Constructing High-Quality 3D Object Models Using RGB-D Cameras

Mayoore Selvarasa Jaiswal

A dissertation

submitted in partial fulfillment of the

requirements for the degree of

Doctor of Philosophy

University of Washington

2018

Reading Committee:

Ming-Ting Sun, Chair

Eve Riskin

Radha Poovendran

Program Authorized to Offer Degree:

Electrical Engineering

University of Washington

Abstract

Constructing High-Quality 3D Object Models Using RGB-D Cameras

Mayoore Selvarasa Jaiswal

Chair of the Supervisory Committee:
Professor Ming-Ting Sun
Department of Electrical Engineering

With the introduction of economical depth cameras, computer vision research has made a huge leap forward in 3D reconstruction and understanding. However, the quality of the depth images are limited: 1) depth images contain holes and random noise due to the characteristics of the cameras and the physical world, and 2) depth images have a lower spatial resolution. These challenges make 3D object reconstruction with a limited number of RGB-D frames a difficult task. In this dissertation, we propose a method to construct 3D object models with a limited number of RGB and depth frames. We developed a complete 3D object model construction process with automatic object segmentation, pairwise registration, global alignment, model denoising, and texturing, and studied the effects of these functions on the constructed 3D object models. We also developed a process for objective performance evaluation of the constructed 3D object models.

High-quality depth images are paramount to create quality 3D object models. Many depth denoising methods blur object boundaries. Partial point clouds created from depth images denoised by these methods have artifacts that make them unsuitable for 3D object modeling. We propose a method to find the clean depth edge image using the noisy depth image and a color image, then use this clean depth edge image with the proposed edge and color aware adaptive trilateral filter to obtain denoised depth images.

Finally, we propose to apply depth denoising and depth super-resolution in the 3D object modeling process. We preprocess depth images with the proposed depth denoising method and a state-of-the-art depth super-resolution method, then experiment the use of these preprocessed depth images under various conditions in the 3D object modeling process. We show that depth denoising and super-resolution can improve the quality of the 3D object models.

TABLE OF CONTENTS

List of Figures iv

List of Tables viii

Chapter 1. Introduction 2

Chapter 2. RGB-D cameras 4

2.1 Brief review of RGB-D cameras 4

2.1.1 Kinect V1 5

2.1.2 Kinect V2 9

2.2 Related Work 11

Chapter 3. 3D Object Modeling with a Kinect Camera 14

3.1 Background 14

3.2 3D Object Modeling Framework 16

3.2.1 Object Segmentation from the RGB-D Images 17

3.2.2 Registration 19

3.2.3 Global Alignment 21

3.3 Model Denoising and Texturing 26

3.3.1 3D Object Model De-Noising 26

3.3.2 Meshing and Texturing 28

3.4 Evaluation of 3D Kinect Models 30

3.4.1 Comparison of Kinect Model Point Cloud with the Point Cloud of a High-End Laser Scanner 30

3.4.2 Error Calculation 31
3.5 Conclusion 36
Chapter 4. Object Boundary Based Hole-Filling and Denoising for Kinect Depth Images 37
4.1 Background 37
4.2 Proposed Method 40
4.2.1 Color edge extraction 42
4.2.2 Noisy Depth Edge Extraction 43
4.2.3 Generating Clean Depth Edges 44
4.2.4 Depth Hole Filling 49
4.3 Experimental Results 52
4.3.1 Results on Synthetically Degraded Dataset 53
4.3.2 Results on Kinect dataset 58
4.4 Conclusion 59
Chapter 5. Applying Denoising and Super-Resolution in the 3D Modeling Process 61
5.1 Introduction 61
5.2 Comparing Different Approaches 63
5.2.1 Denoising depth images followed by the baseline algorithm 64
5.2.2 Super-resolving depth images followed by the baseline algorithm 66
5.2.3 Denoising followed by super-resolving depth images before the baseline algorithm 68
5.3 Experimental Results 70
5.4 Discussion and Future Work 77

5.5 Conclusion 78

Chapter 6. Conclusion and Future Work 79

6.1 Conclusion 79

6.2 Future Work 81

6.2.1 Using multiple RGB-D cameras 81

6.2.2 Improved partial point cloud registration 82

6.2.3 Data-driven depth denoising 83

6.2.4 Comparing effect of super-resolution techniques on 3D models 83

Appendix A 85

Bibliography 115

LIST OF FIGURES

Figure 2-1. The Kinect V1 camera with its hardware components labeled. 6
Figure 2-2. RGB (a) and Depth (b) image captured by Kinect from one view. 8
Figure 2-3. The Kinect V2 camera with its hardware components labeled. 10
Figure 2-4. Overview of KinectFusion: (a) The setup for creating a 3D model using Kinect. A Kinect is slowly moved around the scene. (b) Flowchart of the KinectFusion algorithm. 13
Figure 2-5. Overview of CopyMe3D. Courtesy of [7]. 13
Figure 3-1. (a) A flowchart of the Kinect based 3D object modeling process. Partial point clouds are registered to form the complete 3D point cloud. Global alignment is performed to align the last view with the first view. Model is then de-noised, followed by surface representation and color texturing. (b) The point-cloud generation and registration process. 17
Figure 3-2. Object placed in the green background and the captured RGB (a) and Depth (b) images. 18
Figure 3-3. (a) and (b) show the RGB and depth images, respectively, with their object boundary highlighted, and the object boundary obtained from the segmentation mask. 19
Figure 3-4. Global alignment (a) shows a 3D model obtained without the global alignment. The red highlight shows the result of error propagation when global alignment is not performed. (b) shows the loop closure situation. 22
Figure 3-5. The global alignment procedure used to eliminate error accumulated due to the pairwise registration. 23
Figure 3-6. Modeling result after the global alignment. (a) Before the global alignment (after pairwise registration) (b) After the global alignment procedure. 26
Figure 3-7. Effects of depth image de-noising in 3D projection. (a) 3D point cloud before de-noising the depth image. (b) 3D point cloud after de-noising the depth image using a joint bilateral filter. 27

Figure 3-8. Modeling result of MLS based de-noising. (a) Before 3D de-noising (after registration). (b) After MLS de-noising. 28
Figure 3-9. A Delaunay triangulation in the plane with the circumcircles shown. 29
Figure 3-10. 3D Modeling result of meshing and texturing. (a) Meshing (b) Texture Mapping. 30
Figure 3-11. Methodology for calculating the error between two models. 32
Figure 3-12. An illustration of error calculation methodology. (a) and (b) are point cloud representations of two 3D models. (c) Result obtained by applying RANSAC. (d) Result obtained by applying ICP fine registration on (c). Note the error between the two models. 33
Figure 3-13. Objects used for evaluating the results (a) Biscuit box, (b) Nut container, (c) House model, and (d) Mug. 34
Figure 4-1. A depth image and its corresponding color image obtained from a Kinect camera. A sample of a shadow hole is highlighted in red and a random noise is highlighted in green. 37
Figure 4-2. Illustration of the proposed method. Given a noisy depth image and the color image, the corresponding edge images are extracted. These edge images are used to produce a clean depth edge image, which along with the noisy depth image is used to produce the hole-filled depth image. 41
Figure 4-3. (a) Input color image (b) Edges extracted from the color image using the adaptive Canny edge detection 43
Figure 4-4. (a) Input noisy depth image (b) Edges extracted from the noisy depth image using Canny edge detector. 44
Figure 4-5. Depth edges E from Equation (4.7) overlaid on the color image (a) and on the noisy input depth image (b). 49
Figure 4-6. Thumbnails from the synthetically degraded dataset [25]. The top row shows the color images. The middle row shows the corresponding depth images degraded by random noise and structural holes. The bottom row shows the ground truth depth images. . 54
Figure 4-7. A visual comparison of results from the state-of-the-art methods on the ‘Art’ image. (a), (b) and (c) are the color, synthetically degraded depth, and the ground-truth depth

respectively. (d) – (i) show zoomed in view of different methods in the red rectangle region in the ‘Art’ image. (d) bicubic (e) JBF [32] (f) Guide [21] (g) CLMF [22] (h) AAR [25] and (i) ours. Note that our method preserves the edge structures the best without creating a halo around the object boundary as shown in the boxes highlighted in black..... 57
Figure 4-8. A visual comparison of results from the state-of-the-art methods on the ‘Book’ image. (a), (b) and (c) are the color, synthetically degraded depth, and the ground-truth depth respectively. (d) – (i) show zoomed in view of different methods in the red rectangle region in the ‘Book’ image. (d) bicubic (e) JBF [32] (f) Guide [21] (g) CLMF [22] (h) AAR [25] and (i) ours. Note that our method preserves the edge structures the best, there are no “orange” colored pixels in the colored recovered depth image as illustrated in the boxes highlighted in black..... 58
Figure 4-9. A visual comparison of results from the state-of-the-art methods on a set of Kinect images. The thumbnails highlighted in red illustrate zoomed-in portions of the image above them. Column (a) is the color image from Kinect and (b) is the corresponding depth from Kinect. (c) AAR [25]. (d) Guide [21]. (e) Our method..... 59
Figure 5-1. (a) An example depth image (b) the corresponding depth image denoised using [36]. 64
Figure 5-2. The flowchart for denoising the depth images before the partial point cloud registration. 65
Figure 5-3. (a) Depth image (b) the corresponding depth image super-resolved by a factor of 2 using [37]. 67
Figure 5-4. The flowchart for super-resolving the depth images. The initial depth super-resolution of the depth images is the difference between the baseline method and this method..... 68
Figure 5-5. (a) Depth image (b) the corresponding depth image first denoised using [36] and then super-resolved by a factor of 2 using [37]. 69
Figure 5-6. The flowchart for this method. The initial depth denoising and super-resolution of the depth images is the difference between the baseline method and this method... 70
Figure 5-7. Objects tested under various denoising and super-resolution conditions. 71

Figure A-1. Typical thick film microscope image. This field-of-view image contains only two parasites indicated by yellow circles with enlargements at the right. 86
Figure A-2. Ring form *P. falciparum* malaria parasites. 88
Figure A-3. An FoV image of a negative sample, i.e. with no malaria parasites. WBCs are indicated with red circles. 89
Figure A-4. Examples of distractors. The objects in the upper left and lower right corner are platelets. 90
Figure A-5. FROC curves for *P. falciparum* detection based on standard grayscale vs. adaptive grayscale images. 98
Figure A-6. Diagnosis by parasitemia for 4 holdout sets. Each dot represents a positive patient. Blue dot: correct diagnosis, red dot: false negative. Green lines indicates parasitemia range used for WHO evaluation (90% sensitivity @ 90% specificity = competency level 1). 105
Figure A-7. Parasitemia quantitation performance on holdout sets. Green lines indicate ± 25% range. 106

LIST OF TABLES

Table 2-1. Comparison between Kinect V1 and V2 features 11
Table 3-1. The root mean square error between two randomly subsampled point clouds from the same laser point cloud. 34
Table 3-2. Comparing results with KinectFusion for 3D object model construction using Kinect V1. 35
Table 3-3. Comparing results with KinectFusion for 3D object model construction using Kinect V2. 35
Table 4-1. Table lists the quantitative results comparing our method to the state-of-the-art methods tested on the synthetically degraded dataset [25] using Equation (4.12) to calculate Mean Absolute Error (MAE) and Structure Similarity Index (SSIM). Our method improves the best MAE for each image by 14.02% on average. Our method improves or on par with the other methods on the SSIM metric. 55
Table 4-2. Table lists the quantitative results comparing our method to the state-of-the-art methods tested on the synthetically degraded dataset [25] using Equation (4.13) to calculate Mean Absolute Error (MAE). Our method improves the best MAE for each image by 36.78% on average. 56
Table 5-1. Comparing the results from the baseline 3D object modeling process (A) vs the denoising depth images followed by the baseline 3D object modeling process (B). 72
Table 5-2. Comparing the effect of different depth denoising methods on the resultant 3D models by using 45 RGB-D frames to create 3D models. 73
Table 5-3. Comparing the results from the baseline 3D object modeling process (A) vs the depth superresolution followed by the baseline 3D object modeling process (B). 74
Table 5-4. Comparing the results from the baseline 3D object modeling process (A) vs the depth denoising and super-resolution followed by the baseline 3D object modeling process (B). 76
Table A-1. Summary of thick film malaria database. 94
Table A-2. Confusion matrix for CNN on validation set. 107

Table A-3. Metrics of manual and automated malaria diagnosis algorithms. 108

ACKNOWLEDGEMENTS

First and foremost, I want to express my most sincere gratitude to my advisor Prof. Ming-Ting Sun. I appreciate all his contributions, especially his time, and ideas to make my Ph.D. experience productive and stimulating. I am extremely grateful for his patience, guidance, useful discussions and deep insights that have helped me at various stages of my research. Thank you for your mentorship Professor Sun! I would also like to thank my reading committee members Prof. Eve Riskin and Prof. Radha Poovendran for their time, insightful questions and helpful comments, and the graduate school representative (GSR) on my committee Prof. Daniel Ratner for his time and interest.

I want to thank my lab-mates, especially Jun Xie and Yu-Ying Wang, at the Multimedia Signal Processing lab for their collaboration. I am indebted to Prof. Radha Poovendran for the opportunity to collaborate with the Network Security Lab. Words cannot express my thankfulness to my colleagues at the Intellectual Ventures Laboratory, especially Charles Delahunt, Courosh Mehanian, Cary Champlin and Matt Horning.

My time at the University of Washington was enriched in large part due to the groups that became a part of my life. Specifically the PEERs Leaders, Husky Experience Student Advisory Council (HESAC), College of Engineering Student Advisory Council (COESAC), Husky Leadership Initiative and the UWEE IEEE student chapter. I am grateful to Prof. Eve Riskin, Dr. Joyce Yen, and Prof. Sapna Cheryan for giving me the opportunity to be a PEERs leader. I owe everything I learned about project management to Katy DeRosier at the UW Graduate School.

Very special thanks to the University of Washington Electrical Engineering department for giving me the opportunity to carry out my doctoral research and for their financial support by the way of teaching assistantships. I was honored to receive the Paul C. Leach Fellowship and Eugene Beebe Scholarship (GO-MAP) in 2013 and 2015 respectively. Many thanks to the Electrical Engineering graduate advisors Brenda Larson and Bryan Crockett for patiently answering many administrative questions in a timely manner. I wish to express my most sincere appreciation to all those who have contributed to this dissertation, crossed paths and supported me in one way or the other during this amazing journey.

Last but not the least, I would like to thank my family for all their love and encouragement. For my mom and dad who put me before anything else in their life. They raised me with a love of science, freedom to dream without bounds and supported me unconditionally in all my pursuits. For my supportive, encouraging, and patient husband Prasanna, Thank you!

DEDICATION

To my mom and dad

Chapter 1. INTRODUCTION

RGB-D (Kinect-style) cameras are novel low-cost sensing systems that capture RGB images along with per-pixel depth information in real-time. In this dissertation, we investigate the use of such cameras for building a high-quality 3D model of an object with a limited number of available views of the object. Such 3D object models have applications in a wide range of industries. We developed a complete 3D object model construction process with object segmentation, pairwise registration, global alignment, model denoising, and texturing, and studied the effects of these functions on the constructed 3D object models. We also developed a process for objective performance evaluation of the constructed 3D object models. We collected laser scan data as the ground truth using a Roland Picza LPX-600 Laser Scanner to compare to the 3D models created by the proposed process and other state-of-the-art methods and show the advantages of our proposed approach.

One of the limitations of RGB-D cameras is that its depth image contains holes, noises, and inaccurate depth measurements. The accuracy of various RGB-D related applications, such as 3D modeling, suffers from these depth image errors. In this work, we propose a solution to the depth image denoising problem by estimating depth edges that correspond to the object boundaries and using them as priors with the edge and color aware adaptive trilateral filter. This method exhibits quantitative and qualitative improvements over the current state-of-the-art methods.

The quality of depth images is limited by noise and relatively low resolution. Several approaches have been proposed to denoise and super resolve depth images. We show that the quality of the 3D object models can be improved by using the depth denoising method proposed in this work and a state-of-the-art depth super-resolution method to rectify depth images before the 3D object modeling process.

As part of the Ph.D. study, we have collaborated on malaria diagnosis and quantification with Intellectual Ventures Laboratory in Bellevue, WA. The results of this work have been documented in a paper. Since malaria diagnosis and quantification is not directly related to the 3D object modeling work which is the main theme of this dissertation, we attach that work as an Appendix of this dissertation.

