).

[17] Pal, Chandrajit & Chakrabarti, Amlan & Ghosh, Ranjan. A Brief Survey of Recent Edge-Preserving Smoothing Algorithms on Digital Images. (2015).

[18] Liu Ying-hui, Gao Kun and Ni Guo-qiang, "An improved trilateral filter for Gaussian and impulse noise removal," 2010 The 2nd International Conference on Industrial Mechatronics and Automation, Wuhan, China, 2010, pp. 385-388.

[19] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus, Indoor segmentation and support inference from RGBD images, in Proc. Eur. Conf. Comput. Vis. (ECCV), Florence, Italy, 2012, pp. 746760.

[20] NYU depth dataset V2. Accessed on Oct. 20, 2017. [Online]. Available: http://cs.nyu.edu/∼ silberman/datasets/nyu_depth_v2.html

[21] S. A. Gudmundsson, H. Aanaes and R. Larsen, "Environmental Effects on Measurement Uncertainties of Time-of-Flight Cameras," 2007 International Symposium on Signals, Circuits and Systems, Iasi, 2007, pp. 1-4.

[22] N. Otsu, "A Threshold Selection Method from Gray-Level Histograms," in IEEE Transactions on Systems, Man, and Cybernetics, vol. 9, no. 1, pp. 62-66, Jan. 1979.

[23] Frdo Durand and Julie Dorsey. 2002. Fast bilateral filtering for the display of high-dynamic-range images. In Proceedings of the 29th annual conference on Computer graphics and interactive techniques (SIGGRAPH '02). ACM, New York, NY, USA, 257-266.