The rest of this dissertation is organized as follows. In Chapter 2, we provide a brief background overview of the Kinect camera and review Kinect Fusion which is often used for constructing 3D scenes. In Chapter 3, we describe our proposed 3D object modeling framework. In Chapter 4, we discuss our proposed algorithm for denoising and hole-filling the depth images. In Chapter 5, we apply and evaluate the usefulness of denoising and depth super-resolution in the 3D modeling framework. Finally, in Chapter 6 we conclude this work and discuss possible future work. Our collaborative work with the Intellectual Ventures Laboratory is attached as an Appendix.

Chapter 2. RGB-D CAMERAS

2.1 BRIEF REVIEW OF RGB-D CAMERAS

Laser scanners and stereo cameras are able to output 3D data of an environment. Laser scanners are precise but are slow to compute a full scan, and are expensive. Stereo cameras can be cheap but require complex processing to compute depth estimations, and have poor depth estimations in homogeneous areas. The revolution brought by the RGB-D cameras is the combination of the features of these devices. RGB-D cameras are sensing systems that capture RGB images along with per-pixel depth information. RGB-D cameras rely on active stereo, structured light or time-of-flight sensing to generate depth estimates at pixels. While sensor systems with these capabilities have been custom-built for years, now they are available at an economical cost and smaller form factors. With the advent of economical RGB-D cameras, there is a significant interest in using them for various 3D applications including 3D object modeling.

RGB-D cameras are capable of delivering real-time, reasonably accurate mid-resolution depth and color data at an economical cost. However, one of the limitations of these cameras is that the depth map is noisy and may contain regions without data due to the surface property of the object and the occlusions caused by the locations of the infrared light projector and the sensors. Another limitation is that the depth images are a relatively low resolution, thus the partial point cloud from each frame is sparse. These limitations make the construction of high-quality 3D object models using RGB-D cameras a challenging task.

Many RGB-D cameras available in the market. PrimeSense Carmine captures 640×480 registered color and depth images at 30 frames per second (fps). Depth images are captured using structured light with an operating range of 0.35 to 1.4 m. Asus XtionPro Live also uses structured

light which projects an Infrared light pattern on the 3D scene and based on the distorted pattern calculates the depth images with 640×480 resolution from 0.8 to 3.5 m distance from the scene. The RGB image resolution is 1280×1024 at 30 fps. Intel RealSense R200 is a stereo depth camera with a depth resolution of 640×480 at 60 fps and full HD color resolution at 30 fps. The depth range is 0.6 to 3.5 m. This camera is one of the few RGB-D cameras that can be used in outdoor environment. Mesa SwissRanger 4000 and CamCube 2.0 are popular Time-of-Flight (ToF) based depth cameras but neither of them collects RGB images.

We use Kinect cameras in our work because they are a good representative of the range of RGB-D cameras available in the market, and popular within the research community. In the following sections, we describe the Kinect cameras in detail.

2.1.1 *Kinect V1*

Microsoft Kinect was released in November 2010 as part of its Xbox gaming console. Kinect's software was developed internally at Microsoft, whereas, its hardware was developed by PrimeSense. Following great interest by the research community, Microsoft released its official Kinect SDK in November 2011. In addition to Kinect SDK, there are other open source alternatives, such as libfreenect OpenKinect and OpenNI.

Figure 2-1 shows the hardware components of Kinect. It has an IR (Infrared) projector, IR camera, RGB camera, motorized tilt and a multi-array microphone. With this hardware, Kinect is capable of recording RGB, IR, depth, skeleton and audio signals. Abiding by the needs of this work, we limit our discussion to the RGB and depth data.

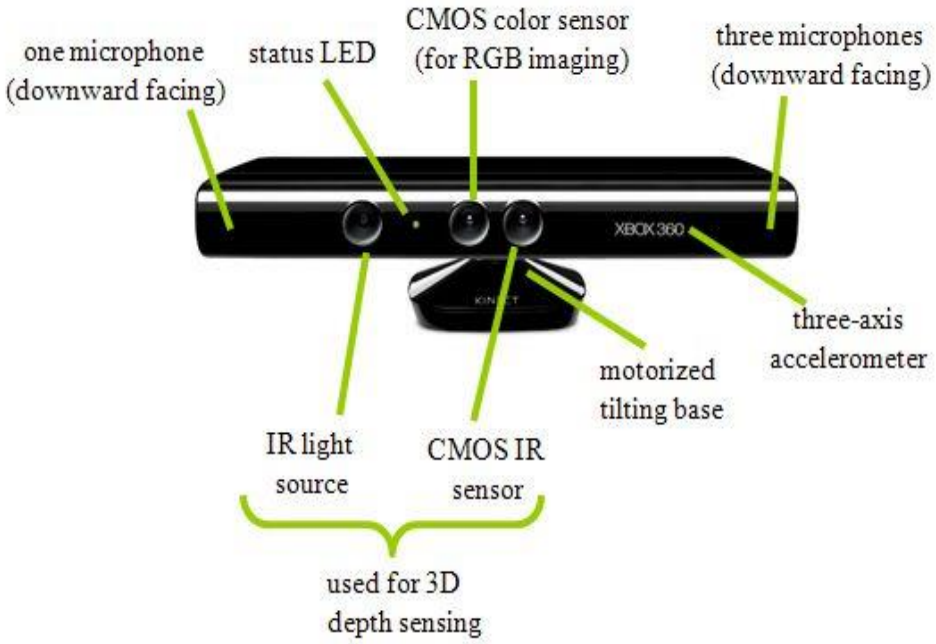


Figure 2-1. The Kinect V1 camera with its hardware components labeled.

2.1.1.1 RGB Images

The RGB image is captured using a CMOS 3-channel color sensor with a Bayer color filter at a frame rate of 30 fps. Each RGB frame has a 640×480 pixel resolution. Each pixel has 8 bits. It is capable of capturing at a 1280×960 pixel resolution with a lower frame rate of 15 fps.

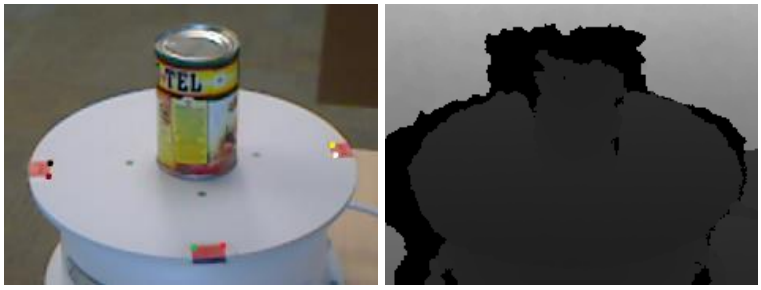
2.1.1.2 Depth Images

The depth image is acquired using an Infrared laser projector and a Monochrome CMOS sensor using structured light imaging. The IR projector projects a known, pseudo-random speckles, Near-Infrared light pattern onto the 3D scene. The IR camera records the distorted light pattern. The projected speckles' size and shape depend on the distance and orientation of an object with respect to the sensor. Kinect uses three different sizes of speckles for three different regions of distances.

In the first region, 0.8 – 1.2 m, it allows obtaining highly accurate depth surfaces of near objects. In the second region, 1.2 – 2.0 m, it allows obtaining medium accurate depth surfaces. Finally, in the third region, 2.0 – 3.5 m, it allows obtaining a relatively low accurate depth surface of far objects. Each speckle in the reference image is triangulated to the observed pattern and its disparity is measured. Using the calibrated speckle pattern, the 3D map of the beginning frame is computed. Then the 3D map is renewed in the x-direction of speckle shift. Camera calibration parameters are evaluated during the manufacturing process using a set of reference images taken at different locations. Using the reference light pattern, calibration parameters and the measured distortion, the real world distance is calculated inside the sensor.

The data from the IR sensor are represented in 11 bits, which includes 1 bit allocated to mark the validity of the depth data. With 10 bits to represent the value, the IR sensor has 1024 levels of sensitivity to store the disparity measurements. After converting to the real world distance in millimeters – the depth image has 13 bits of data for each pixel with the low-order 3 bits for the player index. The raw depth image resolution is 320×240 pixels. The spatial (x/y) resolution is 3mm and the depth resolution is 3mm at 2m distance from the camera plane.

Figure 2-2 shows an example of RGB and depth images captured by a Kinect V1 camera. In the depth image, the black pixels represent pixels where no data is available. We can see that the depth image has a lot of holes (where data are not available) and is very noisy. Also, the depth images have relatively low resolution per frame.



(a) (b)

Figure 2-2. RGB (a) and Depth (b) image captured by Kinect from one view.

Errors in Kinect data may originate from: 1) the sensor due to inaccurate measurements of disparities, inadequate calibration, or errors in the estimation of calibration parameters, 2) the measurement setup is affected by the lighting condition and the imaging geometry, and 3) the properties of the object surface. One of the limitations of Kinect is that it cannot work outdoors because it is limited to a short range (about 3.5 m although it could be pushed to 8m with somewhat unreliable depth accuracy) and its IR sensor cannot distinguish between the Sun's IR and the Kinect's IR source due to insufficient contrast between the two. Also, its precision rapidly decreases with the distance, and the depth map is noisy and may contain regions without data.

The depth image obtained by the Kinect sensor has a few shortcomings that are discussed here. Holes are regions where there is no reliable depth information available. This occurs due to smooth surfaces, shadows, occlusion and low/high reflectivity of materials. The lighting condition influences the IR speckle pattern. In strong light, the laser speckles appear in low contrast in the infrared image, which can lead to outliers or gaps in the resulting point cloud. Inadequate calibration or error in the estimation of calibration parameters can lead to systematic errors in the object coordinates of individual points. The IR sensor resolution is 1280×1024 pixels per frame. However, the Kinect's processor can only process 320×240 pixels per frame due to the Xbox 360's USB2 bandwidth limitation. The Kinect camera has a relatively short normal operating range

of 0.5 – 3.5m. The raw depth and RGB data are not aligned. To eliminate distortions in the point cloud and misalignments between the color and the depth data, an accurate stereo calibration of the IR camera and the RGB camera is necessary. This can be achieved using the Kinect SDK or OpenKinect SDK.

Depth resolution is the minimum depth difference that can be measured. The depth resolution of a Kinect camera decreases quadratically with increasing distance from the sensor. Thus, the errors of depth measurements increase quadratically with increasing distance from the sensor. In general, for 3D scanning applications, the data should be acquired within 1–3m distance to the sensor. At larger distances, the quality of the data is degraded by the noise and low resolution of the depth value measurements. Depth images also have a relatively low spatial resolution. Thus, each partial point cloud created from a depth image is relatively sparse.

2.1.2 *Kinect V2*

Microsoft launched the next generation of Kinect V2 sensor (Figure 2-3) in 2014. Table 2-1 summarizes the Kinect V1 and V2 features. Kinect V2 captures in real-time, color images at full HD resolution of 1920×1080 and a depth sensor having a resolution of 512×424 using the Time-of-Flight (ToF) principle. ToF cameras estimate the distance to an object by measuring the phase difference between the IR waves radiated by its emitter and the corresponding reflected IR waves detected by its sensors. These values can be used to estimate the 3D geometry of the scene directly. Kinect V2 also provides a wider field of view (70 degrees horizontally and 60 degrees vertically), a similar working range (0.5-8 meters) and the same frame rate (30 fps) as its predecessor. The depth capture mechanism has reduced dependence on scene illumination and no

dependence on surface texturing. Since ToF depth sensing is used, the occlusion area is significantly reduced and the depth values are more stable than the first generation Kinect.

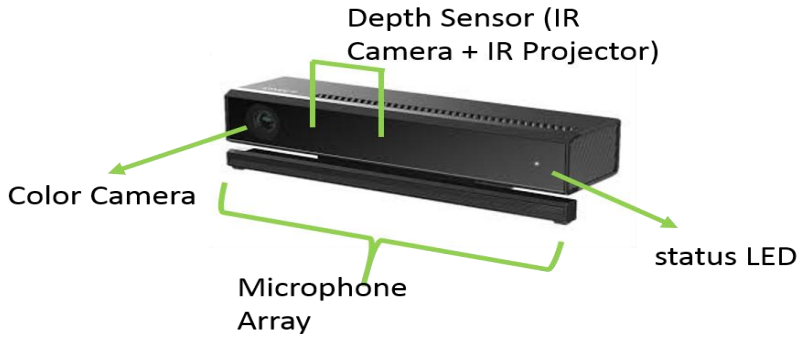


Figure 2-3. The Kinect V2 camera with its hardware components labeled.

However, Kinect V2 shares the same imperfections as other ToF cameras which include depth distortion, systematic error, and light scattering errors. Depth distortions and distance errors happen when a small amount of received light from surfaces with low IR reflectivity. Systematic errors arise due to IR demodulation error, integration time error, amplitude ambiguity and temperature error. Light scattering errors, commonly known as the “flying pixels” phenomenon occurs due to diffuse, specular, translucent and subsurface light scattering from objects like mirrors, computer monitors, and object boundaries. Light interference affects the sensor when the signal received by the detector is mixed with signals that were reflected in the scene multiple times (instead of direct reflection), emitted IR waves are attenuated and scattered in the scene, and interfered by other sources of near-infrared lights (e.g., sunlight, infrared marker-based tracking systems, other ToF cameras). Hence, depth map restoration is still a crucial step for most depth based applications.

Although Kinect V2 uses the same principle as the traditional ToF cameras, there exists a notable difference on the IR illumination scheme. Comparing to SwissRanger 4000 and CamCube 2.0, the

IR emitter of Kinect V2 is located only on one side of the IR sensor. This hardware configuration creates holes due to the occlusions caused by the displacement between IR emitter and IR sensor. Kinect V2 was discontinued by Microsoft on October 25, 2017. However, the techniques discussed in this dissertation can still be used with other RGB-D cameras such as the Intel RealSense camera.

Table 2-1. Comparison between Kinect V1 and V2 features

Feature	Kinect V1	Kinect V2
Color Camera	640 x 480 pixels @30 fps	1920 x 1080 pixels @30 fps
Depth Camera	320 x 240 pixels	512 x 424 pixels
Max Depth Distance	~4.5 m	~4.5 m
Min Depth Distance	40 cm in near mode	50 cm
Horizontal Field of View	57 degrees	70 degrees
Vertical Field of View	43 degrees	60 degrees
Tilt Motor	Yes	No
Skeleton Joints Defined	20 joints	26 joints
Full Skeletons Tracked	2	6
USB Standard	2.0	3.0
Supported OS	Win 7, Win 8	Win 8

2.2 Related Work

The seminal paper KinectFusion [1] and its variants [2, 3] create 3D reconstructions of indoor scenes using only the depth data captured by Kinect in real time. Figure 2-4 (a) illustrates the process for creating a 3D model and (b) shows the main components of the algorithm using Kinect Fusion. This system relies on scanning with the slow motion of the Kinect camera and high cluttered environment to maintain its tracking. KinectFusion relies on spatial and temporal

averaging to reduce the noise. In the registration of the 3D partial point clouds from different views, KinectFusion uses the ICP (Iterative Closest Point) algorithm [4, 5, 6] which requires the 3D partial point clouds to be close to each other. Thus, the scene has to be scanned slowly in order to make sure the neighboring views have significant overlaps so that the denoising and the registration work well. Also, to fill the holes in a particular location in the 3D model, one has to scan the scene slower and take more data at that location. Another potential problem is that in the registration KinectFusion relies on salient structural features in the registration, thus, it may have difficulty in handling objects lacking salient structural features (such as cylindrical or round objects) and when the texture information of the 3D model is also needed. CopyMe3D [7] proposes an approach to create 3D models of people sitting on a swivel chair and scanned by a Kinect as shown in Figure 2-5. Their algorithm utilizes the frames available in the RGB-D video captured by the Kinect. Both KinectFusion and CopyMe3D use continuous video feed (30 fps) from the Kinect with human control so that adjacent frames have significant overlaps. ReconstructMe and KScan are two commercial products that use Kinect to create 3D models of objects.

In this work, we investigate a scenario where only a limited number of views from Kinect are available for constructing the 3D object models. Since only a relatively small number of frames are available, the overlapped areas between frames from different views can be rather limited, and thus, the denoising and registration as in KinectFusion or CopyMe3D may not be as effective. Also, the algorithm needs to be able to take care of objects lacking salient structural features and provide good texture for the constructed 3D object models. Our proposed framework in Chapter 3 overcomes these problems. Specifically, in the registration process, we add an initial registration step using RANSAC (Random Sample Consensus) before the ICP fine registration. We also modify the ICP algorithm to include texture features and more robust outlier rejections so that it

can handle objects without salient structural features. We detect the loop closure and perform global alignment to overcome the error-propagation problem. We also perform model denoising and texturing to remove the noise and give texture to the 3D object models. Our Chapter 4 and Chapter 5 deal with the noise and limited resolution issues of the depth images.

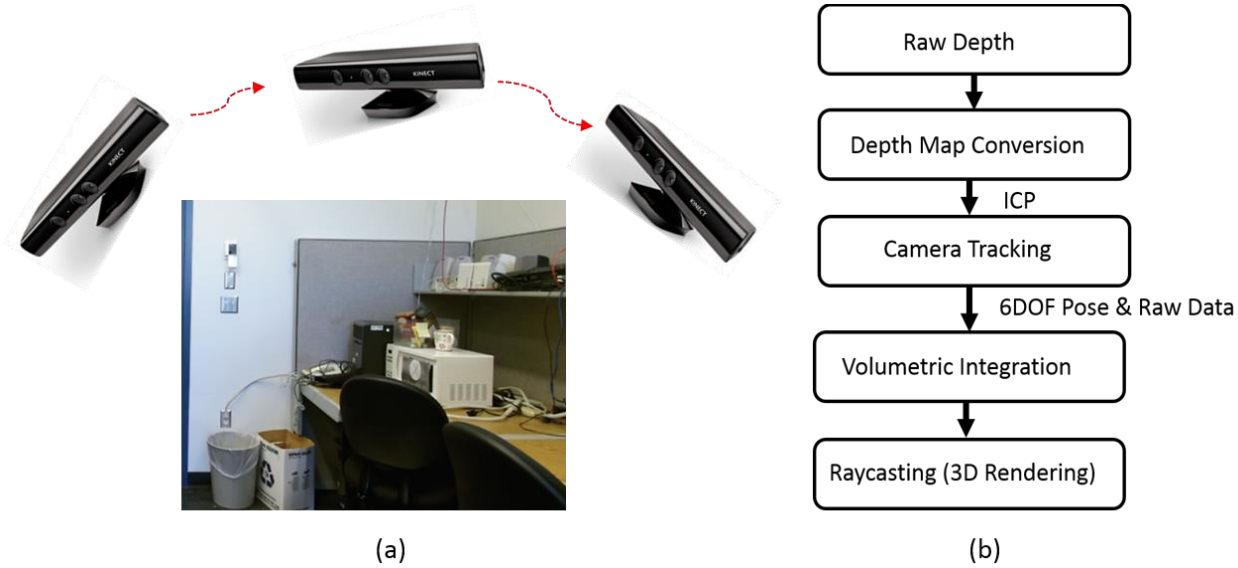


Figure 2-4. Overview of KinectFusion: (a) The setup for creating a 3D model using Kinect. A Kinect is slowly moved around the scene. (b) Flowchart of the KinectFusion algorithm.

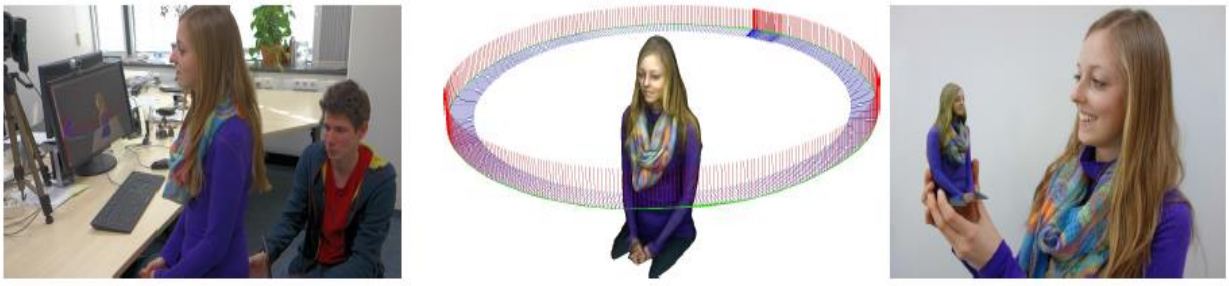


Figure 2-5. Overview of CopyMe3D. Courtesy of [7].

Chapter 3. 3D OBJECT MODELING WITH A KINECT CAMERA

3.1 BACKGROUND

3D object modeling is the process of developing a representation of the three dimensional surface of an object. A 3D model can represent a 3D object using a collection of points in the 3D space known as a 3D point cloud. These points can also be connected by various geometric entities such as triangles, lines, and curved surfaces. 3D models help to understand complex geometries that are hard to comprehend using 2D images. There are a plethora of 3D modeling applications. The medical industry uses 3D models of organs. The movie industry uses 3D models as characters for animation in motion pictures. The video game industry uses them for computer games. The architectural industry uses them to create replicas of proposed buildings and landscapes. The engineering community uses them to design new devices, vehicles, and structures as well as other uses. 3D models can also be the basis for physical devices that are built with 3D printers/copiers or CNC machines.

With the advent of economical depth cameras, such as Kinect, there is a significant interest in 3D modeling research. A notable system KinectFusion [1] and its variants [2, 3] create 3D reconstructions of indoor scenes using only the depth data in real time. This system relies on scanning with the slow motion of the Kinect camera and high cluttered environment to maintain its tracking to create 3D models of the scene. To obtain a 3D model, one has to scan the scene slowly and take more data in difficult locations to fill the holes in those locations and maintain the stability of the image alignment in the 3D model. It relies on spatial and temporal averaging for denoising. Thus, in the situation when there is only a limited number of views are available, the denoising may not be effective. Also, when only a limited number of views are available, the

overlap region between adjacent views may be small which could cause problems for Iterative Closest Point (ICP) registration. Another problem is that since the sensor is moving slowly, during the process of capturing, deformable object movements could cause registration and reconstruction problems. Also, since ICP registration only uses structural information, it may encounter difficulties when objects lack salient structural features (e.g., symmetrical objects).

Our target applications vary from the applications of the above mentioned work. In many applications, only a limited number of views may be available due to various constraints. For example, in a photo-booth application, a user may capture multiple views of an object with multiple Kinect cameras simultaneously. Simultaneous data capture will prevent problems caused by deformable objects. Our focus is to reconstruct a high-quality 3D model from a relatively smaller number of views of the object. In these situations, due to the limited number of views, there can be a significant shift between two adjacent views of the object. This introduces additional challenges such as the need for a better alignment and denoising process to efficiently utilize the available data. The 3D object model should have both structure and color texture information. We are also interested in methods for creating high-precision 3D models with a relatively low cost for these applications.

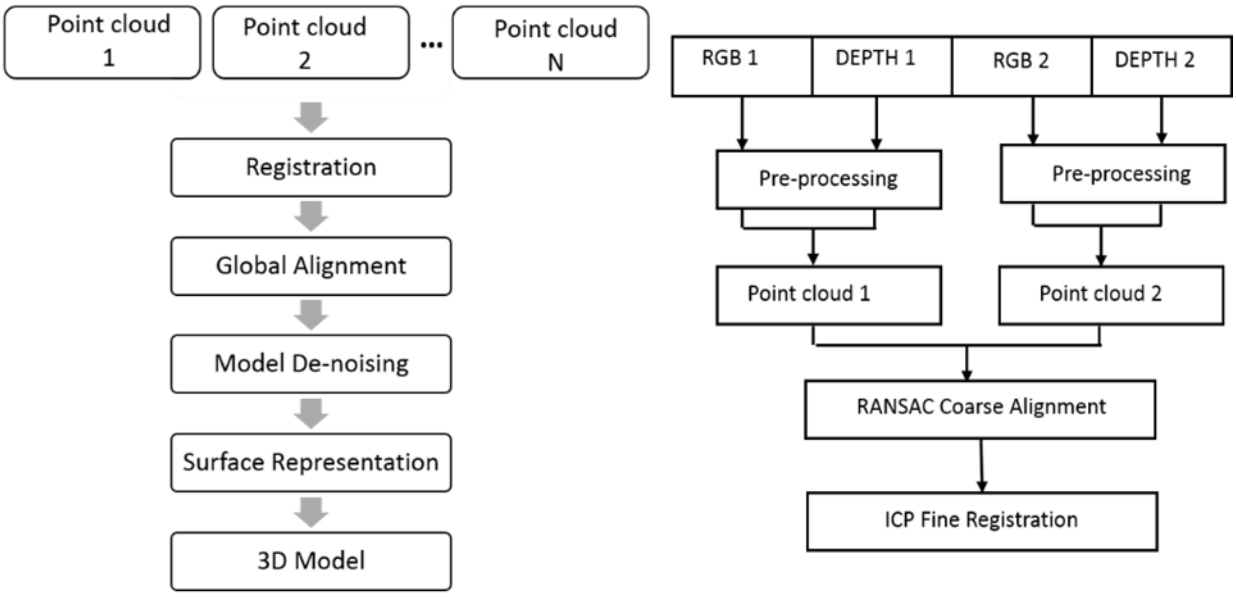
This chapter is organized as follows. In Section 3.2, we describe our proposed 3D object modeling framework. In Section 3.3, we discuss the denoising and the texturing for 3D object models. In Section 3.4, we discuss the evaluation of the 3D object models and compare the results obtained using our proposed approach with the results using laser scanners and other state-of-the-art approaches. Section 3.5 concludes this chapter.

3.2 3D OBJECT MODELING FRAMEWORK

Figure 3-1 shows a framework for the 3D object modeling process with an RGB-D camera such as Kinect. To simplify the automatic segmentation process, an object is placed in front of a green background. In the framework, first, the synchronized RGB and depth images of the object is captured using a Kinect from different angles. The foreground object is then segmented out from the background and represented in a 3D point cloud using the corresponding RGB and depth data. The partial 3D point clouds from different views are then registered together to form a complete 3D point cloud for the object. We will address various issues in each step.

There are many different approaches for registering the partial point clouds. In this work, we use a coarse-to-fine scheme for registration followed by a global alignment to prevent error propagation. In the coarse registration step, the partial 3D point clouds of the first two neighboring views of the object are aligned through a coarse registration algorithm such as RANSAC based on the SIFT feature matching [8, 9]. This initial registration is further refined by a fine registration algorithm such as ICP. The coarse alignment helps the convergence of the fine registration step to achieve stable and precise alignment. After the fine registration, we have the 3D point cloud combining these two views. The 3D point cloud of the next view is then registered with this combined point cloud. The process continues until the loop is closed, i.e., all the views from a complete scan of the object have been registered into a complete 3D point cloud. Due to error propagation, the last view of the object may not align very well with the 3D point cloud of the first view. We perform a global alignment to adjust the model to minimize the misalignment due to the error propagation. After that, the combined 3D point cloud model is de-noised to result in the 3D object model. The model can be transformed to other 3D representations such as polygon mesh,

volumetric representations [10], or Surfels [11] for different applications. In the following sub-sections, we provide details regarding the process.

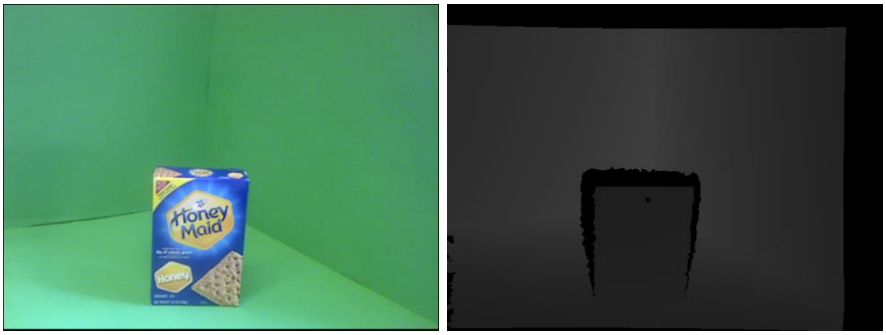


(a) (b)

Figure 3-1. (a) A flowchart of the Kinect based 3D object modeling process. Partial point clouds are registered to form the complete 3D point cloud. Global alignment is performed to align the last view with the first view. Model is then de-noised, followed by surface representation and color texturing. (b) The point-cloud generation and registration process.

3.2.1 *Object Segmentation from the RGB-D Images*

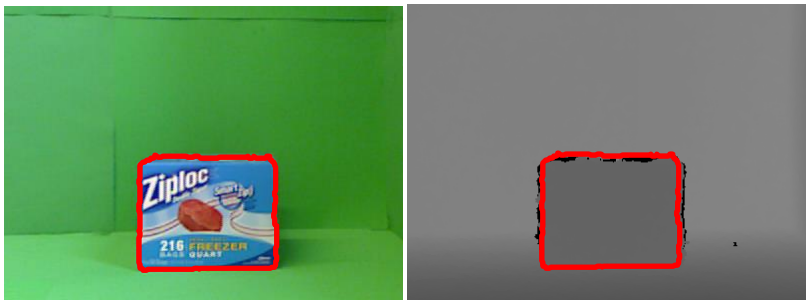
General object segmentation is an ill-posed problem. Since the 3D object modeling can be done in a controlled environment, we use color-based segmentation to get a good segmentation of the object from the RGB image. Specifically, we place the object on a green surface and in front of a green screen as illustrated in Figure 3-2, and implement an algorithm to automatically segment out the object from the RGB image.



(a) (b)

Figure 3-2. Object placed in the green background and the captured RGB (a) and Depth (b) images.

In the algorithm, we first convert the RGB image into the HSV color space. The HSV color space is a more appropriate color space for color based segmentation because hue component is close to the human visual system's perception of colors. We require the algorithm to be robust to changes in illumination. The need is to classify a pixel as foreground or background based on its intensity. A thresholding function $f(h, s, v) \rightarrow \alpha$ is applied to every pixel in the RGB image to obtain the object label. $\alpha = 0$ means the pixel is a background pixel, and $\alpha = 1$ means the pixel is in the foreground object. Morphological operations are performed to close small holes in the mask. This foreground object mask, which was obtained from the RGB image, is then applied to the depth image. Figure 3-3 illustrates an example of the segmentation result. With the cropped RGB and depth images, the 3D point cloud of one view can be generated.



(a) (b)

Figure 3-3. (a) and (b) show the RGB and depth images, respectively, with their object boundary highlighted, and the object boundary obtained from the segmentation mask.

3.2.2 *Registration*

In KinectFusion and related methods, initial pairwise alignment is not performed [1, 2, 3]. These methods use a video stream of RGB and depth data from the Kinect, which is capable of streaming data at 30 fps. Hence two adjacent frames are sufficiently overlapped with no significant changes in viewpoints. This eliminates the need for an initial alignment before the data is fed to the Iterative Closest Point (ICP) algorithm to align the two point clouds. However, in our case, we have frames with limited overlapping regions. ICP could perform poorly if these frames are directly fed to it. For this reason, we employ a coarse to fine registration procedure as described below.

3.2.2.1 Initial Registration

With the RGB and depth information, we first use the RANSAC algorithm to perform a coarse alignment. In the coarse alignment procedure, point clouds from different views are roughly registered based on the SIFT features in the RGB image.

Given the RGB and depth images of two views from the Kinect camera, we obtain two 3D point clouds $p = \{p_1 \dots p_N\}$ and $q = \{q_1 \dots q_M\}$, where *N* and *M* are the numbers of points in the two 3D

point clouds, respectively. The SIFT feature points are extracted from the two RGB images. After the initial alignment provided by the RANSAC process, we find the set of SIFT feature correspondence 3D points as $cf = \{(pf_1, qf_1) \ldots (pf_L, qf_L)\}$ where pf_i and qf_i are the corresponding SIFT feature points in p and q, respectively, and L is the number of matched SIFT feature point pairs.

3.2.2.2 Fine Registration of 3D Point Clouds Fusing Structural and Photometric Information

The second phase of registration is the ICP based fine registration. The standard ICP algorithm aligns two point clouds by iteratively associating points through a nearest-neighbor search and estimating the transformation parameters using a mean square cost function [5]. Define $T^{(k)}$ as the transform matrix after the k^{th} iteration. $T^{(0)}$ is the transform after RANSAC and before the ICP iteration.

For the k^{th} iteration (k=1, 2 …) in the fine registration process:

For each point p_i in the point cloud p, find its corresponding point $q_i^{*(k)}$ in q with the closest distance:

$$q_i^{*(k)} = \arg\left[\min_{q_j \in \{q\}} \left(\left\| p_i \cdot T^{(k-1)} - q_j \right\|\right)\right] \tag{3.1}$$

Then the following error function is minimized to get the optimal transform matrix T.

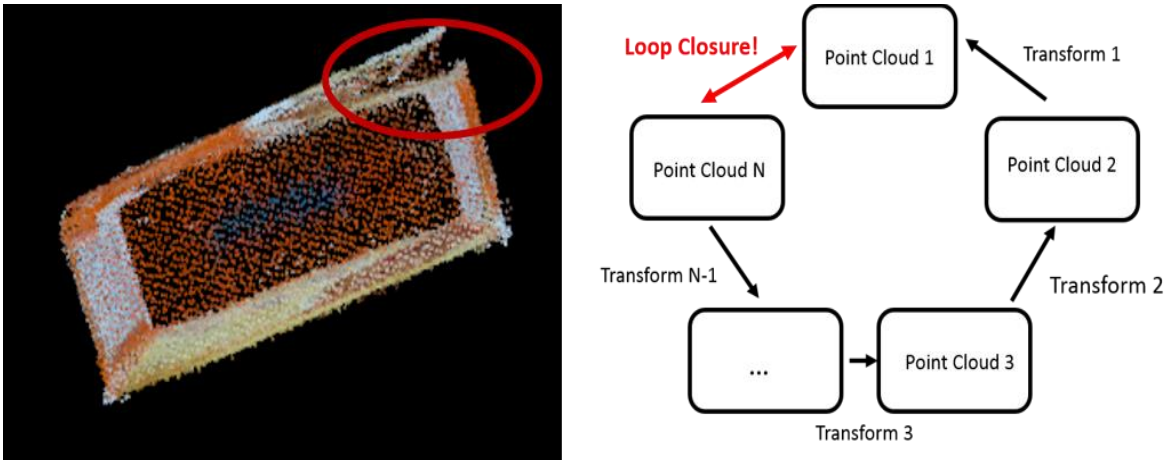
$$f(T) = \sum_{p_i \in p} \alpha_i \left\| p_i \cdot T - q_i^{*(k)} \right\|^2 \tag{3.2}$$

While the ICP algorithm has been widely used for dense point cloud matching, it is limited in its ability to produce accurate results, especially in challenging scenarios involving objects lacking structural features or undergoing significant camera view changes. To address this problem, we used a fine registration algorithm we proposed [12]. A SIFT-based term is added to the cost function to overcome the problem associated with the case of objects lacking structural features. To utilize the structural information of the 3D points without intensive computation of the SIFT descriptor for every 3D point, a constraint involving the spatial distances of the SIFT feature corresponding pairs which are readily available in the iterations is added. This added term effectively constrains the convergence to the correct direction which minimizes the spatial distances of points with structural features and texture features. In addition to this constraint, a dynamic weight is used to properly balance the significance of structural and photometric terms. Moreover, since the initial registration is not perfect, some of the SIFT points may not be exactly paired. Thus, some of the correspondence pairs with large distances are rejected. This is achieved by the outlier rejection method which adaptively utilizes both the statistics of structural characteristics and the spatial distances of SIFT correspondence pairs.

3.2.3 *Global Alignment*

In the pairwise registration process, registration errors could accumulate as more point clouds are added to the model. Figure 3-4(a) illustrates the effects of this error accumulation. Slight inaccuracy at each pairwise registration step leads to a significant misalignment between the last and first frame. This error could be eliminated by a global alignment which is to detect the last frame of a loop closure, register the frame with its previous frame and the first frame, and distribute the error evenly amongst other frames. Figure 3-4(b) demonstrates the global alignment using the

loop closure constraint [13]. Suppose a camera has turned 360 degrees, the first frame and the last frame should have an overlapping region. Using this additional constraint, we can establish a global optimization process to minimize the mismatching.



(a) (b)

Figure 3-4. Global alignment (a) shows a 3D model obtained without the global alignment. The red highlight shows the result of error propagation when global alignment is not performed. (b) shows the loop closure situation.

Global alignment algorithms reported in [13] and [14] are able to align data when the scanning process is unknown. However, we know the scanning process used in our data acquisition process. We assume that the object is scanned 360 degrees in either clockwise or counter-clockwise direction. We use this information to simplify the global alignment algorithms presented in [13] and [14]. Specifically, we simplify the loop closure detection in Explicit Loop Closing Heuristic (ELCH) [14] using the constraint described above. Our implementation is summarized in Figure 3-5 and the algorithm is described in detail in the following sub-sections.

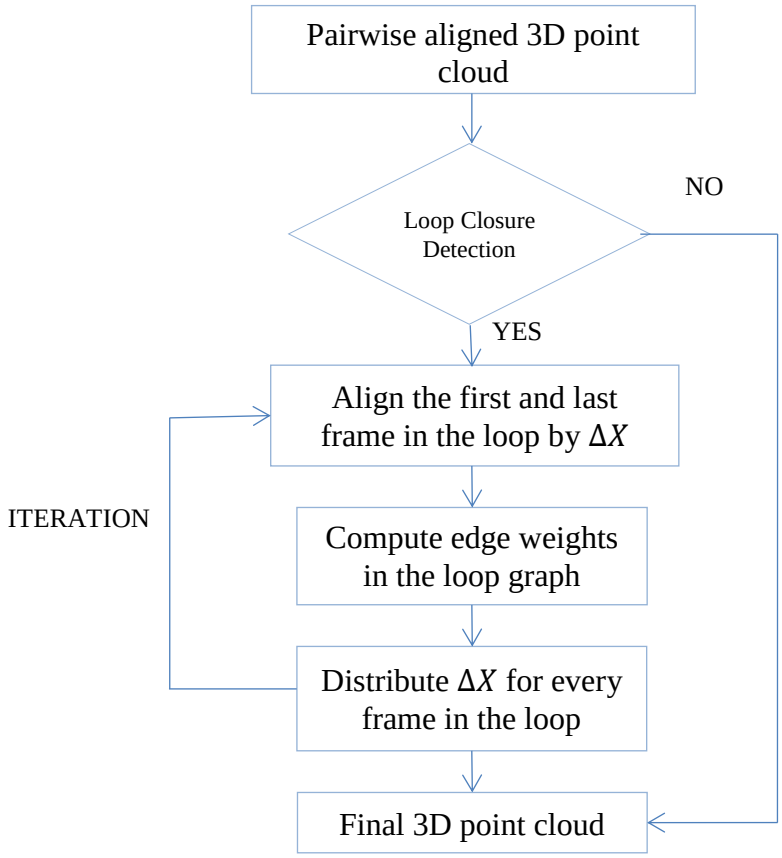


Figure 3-5. The global alignment procedure used to eliminate error accumulated due to the pairwise registration.

3.2.3.1 Loop Closure Detection

Our loop closure detection algorithm is based on the assumption that the camera scans the object in either clockwise or counter-clockwise direction. It should be noted that the algorithm in [14] does not assume the scanning direction. Our loop closure detection algorithm is described in Algorithm 3.1. Given the initial pairwise registration, we first compute the fraction of overlapped 3D points between any previous frame j for each frame i such that $j < i$. Loop closure occurs between two non-consecutive frames which have the largest fraction of overlapping 3D points than any other two non-consecutive frames. We use this criterion to detect the loop closure. As shown

in Figure 3-5, after loop closure is detected, we proceed with the global alignment procedure which is detailed in the following section.

Algorithm 3.1. Loop Closure Detection

Input: Pairwise aligned 3D points from N views. Set $i = 1 \ldots N$ (consistent with the scanning order).
Output: Loop closure pair frames *<start, end>*
for $i=1: N$

1. Compute the fraction f_{ij} which is the ratio between the 3D points in the overlapping region between frame *i* and any other frame *j* ($i < j$)) to the total 3D points in frame *i*.
2. if $f_{i,j} > f_{i,j-1}$ and $f_{i,j} > threshold$, add <*i*, *j*> into the loop closure candidates set *S*.

end
Pick up <*i*, *j*> in *S* such that $f_{i,j}$ is the largest.
$start \leftarrow i, \quad end \leftarrow j.$

Previous techniques have used the Euclidean distance between each frame and all previous frames along with a threshold of a minimal number of intermediate scans [14] to detect loop closures. The above algorithm is computationally less complex than computing the Euclidean distances between each frame and all previous frames.

3.2.3.2 The Global Alignment Procedure

When a loop closure is detected, we formulate a loop graph with each vertex representing one frame of the 3D point cloud. The edge between the successive frames is the pairwise pose transformation. We first use ICP to align the first and last frame with loop closure and get the transformation ΔX. If the pairwise registration does not have any errors, ΔX should be close to an identity matrix. Therefore, the non-identity ΔX can be regarded as the error of pairwise registration and can be distributed among all poses in the loop to minimize the loop closure mismatch. In order

to achieve a consistent map, we need to calculate the weights for the vertices that specify the fraction of ΔX by which the transformation needs to be changed and the weights are computed as follows:

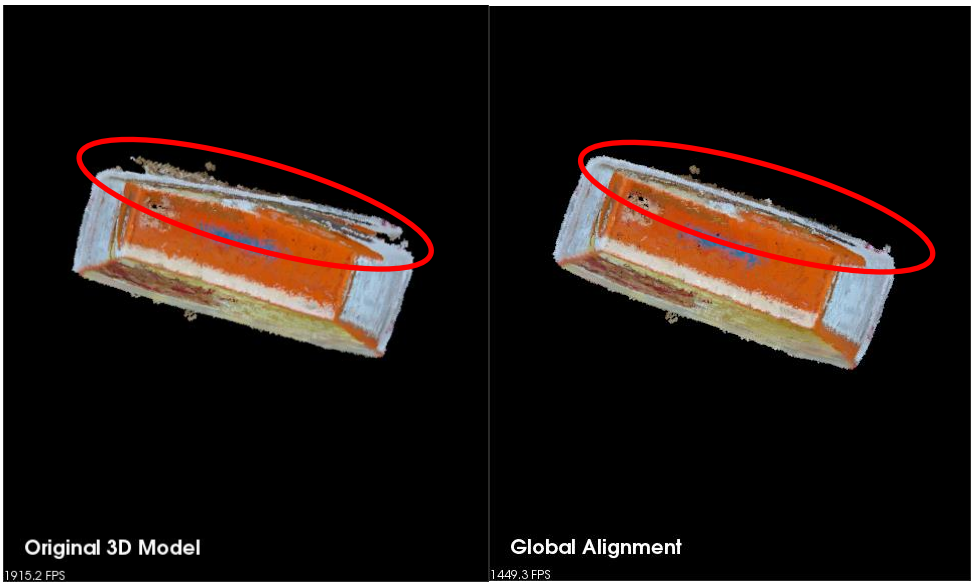
$$w_i = \frac{d(v_s,v_i)}{d(v_s,v_e)} \tag{3.3}$$

where v_s is the first vertex in the loop and v_e is the last one. $d(v_l, v_k)$ is the sum of the edge weights $c_{i,j}$ on the path from v_l to v_k. $c_{i,j}$ is the distance between v_i and v_j. For each point cloud, using the Spherical Linear Interpolation (SLERP) model [15, 16], we can transform the optimization problem into *k* dimensional (*k=4*, i.e., three for the position and one for the quaternion describing the rotation) sub-problems and compute the corresponding fraction w_i of the transformation. The final modified transformation for each frame *i* after the optimization is:

$$T_i = \Delta X \cdot w_i \tag{3.4}$$

The whole process is repeated several times until the convergence is achieved.

Figure 3-6 shows the 3D modeling result before and after our global alignment. The global alignment has eliminated the discrepancy between the first and the last frames used to create the 3D model.



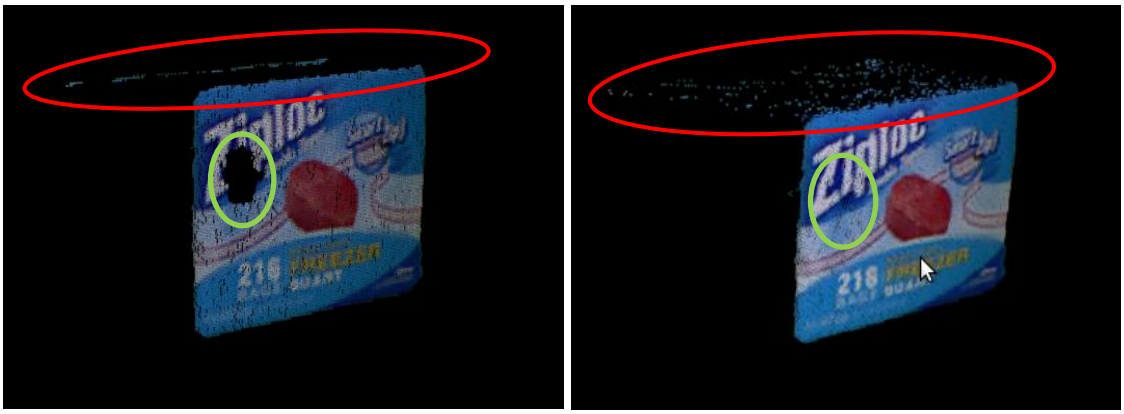
(a) (b)

Figure 3-6. Modeling result after the global alignment. (a) Before the global alignment (after pairwise registration) (b) After the global alignment procedure.

3.3 Model Denoising and Texturing

3.3.1 *3D Object Model De-Noising*

After the registration and the global alignment, the 3D model is well registered. However, it is still very noisy. Due to the inaccurate depth acquisition from the depth camera, the noises make the points deviate from their original positions when projected to 3D. Therefore 3D point cloud de-noising is needed to refine the 3D object model. It should be noted that although we have tried to refine the depth image in the pre-processing stage using a joint bilateral filter [17], it actually introduces unwanted noises in the 3D space. Figure 3-7 shows the effects of depth image de-noising in 3D projection. Although holes are filled up in the 3D view, unexpected noises appear around the edges of the flat plane.



(a) (b)

Figure 3-7. Effects of depth image de-noising in 3D projection. (a) 3D point cloud before de-noising the depth image. (b) 3D point cloud after de-noising the depth image using a joint bilateral filter.

It is evident from Figure 3-7, that filtering of the depth map may introduce extra noise. Hence we need to denoise the 3D object model. In this work, we use the Moving Least Squares (MLS) 3D model denoising method [18]. MLS uses the assumption that on a 3D surface, the 3D point set defines a manifold. The purpose of MLS is to find the manifold that the input 3D point set defines. Based on the manifold assumption, we can approximate the MLS surface by a function.

Let $p_i \in R^3, i \in \{1, \dots N\}$ be points in the noisy 3D point cloud. The goal is to project the points onto a two-dimensional surface that approximates p_i. For each small local neighborhood in the point cloud, we fit the points by a local tangent plane H. Denote the height of p_i over H as ph_i, we can estimate a polynomial approximation g so that the weighted squares error:

$$\sum_{i=1}^{N}(g(x_i, y_i) - ph_i)^2\theta(\|p_i - h\|) \tag{3.5}$$

is minimized, where θ is a smooth monotonically decreasing function, which is positive on the whole space (e.g., Gaussian approximation in our implementation). $(x_i,\ y_i)$ is the coordinate of

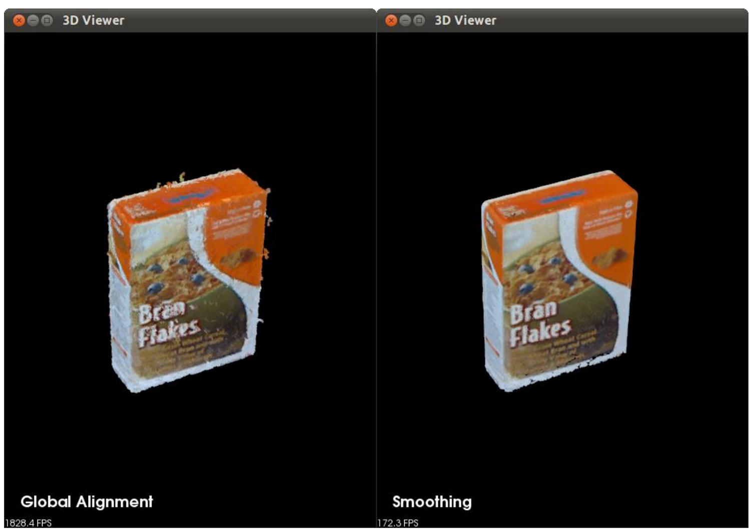
t

h

e

p_i's projection on H. After minimizing $i=1N(g(x_i,y_i)-f_i)2\theta(p_i-q)$ (3.5), coefficients of the polynomial function g can be computed. The projection approximation process is run for every point in the 3D point set and it has been proved that it can nicely preserve the manifold property [18].

Figure 3-8 shows the result of the 3D model before and after MLS de-noising. From the results, we can see that the surface of the 3D box model is much smoother and the outer noises have been significantly removed.



(a) (b)

Figure 3-8. Modeling result of MLS based de-noising. (a) Before 3D de-noising (after registration). (b) After MLS de-noising.

3.3.2 *Meshing and Texturing*

The initial 3D model is represented in point clouds. In many situations, we also like to represent the 3D models in other representations and with texture. Performing meshing and texture mapping can make the model resemble the real-life object. For this purpose, we triangulate the point clouds to build a mesh surface and then map the RGB color from the RGB image to the mesh.

The built 3D point cloud based model can be converted to other 3D representations. For our 3D modeling application, we use the Delaunay triangulation method [19] to convert the 3D point clouds into meshes. A Delaunay triangulation for a set P of points in a plane is defined as a triangulation $DT(P)$ such that no point in P is inside the circumcircle of any triangle in $DT(P)$. Delaunay triangulations maximize the minimum angle of all the angles of the triangles in the triangulation; they tend to avoid skinny triangles. Figure 3-9 shows a Delaunay triangulation with circumcircles shown. When points are connected to from Delaunay triangles, we can interpolate space inside the triangles to get a locally smoothed surface.

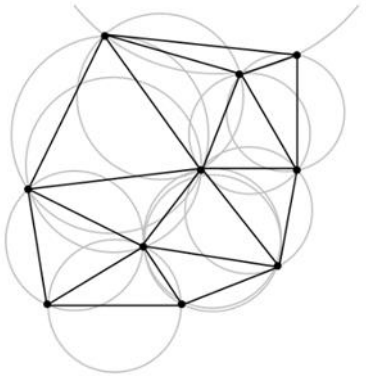
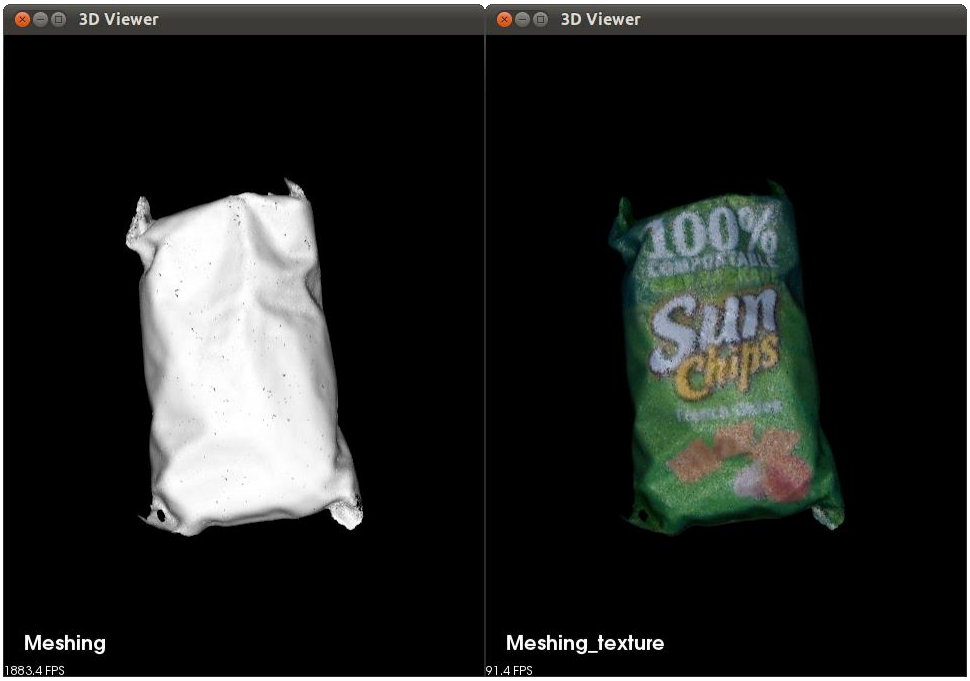


Figure 3-9. A Delaunay triangulation in the plane with the circumcircles shown.

After meshing, we assign the color to each vertex and simply interpolate the color in each triangle faces. This texture mapping requires that the resolution of triangulation should be high enough so

that interpolation will not introduce many artifacts. Figure 3-10 (a) shows the meshing results of a chip bag by using the Delaunay triangulation algorithm. Figure 3-10 (b) shows the meshing results of the chip bag after texture mapping.



(a) (b)

Figure 3-10. 3D Modeling result of meshing and texturing. (a) Meshing (b) Texture Mapping.

3.4 Evaluation of 3D Kinect Models

3.4.1 *Comparison of Kinect Model Point Cloud with the Point Cloud of a High-End Laser Scanner*

To investigate the error in a 3D model obtained using Kinect data, a comparison was made with a point cloud obtained by a high-end laser scanner. The 3D model point cloud was obtained from the RGB and depth images of a Kinect V1 or Kinect V2 sensor using the above-outlined procedure.

The laser scanner point cloud was obtained from the same object by a calibrated Roland Picza LPX-600 laser scanner. The Roland Picza LPX-600 laser scanner is a compact and enclosed scanner that produces high-resolution point cloud of objects using a non-contact red laser and high-quality optics to capture parts. The nominal range accuracy of the laser scanner is +/-0.05 mm [20]. It was therefore assumed that the laser scanner point cloud is sufficiently accurate to serve as a reference to evaluate the Kinect point cloud. In the absence of any systematic errors, the mean square of discrepancies between the two point clouds is expected to be close to zero.

3.4.2 *Error Calculation*

We propose the following methodology to calculate the error between the 3D model constructed and the 3D point cloud obtained from the laser scanner which is used as the ground truth. One difficulty for evaluation is that the 3D point cloud of the object model may contain much fewer points compared to that from the laser scanner. Also, the points in the 3D point cloud of the object model may be at a different location from those from the laser scanner.

Assume the 3D object model contains R_1 points, and the ground truth 3D point cloud contains R_2 points. We first randomly sample R points from the ground truth 3D point cloud and the 3D object model, such that:

$$R = \min(R_1, R_2) \tag{3.6}$$

To enable this analysis, first, an accurate registration of the two point clouds is necessary. The registration accuracy is important because any registration error may be misinterpreted as an error in the Kinect 3D object model. We use the coarse to fine registration methodology described before

and illustrated in Figure 3-11 to align the two point clouds. First, RANSAC is applied to the point clouds of the two 3D models to coarsely align the two point clouds. Then we use the proposed improved ICP registration to achieve an accurate fine registration. Now the misalignment between the point clouds represents the error between the two point clouds. This error is quantified using Root Mean Square Error (RMSE). The RMSE is calculated as:

$$RMSE = \sqrt{\frac{1}{R}\sum_r [p(x,y,z) - q(x,y,z)]^2} \tag{3.7}$$

where $p(x,y,z)$ and $q(x,y,z)$ are corresponding points in the Kinect 3D object model point cloud and the laser scanner point cloud respectively after alignment. Since the points from the laser scanner are random sampled, we repeat the process five times and average the resultant five RMSEs to get a consistent result. Figure 3-11 and Figure 3-12 illustrate the error calculation process using two example point clouds.

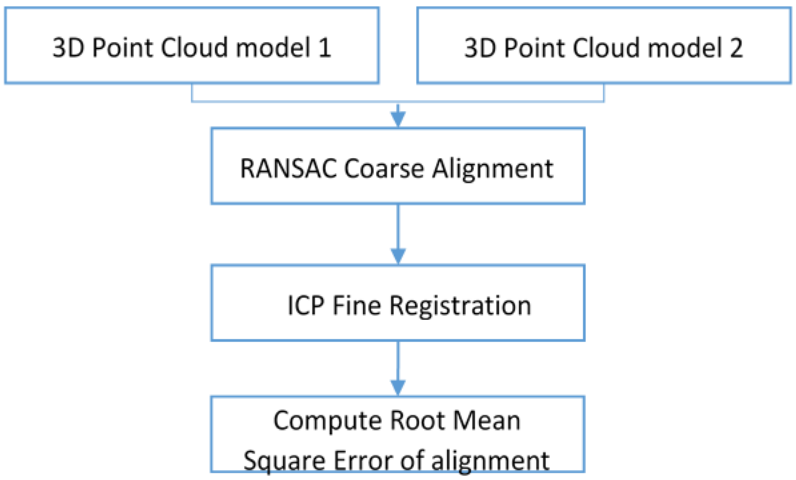
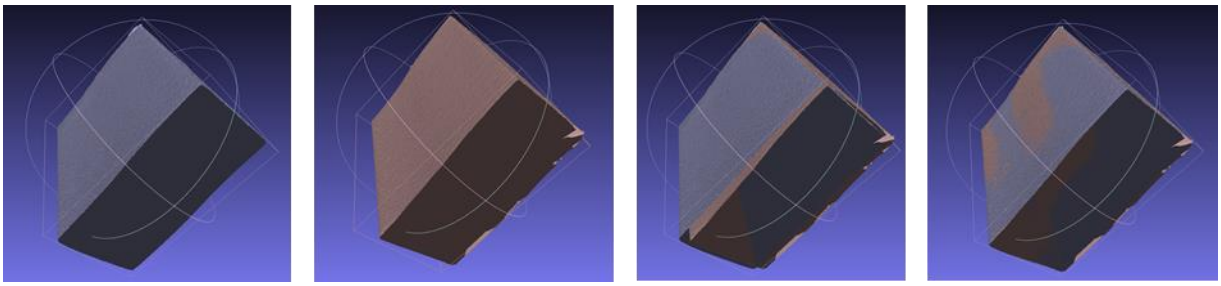


Figure 3-11. Methodology for calculating the error between two models.



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

Figure 3-12. An illustration of error calculation methodology. (a) and (b) are point cloud representations of two 3D models. (c) Result obtained by applying RANSAC. (d) Result obtained by applying ICP fine registration on (c). Note the error between the two models.

In order to evaluate this error calculation methodology, a subset of p points from the laser point cloud is randomly sampled. The same point cloud is again randomly sampled to get another subset of p points. Each 3D model has a varying number of points. Hence p is chosen to be at least 80% of the total number of points in a point cloud. The error between these two randomly subsampled point clouds are calculated. Since these two point clouds are subsets of the same initial point cloud the error between these point clouds should be close to zero. We use the four objects shown in Figure 3-13 to evaluate the results. The results obtained in this experiment are given in Table 3-1. One can observe that the RMS error is indeed close to zero, thus validating the proposed error calculation method.

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

Figure 3-13. Objects used for evaluating the results (a) Biscuit box, (b) Nut container, (c) House model, and (d) Mug.

Table 3-1. The root mean square error between two randomly subsampled point clouds from the same laser point cloud.

Object	**Root Mean Square Error (mm)**
Biscuit box	0.270
Mug	0.581
House model	0.440
Nut container	0.142

An example of the measured errors of the objects using our proposed process compared with KinectFusion using a Kinect V1 camera is shown in Table 3-2. As can be seen from the table, the results obtained using our proposed approach is, in general, more accurate. Most importantly, when the number of frames of an object is relatively small, KinectFusion shows convergence problems and has difficulties in constructing the 3D object model. While the proposed approach shows no problem in converging.

Table 3-2. Comparing results with KinectFusion for 3D object model construction using Kinect V1.

Number of frames	**Average RMSE (mm)**							
	Biscuit Box		**Nut Container**		**House Model**		**Mug**	
	KinFu	**Our 3D model**	**KinFu**	**Our 3D model**	**KinFu**	**Our 3D model**	**KinFu**	**Our 3D model**
65	3.284	2.597	3.939	1.724	2.974	3.088	3.025	3.183
55	4.362	2.732	5.237	1.801	3.482	3.228	3.892	3.259
45	Failed	2.863	Failed	1.932	3.927	3.362	4.823	3.468
35	Failed	2.959	Failed	2.066	Failed	3.485	Failed	3.621
25	Failed	3.099	Failed	2.189	Failed	3.609	Failed	3.798
15	Failed	3.298	Failed	2.311	Failed	3.784	Failed	3.918

We also perform the experiments with Kinect V2. The results are shown in Table 3-3. The results are more accurate and similar conclusions can be drawn.

Table 3-3. Comparing results with KinectFusion for 3D object model construction using Kinect V2.

Number of frames	**Average RMSE (mm)**							
	Biscuit Box		**Nut Container**		**House Model**		**Mug**	
	KinFu	**Our 3D model**	**KinFu**	**Our 3D model**	**KinFu**	**Our 3D model**	**KinFu**	**Our 3D model**
75	0.93	0.44	0.86	0.49	1.05	0.45	1.19	0.51
60	2.55	0.59	2.46	0.59	Failed	0.57	Failed	0.73
45	Failed	0.61	Failed	0.67	Failed	0.71	Failed	0.85
30	Failed	0.79	Failed	0.95	Failed	0.89	Failed	0.99
15	Failed	0.98	Failed	1.23	Failed	1.14	Failed	1.35

3.5 CONCLUSION

In this chapter, we present our work on 3D object modeling using a Kinect camera. We investigate the use of such cameras for acquiring RGB and depth images of small objects from a limited number of viewpoints and building a complete 3D model of the object. We address problems such as fine registration, global alignment, texture mapping, denoising, and objective error measurement of the 3D object models. For future work, we will investigate various denoising and other techniques to further improve the 3D models.

Chapter 4. OBJECT BOUNDARY BASED HOLE-FILLING AND DENOISING FOR KINECT DEPTH IMAGES

4.1 BACKGROUND

Popular commercial RGB-D cameras are able to simultaneously capture color images and depth images at an economical price. However, the quality of the depth images is degraded by various holes and noises which may occur in the depth images along the boundaries of objects, and in smooth, shiny, reflective or dark surfaces. For example, a Kinect depth image and its associated color image is shown in Figure 4-1. Shadow holes caused by the displacement between the IR projector and IR sensor, generally occurs along object boundaries in depth images as highlighted in red in Figure 4-1. Additionally, depth sensors often are unable to accurately estimate the depth values of smooth, shiny, reflective, and dark surfaces because IR rays reflected from these surfaces are weak or scattered. This results in random noises in depth images as highlighted in green in Figure 4-1, hence, making the raw depth images unsuitable for various 3D applications.

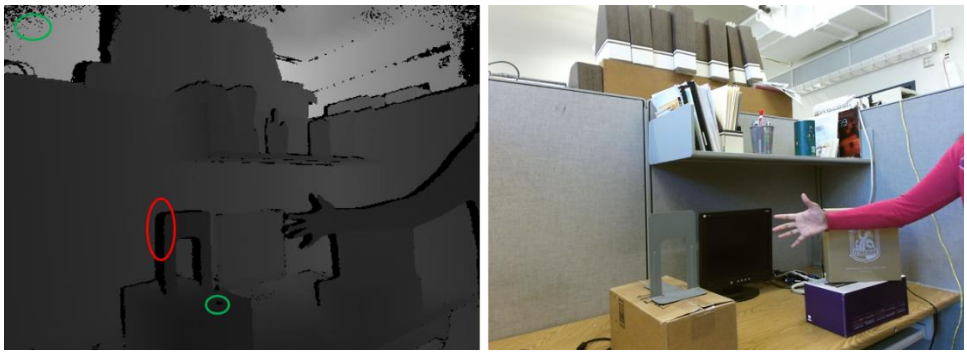


Figure 4-1. A depth image and its corresponding color image obtained from a Kinect camera. A sample of a shadow hole is highlighted in red and a random noise is highlighted in green.

Depth hole filling and denoising is challenging because it needs to accurately estimate depth while preserving depth discontinuities around object boundaries. In the process, the accurate detection of the object boundary plays a critical role. Researchers have been using information contained within the depth image and the color image to rectify errors in the depth image. The problem with using color edges to fill hole-pixels is the appearance of texture copy on the recovered depth image and the inability to distinguish depth edges between objects with similar color. The problem with using only depth is the potential to smooth object boundaries. Many approaches for denoising and hole-filling for depth images have been proposed in the literature. They fall into one of the following broad categories: 1) filtering based [21, 22], 2) plane-fitting based [23, 24], and 3) probabilistic methods [25]. Filtering based approaches use variations of bilateral filters to denoise the depth image. However, due to the false edges from the texture of color images, illumination of the scene, or weak edges due to the similar color of the object and the background, the bilateral filters or the variations of thereof may not produce the best result. Authors of [26] fill hole-pixels by iteratively applying a joint bilateral filter based on color intensity, the depth value, and temporal consistency. The method relies on depth values, not the depth edges. This leads to blurring the depth edge when the foreground object has a similar color as the background. Also, the Gaussian kernels used in the bilateral filter could cause some blurring due to the effect of pixel values on the opposite side of an edge. Authors of [21, 27] propose using the color image as a guide to fill hole-pixels, which lead to blurring and texture copy effects. Authors of [28] propose a trilateral filter that takes spatial distance and structural similarity into account. The trilateral kernel is expensive to compute and smoothed some object boundaries. Plane fitting based methods such as [23, 24] fit planes to neighborhood pixels to estimate the hole-pixels. However, they assume that the depth image consists of smooth slowly varying depth values inside the holes without

considering the depth discontinuities inside the holes from object boundaries. Thereby, these methods fail when there are object boundaries inside the holes and in complex scenes consisting of multiple objects with complex geometries. For an example of the probabilistic methods, an autoregressive model with color images is proposed in [25]. Though this method produces qualitatively good results in large structures, it blurs smaller object boundaries.

The main contribution of this work is that, compared to previously published works which mostly focus on using bilateral filtering to fill the holes, we propose an approach which focuses on constructing edges which correspond to the true object boundaries and using them to guide the hole filling process. We propose a novel depth edge estimation algorithm which estimates depth edges using information from the color edges and noisy depth edges by observing the statistics of the neighborhood depth pixel values and the corresponding color image. Inside the hole regions, we rely on edges from color images to recover the sharp object boundaries. We use adaptive thresholds in the edge detector to detect faint color edges in the hole regions. This enables the algorithm to find edges even when the foreground object and background have a similar color. We differentiate the holes generated by random depth errors and the holes that may contain object boundaries, and process them differently to produce better results. It produces a clean depth edge image with edges corresponding to the true object boundaries which are important to guide the hole-filling and denoising process. This refined depth edge image can be used in various other applications, such as depth denoising and object segmentation. In the hole-filling process, we use a trilateral filter to prevent pixels from the opposites of an object boundary being used as reference pixels which could cause the depth edges to be blurred. We also use an adaptive window size for the trilateral filter to make sure there are enough available depth values in the support region to generate reliable interpolations. Experiments show that our proposed edge detection algorithm can

extract edges without including non-boundary edges caused by the texture or illumination variation in the color image and edges caused by holes or noisy pixels in depth images.

The rest of this chapter is organized as follows. Section 4.2 describes the proposed method in detail. Section 4.3 shows the quantitative and qualitative advantages of the proposed method using various datasets. Finally, Section 4.4 concludes the chapter.

4.2 PROPOSED METHOD

To recover the pixels in the hole-regions of depth images, we propose a method as demonstrated in Figure 4-2. The basic idea is to reliably estimate the depth edges which correspond to true object boundaries based on both color and depth information and then use them to guide the hole-filling and denoising process.

First, edges are extracted from the input color and raw depth images using the Canny edge detector. The edges extracted from a raw depth image include edges that correspond to true object boundaries and many false edges from random noise and holes in the raw depth image. The edges that coincide with true object boundaries are determined by using the presence of non-hole depth pixels in the local neighborhood of the corresponding raw depth image. These edges are preserved while the remaining edges caused by random noise and holes in the raw depth image will be removed. Edges in the hole regions of the raw depth image are extracted from the edges of the related color image. An adaptive threshold is used in the Canny edge detector to detect color edges so that even when the foreground and background have a similar color the edge can still be detected. Subsequently, the edge information obtained from the raw depth image and its related color image is combined to produce a refined depth edge image containing depth edges that correspond to the true object boundaries.

Second, the refined depth edge image is used as a guide in an edge and color aware trilateral filter with an adaptive window-size to fill the holes in the depth image. When a hole-pixel in the depth image is filled using the edge and color aware adaptive trilateral filter, only depth pixels on the same side of the object boundary are used. Therefore, object boundaries in the refined depth image are preserved because the trilateral filter does not perform Gaussian averaging across edges. The order of applying the trilateral filter to fill the hole could significantly affect the result. We determine the order based on the number of available pixels in the local neighborhood of the hole-pixels. The hole-pixels are filled from the perimeter of a hole and grown inward to completely fill the entire hole. The size of the local neighborhood used to fill a depth hole-pixel varies adaptively based on the number of non-hole pixels in its vicinity. The objective is to use more non-hole pixels in the trilateral filter, which leads to the estimated depth pixel values being less affected by random noise in the raw depth image. The process is described in detail in the following sections.

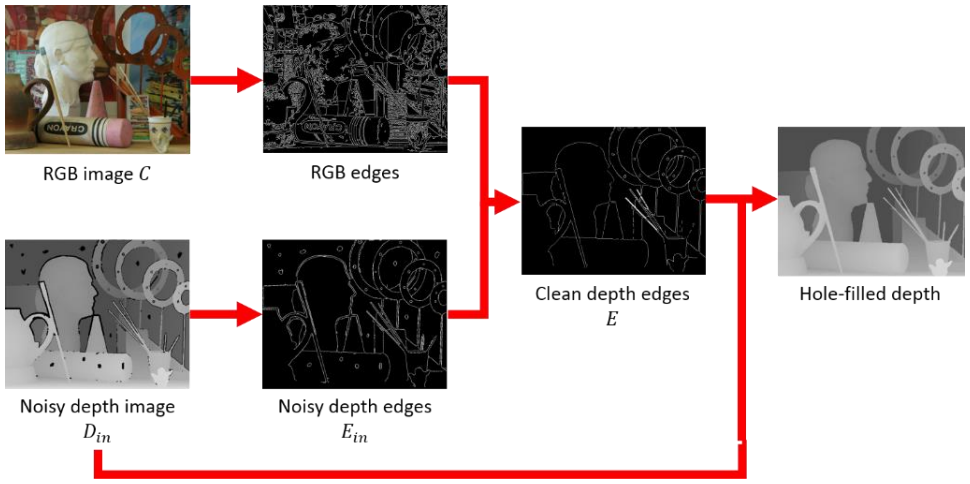


Figure 4-2. Illustration of the proposed method. Given a noisy depth image and the color image, the corresponding edge images are extracted. These edge images are used to produce a clean depth edge image, which along with the noisy depth image is used to produce the hole-filled depth image.

4.2.1 *Color edge extraction*

Edges in color images occur due to discontinuity in surface color, illumination or object boundary. The objective of this color edge extraction module is to be sensitive to the edges in the color image and be able to extract all possible edges in the image even when the foreground and background have similar colors. The subsequent processes will select the edges caused by the object boundaries and reject the edges caused by color texture and illumination changes.

The edges of the input color image were extracted using the Canny edge detector [29]. The first step of the Canny algorithm is calculating the gradient magnitudes and orientations of the Gaussian filter smoothed color image. Then local maximums of the gradient magnitudes along the gradient direction are detected as possible edge pixel candidates. In the third step, known as hysteresis, thresholding and linking of edge pixels are performed.

To reduce the localization error, we use a small standard deviation of 0.1 for the Gaussian filter. This reduces smoothing in the color image and extracts fine edge features in the color image. Also, in order to detect all possible edges, we use 0.01 and 0.1 for the lower and upper hysteresis thresholds respectively. These choices would yield many edges caused by texture and illumination changes in the scene. However, the edges not corresponding to the object boundaries would be suppressed later by using the depth image.

In the hole regions, we will be relying on color edges to recover the object boundary. Many previous algorithms have problems when the foreground object has a similar color as the background since the color edges will be faint and may not be detected by the edge detector. To overcome this problem, we use a threshold value proportional to the variance of the pixel color values inside the hole region. With this adaptive thresholding strategy, the faint edges can be detected inside the hole regions, without generating many false edges in other regions. Figure 4-3

shows an example input color image and the extracted edges from the color image using the adaptive Canny edge detector.

Figure 4-3. (a) Input color image (b) Edges extracted from the color image using the adaptive Canny edge detection

4.2.2 *Noisy Depth Edge Extraction*

Color edges correspond to structures and textures in the scene. Ideally edges in a depth image should only correspond to the structures in the scene. Nevertheless, raw depth images have many erroneous and missing depth pixel values caused by the hardware from which the depth images were collected and the physical properties of the objects in the scanned scene. These error sources give rise to holes and random distortions in the depth pixel values. These errored pixel values give rise to two kinds of inaccurate depth edges. The randomly distorted depth pixels give rise to random noise edges. The holes near object boundaries give rise to false edges around the perimeter of the hole instead of a single edge at the object boundary. Both these kinds of errored edges must be identified and removed from the depth edges.

A 5×5 2D median filter is applied to the raw depth image to filter out small random perturbations in the raw depth image. The raw depth edge image E_{in} is extracted using Canny edge detector [29] as shown in Figure 4-4. To extract fine edge details, a small standard deviation 0.1 for the Gaussian filter and low threshold values of 0.01 and 0.1 for hysteresis were used. The edges around the holes will be removed and the edges corresponding to object boundaries inside the holes will be restored from color edges in the next stage. Figure 4-4 shows an example noisy depth edge image and the edges extracted using Canny edge detector.

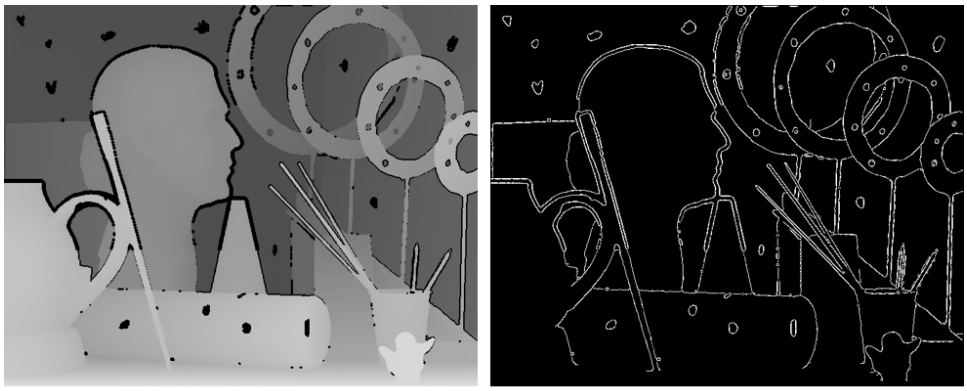


Figure 4-4. (a) Input noisy depth image (b) Edges extracted from the noisy depth image using Canny edge detector.

4.2.3 *Generating Clean Depth Edges*

The noisy depth edge image E_{in} obtained from the previous step has false edges caused by random noise and holes in the raw depth image, and true edges produced by object boundaries in the scene. These random noise edges were too large to be filtered out by the median filter in the noisy depth edge extraction process. The clean depth edges are produced in two stages. First, edges coinciding with object boundaries are identified and retained using the local depth value statistics in the raw

depth image. Other false edges are suppressed. Second, object boundaries in the hole regions of the raw depth image are extracted from the edges obtained from the corresponding color image.

The first step to removing the noisy depth edges is to identify edges corresponding to object boundaries in the raw depth image. To identify edges associated with boundary edges, it is observed that the depth edges which match with object boundaries in the scene are not in the vicinity of hole-pixels in the raw depth image. If there are fewer hole-pixels in the input depth image D_{in} corresponding to a neighborhood of an edge pixel location $E_{in}(p)$, this edge pixel is likely caused by the presence of an object boundary. Otherwise it is probably caused by erroneous depth values in the input depth image. Let I be a binary mask and I_q is the value of the mask at pixel q. $I_q = 1$ if q is a hole-pixel in the input depth image D_{in}, otherwise $I_q = 0$. Equation (4.1) is used to find edges caused by the presence of object boundaries.

$$E_1(p) = \begin{cases} 1, & if \ \sum_{q \in N_m(p)} G_q \cdot I_q < th, \ and \ E_{in}(p) = 1; \\ 0, & otherwise. \end{cases} \tag{4.1}$$

where G is an $m \times m$ Gaussian kernel. $E_{in}(p) = 1$ means p are pixels on edges in the noisy depth edge image E_{in}, th is a threshold, $N_m(p)$ are pixels in the $m \times m$ local neighborhood centered at p and E_1 contains the edges caused by object boundaries. G_q is the weight at location q with the Gaussian kernel G centered at p. $E_1(p) = 1$ indicates that p is on an edge caused by an object boundary. Gaussian kernel G with σ_m is used to spatially weigh the depth pixel locations within the $m \times m$ local neighborhood. By using a Gaussian kernel, a distant hole pixel from p would weigh less than a hole pixel close to p within the neighborhood. Thereby edge pixels which have distant hole pixels within the local neighborhood can be retained.

After identifying edge pixels caused by object boundaries, the remaining edges are false edges either caused by random noise or holes in the raw depth image. In step two of cleaning depth edges, false edges produced by random noise and holes in the depth image are removed. It is necessary to distinguish between random noise and holes that contain object boundaries in the raw depth image because the random noise holes may contain color edges which do not correspond to object boundaries. We would like to extract the true object boundaries that occur only in the object boundary hole regions of the raw depth image from the edges in the related color image. The variation of depth pixel values in the locale of false edges caused by random noise in the depth image is minimal. Whereas in the neighborhood of false edges caused by holes in the depth image, the variation of depth pixel values is generally higher when there is an object boundary within the hole. The variation, in this case, is higher because the depth values on the background side of an object boundary would be different from the depth values of the object. The variation of depth values is evaluated by fitting a plane to the local neighborhood of a false edge pixel in the raw depth image and computing the distances from each depth pixel in this local neighborhood to the fitted plane. If the variance of the computed distances is larger than a given threshold, the false edge pixel was likely caused by a hole with an object boundary in the raw depth image. These false edges are replaced by edges from the corresponding regions in the color image.

To evaluate the variation of the depth pixel values, we fit a plane in Equation (4.2) to the depth values in an $h \times h$ window centered at a false edge pixel p by using Equation (4.3).

$$z = ax + by + c \tag{4.2}$$

$$[\hat{a}, \hat{b}, \hat{c}] = \underset{[a,b,c]}{\operatorname{argmin}} \sum_{i \in N_h(p)} \|z(i) - ax(i) - by(i) - c\|^2 \tag{4.3}$$

where z is the depth value, x and y are the spatial coordinates of a depth pixel in the input raw depth image D_{in}. a, b and c are plane coefficients. $N_h(p)$ is the $h \times h$ local neighborhood of a depth pixel p. i is a pixel in $N_h(p)$. $x(i)$, $y(i)$ and $z(i)$ are the x, y location and depth values of depth pixel i respectively. The window size, h is adaptively chosen such that, h is the smallest window size in which at least 80% of the pixels in the $h \times h$ neighborhood are non-hole depth pixels. The adaptive window size ensures that sufficient number of non-hole depth pixels are taken into account to distinguish between the false edges caused by random noise versus holes in the raw depth image. The initial window size h is set to 21.

The perpendicular distance from a depth pixel to the fitted plane (Equation (4.2)) is calculated using Equation (4.4).

$$\Gamma_{\mathrm{p}} = \left\{ \gamma_i \mid \gamma_i = \frac{|z(i) - \hat{a}x(i) - \hat{b}y(i) - \hat{c}|}{\sqrt{\hat{a}^2 + \hat{b}^2 + 1}}, i \in N_h(p) \right\} \tag{4.4}$$

Where Γ_{p} is the set of perpendicular distances from all non-hole depth pixels in $N_h(p)$ to the fitted plane in Equation (4.2). γ_i is the individual distance from pixel i to fitted plane in Equation (4.2). Then, Equation (4.5) is used to identify the false edges produced by holes in the raw depth image D_{in}.

$$J_p = \begin{cases} 1, & if \;\; var(\Gamma_{\mathrm{p}}) > t; \\ 0 & otherwise. \end{cases} \tag{4.5}$$

where $var(\Gamma_{\mathrm{p}})$ is the variance of the set Γ_{p} and t is a given threshold. When the variance of the perpendicular distances from non-hole depth pixels in $N_h(p)$ to plane in Equation (4.2) is greater

than a threshold t, the false edge is classified to be generated by a hole in the depth image D_{in}. Otherwise, the false edge was produced by random noise in depth image D_{in}, and removed from further processing. J is the set of J_p, which is essentially a binary mask for the depth hole regions which contain depth pixel locations where there may be true object boundaries in the locality.

Next, to find object boundaries present in the hole regions of the raw depth image D_{in}, we elicit the edges from the corresponding color image. First, the color variance of the pixels in a hole region is computed. The hysteresis threshold to find the Canny edge of the color image is set proportional to the color variance in the hole region. Thus the threshold is set adaptive to the colors in the hole region. For example, in a hole region where the foreground and background are of similar colors, the variance of the color values in the region would be low. This would lead to a lower threshold value, enabling the detection of the subtle edges in image regions with similar colors.

Equation (4.6) is used to find the color edges in the depth hole regions J that likely are true object boundaries. The depth hole regions J with likely object boundaries are dilated with a 3×3 structuring element with all the elements equal to 1 and the origin at the center, to get J', to accommodate for the possibility that color edges may be close to a hole boundary. Edges produced by true object boundaries lie in the intersection of C and J'.

$$E_2 = J' \cdot C \tag{4.6}$$

where E_2 is the depth edge image corresponding to object boundaries extracted from the color edge image C and J' is the dilated depth hole regions where object boundaries are likely to be found.

The clean depth edge image E is the combination of edges obtained from Equations (4.1) and (4.6) as shown in Equation (4.7).

$$E = E_1 + E_2 \tag{4.7}$$

Figure 4-5 shows the depth edges E obtained from Equation (4.7) overlaid on the color and depth image, respectively.

Figure 4-5. Depth edges E from Equation (4.7) overlaid on the color image (a) and on the noisy input depth image (b).

4.2.4 *Depth Hole Filling*

Once the clean depth edges are recovered, we use an extended version of edge aware bilateral filter in [30] as described in Algorithm 4.1 to construct the recovered depth image D_{rec}. An edge and color aware adaptive trilateral filter in Equation (4.8) is used to fill the hole-pixels progressively from the perimeter of the hole to its center. The non-hole pixels outside the perimeter of the hole is used to estimate the hole depth pixels. The size of the local neighborhood $n \times n$ used to fill a

depth hole-pixel p varies adaptively to ensure that at least 80% of pixels in the neighborhood are non-hole pixels with initial $n = 7$. With this approach, more non-hole pixels are used in the trilateral filter than using a fixed neighborhood size. This leads to the estimated depth pixel values being less affected by random noise in the raw depth image. By successively using these recovered depth values and the clean depth edge image, the algorithm evaluates hole-pixels without blurring object boundaries and smooths non-object boundary regions of the image.

Algorithm 4.1. Depth hole-filling with edge and color aware adaptive trilateral filter

Input: Recovered depth edge image $\boldsymbol{E}$, input raw depth image $\boldsymbol{D_{in}}$
Output: Recovered depth $\boldsymbol{D_{rec}}$
Initialization: Push every hole-pixel in input depth image $\boldsymbol{D_{in}(p)}$ into a priority queue $\boldsymbol{Q}$ decreasing ordered by the percentage of non-hole pixels in a $\boldsymbol{n \times n}$ window centered at $\boldsymbol{p}$, initial $\boldsymbol{n = 7}$
While $\boldsymbol{Q}$ is not empty:
    $\boldsymbol{D_{in}(p) \leftarrow pop\ Q}$
    Update $\boldsymbol{D_{in}(p)}$**:**
        $\boldsymbol{num} \leftarrow$ percentage of non-hole pixels in $\boldsymbol{n \times n}$ window centered at $\boldsymbol{D_{rec}(p)}$
        If $\boldsymbol{num < 80\%}$:
            $\boldsymbol{n \leftarrow n + 2}$
            Push $\boldsymbol{D_{in}(p)}$ into $\boldsymbol{Q}$
        Else:
            Compute $\boldsymbol{D_{rec}(p)}$ using Equation (4.8)
        End
End
return $\boldsymbol{D_{rec}}$

The depth hole-pixels $D_{rec}(p)$ can be found as:

$$D_{rec}(p) = \frac{1}{k_p} \sum_{q \in N_n(p)} D_{in}(q) \cdot f_s(\|q - p\|) \cdot f_r(E, q, p) \cdot f_i(C, q, p) \qquad (4.8)$$

where $N_n(p)$ is a $n \times n$ window centered at pixel p. $f_s(\cdot)$ is a zero mean spatial Gaussian kernel with a standard deviation σ_d. k_p is a normalizing factor. $\|\cdot\|$ is the Euclidean distance measure. The range kernel $f_r(\cdot)$ is a binary indicator defined as:

$$f_r(E,q,p) = \begin{cases} 1, \ if\ the\ shortest\ path\ connecting\ pixels\ q\ and\ p \\ (excluding\ pixels\ q\ and\ p) \\ does\ not\ intersect\ an\ edge\ in\ E; \\ 0, \ otherwise. \end{cases} \tag{4.9}$$

The range kernel f_r assures that only pixels on the same side of an edge are Gaussian averaged to estimate a hole depth pixel value which preserves object boundaries in the recovered depth image. The shortest path from hole-pixel p to every other pixel q in the $n \times n$ window centered at p is computed. These shortest paths do not include end pixels p or q, rather the intermediate depth pixel locations on the path from q to p. Pixels q whose shortest path to p does not intersect an edge in E are on the same side of the edge as p. The intensity kernel $f_i(\cdot)$ is a binary indicator defined as:

$$f_i(C,q,p) = \begin{cases} 1, & if\ \|C(q) - C(p)\| < \varepsilon\ ; \\ 0, & otherwise. \end{cases} \tag{4.10}$$

Where $\|\cdot\|$ is the Euclidean distance measure. C is the color image with the R, G and B components normalized to $[0, 1]$ and ε is a threshold. The intensity kernel $f_i(\cdot)$ ensures that neighborhood depth pixels with similar color in the corresponding color image is used to fill a hole depth pixel. Specifically, a hole-pixel p utilizes only pixels in the $n \times n$ window with similar color to p as in Equation (4.10) to fill p. When p is an edge pixel, it is ambiguous which side of the edge does p

belong to. If p is an edge pixel, then a surrogate pixel p' is found using Equation (4.11). Surrogate pixel p' is a pixel in the 8 connected neighborhood of p that is not an edge pixel and has the closest color to p in the corresponding color image. The surrogate pixel p' is used to determine which side of an edge, pixel p belongs to. The color image C is used to find p' such that the color difference between p and p' is least amongst all non-edge 8 connected neighbor pixels of p.

$$p' = \min_{u \in N'(p)} \|C(p) - C(u)\| \tag{4.11}$$

where u is a pixel in the neighbhorhood $N'(p)$ such that it is one of the 8 connected pixels of p but is not an edge pixel. $C(p)$ is the RGB color value of pixel p in the corresponding color image C. Estimating the hole-pixels from exterior to the interior of a hole and using the above edge and color aware adaptive trilateral filter enables the method to smoothly fill depth hole-pixels in the non-edge areas while preserving the object boundaries. By using a varying local neighborhood, the method is able to fill large holes with depth values consistent with the rest of the image.

4.3 Experimental Results

In this section, we present results from evaluating our method with respect to other state-of-the-art methods. First, we used the synthetically degraded dataset introduced in [25] to quantitatively and qualitatively compare against ground truth data. Then we applied the proposed method on RGB-D images obtained from a Kinect V1 camera and present visual comparison results.

We set the parameters in all of our experiments to minimize Mean Absolute Error (MAE) as in Equation (4.12).

$$MAE = \frac{1}{N} \sum_{p \in \{P \mid D_{syn}=0 \cap D_{gt} \neq 0\}} \left| D_{rec}(p) - D_{gt}(p) \right| \quad (4.12)$$

where D_{rec}, D_{gt} and D_{syn} are the recovered depth, ground truth depth and synthetically degraded depth, respectively. N is the number of hole-pixels in the synthetically degraded depth that are not hole-pixels in the ground truth depth. MAE is calculated only in the synthetically degraded hole-pixels in D_{syn}, which means $D_{syn}(p) = 0$ but the ground truth depth image D_{gt} has valid depth pixel values. This enables us to quantitatively evaluate the algorithms' ability to recover synthetic hole-pixels in object boundary and smooth regions.

Parameters Setting: In Equation (4.1), $m = 9$, $\sigma_m = 1$ and $th = 0.055$. In Equation (4.5) $t = 0.09$. In depth aware hole filling Equation (4.8) σ_d is 1. In Equation (4.10) ε is 0.1.

4.3.1 *Results on Synthetically Degraded Dataset*

Yang et al. introduced a synthetically degraded dataset in [25] which consists of a subset of the Middlebury dataset [31] with random noise and structural holes created on the depth images of these data to imitate the holes in depth images produced by a Kinect. Figure 4-6 shows a sample of the images in the Yang dataset. To mimic Kinect depth images, structural missing pixels are created along depth discontinuities, and random missing pixels are generated in flat areas.

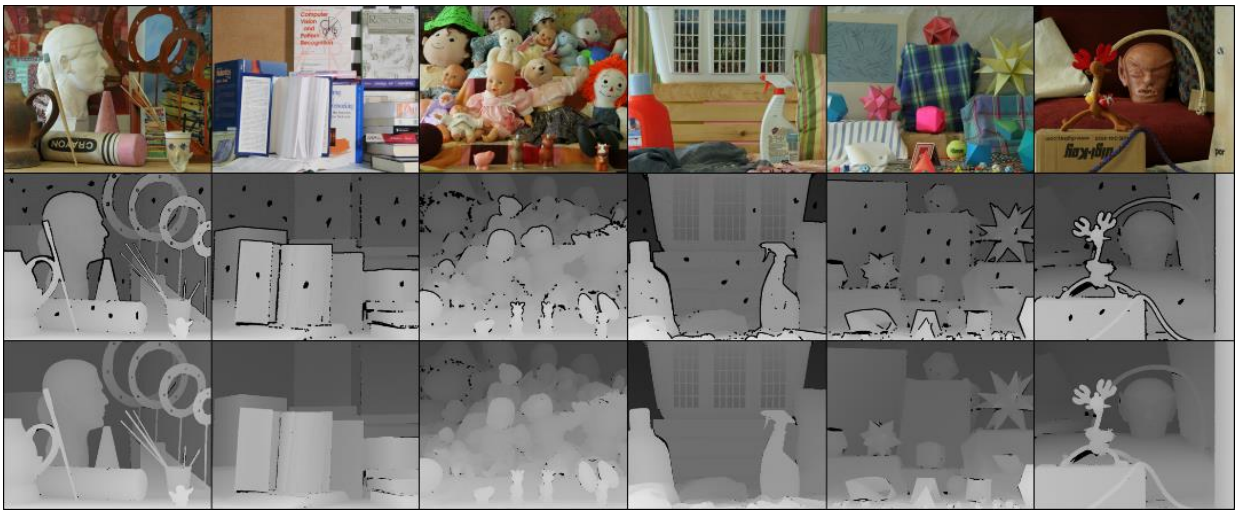


Figure 4-6. Thumbnails from the synthetically degraded dataset [25]. The top row shows the color images. The middle row shows the corresponding depth images degraded by random noise and structural holes. The bottom row shows the ground truth depth images.

The proposed method is compared to bicubic interpolation and four state-of-the-art methods to produce recovered depth images. The baseline methods are: JBF [32], Guide [21], CLMF [22] and AAR [25]. Recovered depth images are obtained using the baseline methods and the proposed method. The structure similarity index (SSIM) is calculated as described in [33]. The mean absolute error (MAE) is calculated as per Equation (4.12). Table 4-1 tabulates the quantitative results from comparing various methods. The proposed method gets the least MAE score on all the images. On average the method reduces the MAE error by 14.02%. Our method also gets better or comparable SSIM values on the dataset.

Table 4-1. Table lists the quantitative results comparing our method to the state-of-the-art methods tested on the synthetically degraded dataset [25] using Equation (4.12) to calculate Mean Absolute Error (MAE) and Structure Similarity Index (SSIM). Our method improves the best MAE for each image by 14.02% on average. Our method improves or on par with the other methods on the SSIM metric.

	Art		**Book**		**Dolls**		**Laundry**		**Moebius**		**Reindeer**	
Algorithm	**MAE**	**SSIM**	**MAE**	**SSIM**	**MAE**	**SSIM**	**MAE**	**SSIM**	**MAE**	**SSIM**	**MAE**	**SSIM**
AAR	4.78	0.987	1.99	0.991	1.91	0.988	3.14	0.982	2.40	0.989	2.12	0.990
Bicubic	7.45	0.984	2.30	0.991	2.12	0.986	3.81	0.980	2.65	0.988	3.05	0.987
CLMF	6.40	0.988	2.07	0.992	1.92	0.988	3.51	0.982	2.38	0.990	2.80	0.988
Guide	6.94	0.985	2.10	0.992	1.94	**0.989**	3.42	**0.985**	2.42	0.990	2.81	0.989
JBF	6.97	0.983	2.36	0.989	2.13	0.985	3.70	0.979	2.76	0.986	2.95	0.984
Ours	**4.67**	**0.988**	**1.61**	**0.993**	**1.79**	0.988	**2.50**	0.984	**1.89**	**0.991**	**1.81**	**0.991**

To test the proposed method's ability to reconstruct depth values in hole regions with object boundaries, we repeated the above experiment by calculating the mean absolute error using Equation (4.13).

$$MAE_{ob} = \frac{1}{N} \sum_{p \in \{P \mid D_{syn} = 0 \cap D_{gt} \neq 0 \cap J' = 1\}} \left| D_{rec}(p) - D_{gt}(p) \right| \qquad (4.13)$$

where D_{rec}, D_{gt} and D_{syn} are the recovered depth, ground truth depth and synthetically degraded depth, respectively. J' is a mask indicating the depth hole regions which have object boundaries. It is computed as described in Section 4.2.2. N is the number of hole-pixels in the synthetically degraded depth that are not hole-pixels in the ground truth depth but contained in J'. MAE_{ob} is

calculated only in the synthetically degraded hole-pixels in D_{syn} which have object boundaries inside the hole region, which means $D_{syn}(p) = 0$ and $M(p) = 1$ but the ground truth depth image D_{gt} has valid depth pixel values. MAE_{ob} enables us to quantitatively compare an algorithm's ability to estimate depth values in hole regions which have object boundaries relative to the ground truth depth value. Table 4-2 tabulates the quantitative results from comparing ours and the baseline methods. The proposed method gets the best results on all the images. On average the method reduces the MAE error by 36.78%.

Table 4-2. Table lists the quantitative results comparing our method to the state-of-the-art methods tested on the synthetically degraded dataset [25] using Equation (4.13) to calculate Mean Absolute Error (MAE). Our method improves the best MAE for each image by 36.78% on average.

Algorithm	**Art**	**Book**	**Dolls**	**Laundry**	**Moebius**	**Reindeer**
AAR	10.50	5.93	4.62	8.06	5.55	6.17
Bicubic	17.45	7.71	5.67	10.65	6.55	10.80
CLMF	14.57	6.84	5.08	9.85	5.80	9.93
Guide	16.08	6.96	5.13	9.53	5.93	9.91
JBF	16.35	8.00	5.69	10.15	6.92	10.17
Ours	**10.31**	**3.80**	**3.87**	**5.58**	**3.54**	**4.34**

Figure 4-7 and Figure 4-8 further illustrate the effectiveness of the proposed method in preserving depth edges. They show cropped and zoomed-in portions of the 'Art' and 'Book' images in the synthetically degraded dataset. Bicubic interpolation blurs the depth edge. Though the state-of-the-art methods are progressively better at retaining the edge boundary, our method produces the

least distorted edges. Guided [21] and CLMF [22] blur the object boundaries similar to bicubic. JBF [32] and AAR [25] produce jagged artifacts near the edges. Our method is able to fill holes while producing clean edges at object boundaries. Quantitative and qualitative experiments prove the strength of the proposed method.

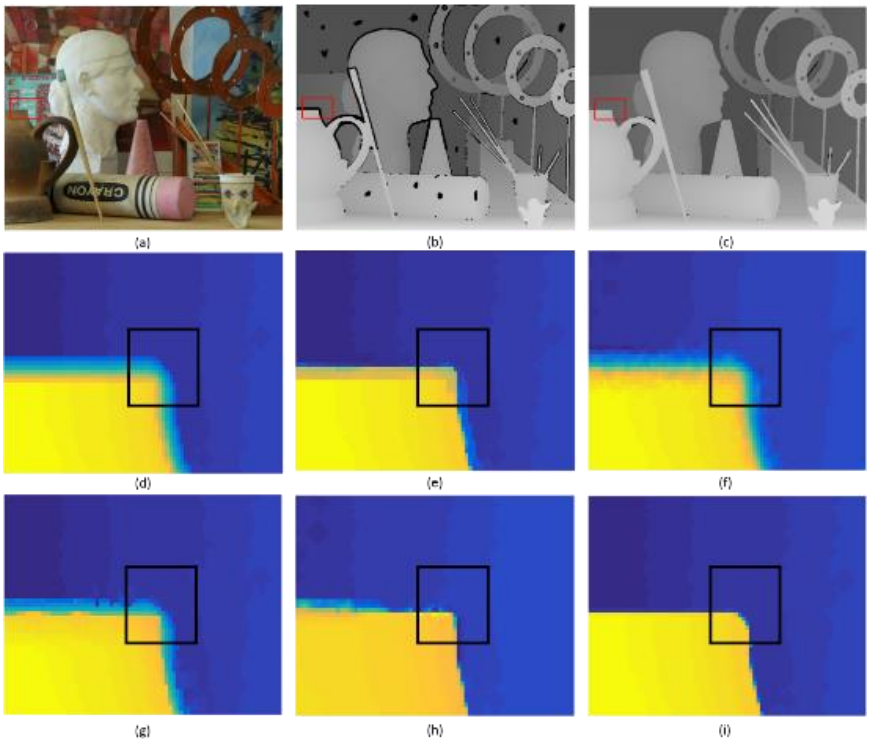


Figure 4-7. A visual comparison of results from the state-of-the-art methods on the 'Art' image. (a), (b) and (c) are the color, synthetically degraded depth, and the ground-truth depth respectively. (d) – (i) show zoomed in view of different methods in the red rectangle region in the 'Art' image. (d) bicubic (e) JBF [32] (f) Guide [21] (g) CLMF [22] (h) AAR [25] and (i) ours. Note that our method preserves the edge structures the best without creating a halo around the object boundary as shown in the boxes highlighted in black.

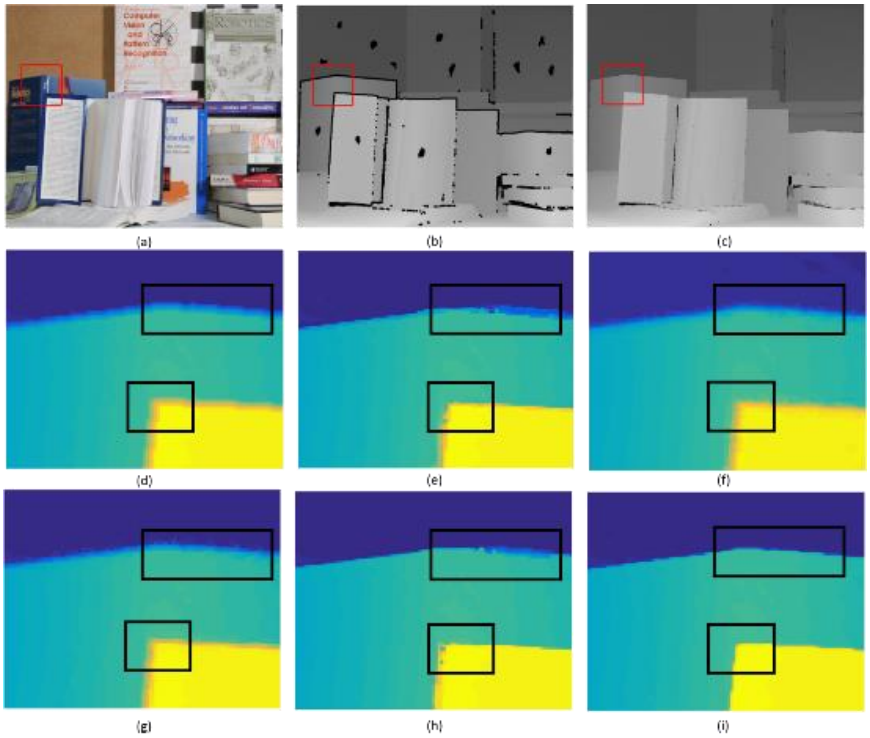


Figure 4-8. A visual comparison of results from the state-of-the-art methods on the 'Book' image. (a), (b) and (c) are the color, synthetically degraded depth, and the ground-truth depth respectively. (d) – (i) show zoomed in view of different methods in the red rectangle region in the 'Book' image. (d) bicubic (e) JBF [32] (f) Guide [21] (g) CLMF [22] (h) AAR [25] and (i) ours. Note that our method preserves the edge structures the best, there are no "orange" colored pixels in the colored recovered depth image as illustrated in the boxes highlighted in black.

4.3.2 *Results on Kinect dataset*

Next, we test the proposed method using color and depth data collected from a Kinect V1 camera. For this purpose, we use the Kinect data provided by [25]. The depth image was denoised using AAR, Guide and our method. Due to the lack of ground truth, we present only a qualitative comparison of the results. Figure 4-9 compares the AAR [25], Guide [21] and our method. The cropped and zoomed images show that the proposed method preserves the depth edges better than

comparison methods. Particularly, AAR and Guide blur and distort the object boundaries in the depth image as highlighted in Figure 4-9.

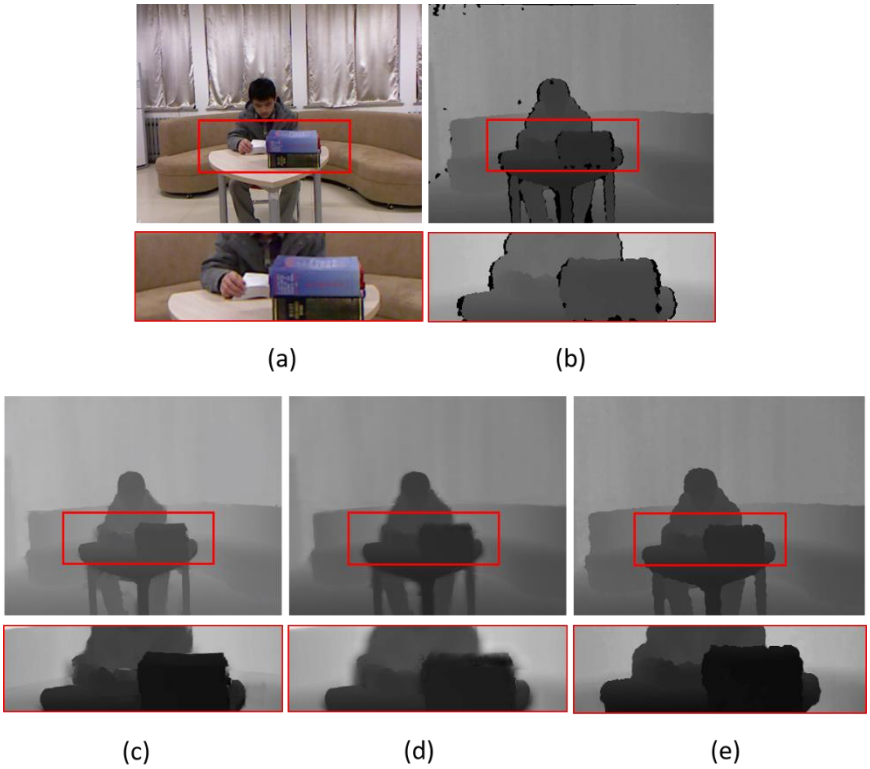


Figure 4-9. A visual comparison of results from the state-of-the-art methods on a set of Kinect images. The thumbnails highlighted in red illustrate zoomed-in portions of the image above them. Column (a) is the color image from Kinect and (b) is the corresponding depth from Kinect. (c) AAR [25]. (d) Guide [21]. (e) Our method.

4.4 CONCLUSION

In this chapter, we propose a new approach to denoise and fill the holes in depth images. The proposed algorithm first extracts an initial depth edge image from the raw depth image which is noisy due to the inaccuracies present in the raw depth image. This initial edge image is refined by using the edges from the color images. The clean depth edge image is used to fill the holes in the

depth image by using an edge and color aware trilateral filter. Many parameters are made adaptive to make the algorithm robust. Quantitative and qualitative experimental results demonstrate that the proposed hole filling strategy can generate more accurate depth images than existing methods.

Chapter 5. APPLYING DENOISING AND SUPER-RESOLUTION IN THE 3D MODELING PROCESS

5.1 INTRODUCTION

The depth images obtained from commercially available economical depth cameras are limited by the noise (including holes where there is no data, and random noise) and the relatively low spatial resolution which cause the partial 3D point cloud from one frame to be relatively sparse. For example, the resolution of SwissRange SR4000 depth camera is only 176 × 144 and the resolution of PMD Camcube camera is only 200 × 200. The resolution of the raw depth image obtained from a Kinect V1 camera is 320 × 240, whereas the color image resolution is 640 × 480 at 30 fps. Similarly, Kinect V2 raw depth resolution is 512 × 424, whereas the color resolution is 1920 × 1080 at 30 fps.

These limitations affect the precision and quality of the final 3D object models created using these depth images. Several approaches have been used to denoise and achieve high-quality 3D object models. For example, using KinectFusion [34], we can slowly scan the more difficult regions of a scene and rely on spatial-temporal averaging for denoising and high-resolution to obtain high-quality 3D object models. However, in many situations such as the ones we are addressing, where only limited numbers of views are available for building the 3D object models, manually slowly scanning the object is not an option.

In the previous chapters, we developed new techniques for constructing high-quality 3D object models using depth cameras with a limited number of views [35]. In the 3D object model construction process described in the previous chapter, depth information in the raw depth images was used directly to form the partial 3D point clouds before the 3D point cloud alignment. The

aligned 3D point clouds go through denoising and meshing to form the final 3D object model. With this process, the noise and low-resolution problems were taken care of by the 3D model denoising and meshing process. Since our ultimate goal is to construct high quality dense 3D point clouds for 3D object models, we are interested in techniques which could improve the overall quality of the dense 3D point clouds and the 3D object models.

In the previous chapter, we also developed new techniques for denoising the depth images utilizing the characteristics of depth images [36]. In this chapter, we investigate the impact of denoising and super-resolving the depth images before the alignment of partial 3D point clouds on the quality of the 3D object models. This question is not trivial, since, although denoising the depth images could make the depth information cleaner, it could also introduce some artifacts. Likewise, super-resolving the depth images could give rise to denser 3D point clouds, but it also could introduce artifacts in the super-resolved depth images and ultimately in the resultant 3D object models. On one hand, the cleaner and denser depth information should improve the overall denoising and meshing results. On the other hand, it is not clear if the final step in denoising the 3D object models could suppress these artifacts and result in better quality 3D object models. In this chapter, we investigate these issues and show that applying denoising and super-resolution on the depth images before aligning partial 3D point clouds in the proposed process described in the previous chapter, can improve the overall quality of the 3D object models.

The role of depth preprocessing in 3D object models is an important topic to study. Especially in our case of using a limited number of RGB-D frames to build 3D models. However, to the best of our knowledge, there is very limited literature studying the impact of denoising and super-resolving the depth images on the 3D object models.

The organization of this chapter is as follows. In Section 5.2, we propose to apply denoising and super-resolution on the depth images before the partial 3D-point cloud alignment. We summarize several possible different approaches of with and/or without denoising and super-resolution. In Section 5.3, we perform simulations to compare the quality of the 3D object models constructed by these different approaches and show that the proposed denoising and super-resolving the depth images before partial 3D point cloud registration can improve the quality of the constructed 3D object models. In Section 5.4, we present some discussions and future work. Finally, in Section 5.5, we conclude this chapter.

5.2 Comparing Different Approaches

We use the 3D object modeling algorithm discussed in Chapter 3 as the baseline algorithm. It consists of the following major steps: registering partial 3D point clouds to obtain the complete 3D object point model, denoising the 3D object model, and meshing. Essentially, the depth noise is taken care of in the 3D object model denoising step, and the low-resolution depth image is taken care of in the final meshing step.

In the next subsections, we investigate the effect of denoising and super-resolving the depth images prior to the 3D object modeling process. Specifically, we compare the baseline algorithm with the following three scenarios:

a) Denoising depth images followed by the baseline algorithm
b) Super-resolving depth images followed by the baseline algorithm
c) Denoising followed by super-resolving the depth images before the baseline algorithm

5.2.1 *Denoising depth images followed by the baseline algorithm*

Depth images produced from RGB-D cameras have holes, noises, and inaccurate depth measurements due to surface properties of objects in the scene, and occlusions caused by the placement of the infrared light projector and the sensor. Depth denoising is the process of correcting noisy depth values and accurately estimating missing depth values by utilizing the raw depth image and/or the RGB image and, hopefully, preserving depth discontinuities around object boundaries.

In this scenario, the depth images are first denoised using the algorithm we presented in Chapter 4. That denoising algorithm utilized the characteristics of depth images to achieve better performance than other state-of-the-art algorithms. Figure 5-1 (a) shows a depth image and (b) the corresponding denoised depth image.

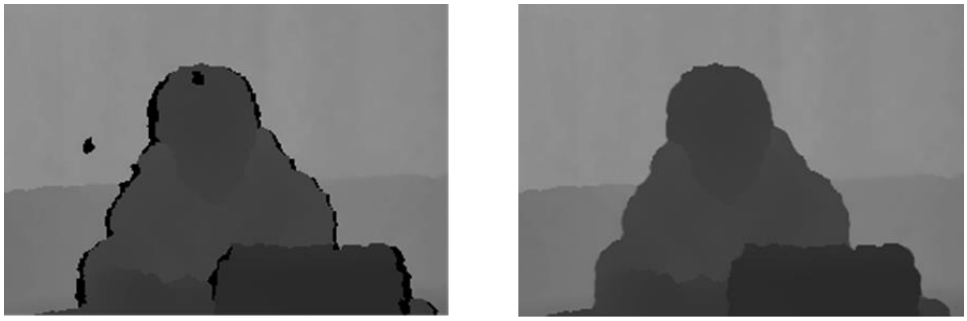


Figure 5-1. (a) An example depth image (b) the corresponding depth image denoised using [36].

Then the partial 3D point clouds are registered to get the complete 3D object model. Finally, the object model is denoised and meshed as in the baseline algorithm. As shown in Figure 5-2, the

only difference between this algorithm and the baseline algorithm is that this method denoises the depth images first before other processing is applied.

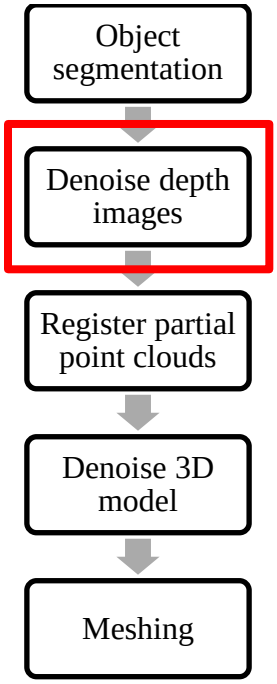


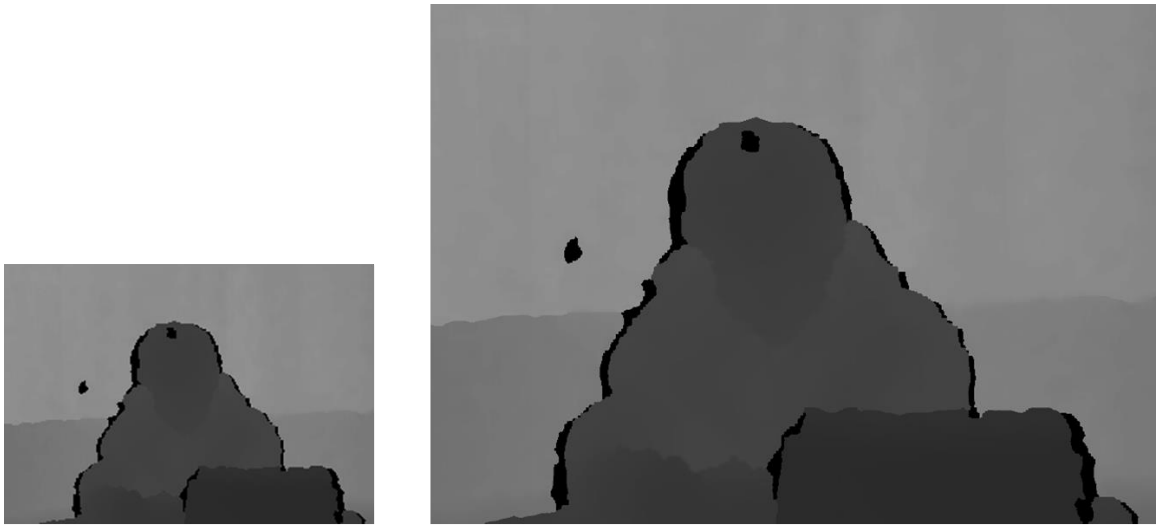
Figure 5-2. The flowchart for denoising the depth images before the partial point cloud registration.

The initial denoising of the depth images is the difference between the baseline method and this method. Essentially, the denoising is carried out in two stages. In the first stage, the noise (including the holes) in the depth images is denoised, which will give cleaner depth images. In the later stage, the noise in the constructed 3D objects is denoised at the 3D object model level. With this two-stage denoising approach, we expect it can achieve better performance than the baseline algorithm.

5.2.2 *Super-resolving depth images followed by the baseline algorithm*

Due to the fledgling depth sensor technology, raw depth images have a lower spatial resolution than their corresponding RGB image resolution. This leads to sparse partial point clouds per RGB-D frame. Super-resolution is the software process of inferring a higher resolution image I_n from one or more low resolution images I with n factor higher resolution in both the horizontal and vertical dimensions than those of the original image I. The image I could be a RGB image or a depth image. Possible issues during super resolution is the creation of artifacts, and the presence of noise in the original image.

In order to achieve a high-resolution 3D object model from the relatively low-resolution depth images, the baseline algorithm relies on meshing performed on the final object model. It is interesting to see if performing super-resolution on the depth images to increase the depth image resolution will help the meshing in achieving better 3D object models. To investigate this issue, the depth images are super-resolved using the algorithm in [37] by a factor of 2. The super-solution algorithm in [37] takes in to account the characteristics of depth images to achieve better performance than other state-of-the-art depth-image super-resolution algorithms. Figure 5-3 illustrates an example depth image (a) and its super-resolved version (b). The color images which have different characteristics from the depth images are super-resolved using the bicubic interpolation by a factor of 2.



(a) (b)

Figure 5-3. (a) Depth image (b) the corresponding depth image super-resolved by a factor of 2 using [37].

Then the partial 3D point clouds are registered to get the complete 3D object model. Finally, the 3D object model is denoised and meshed as in the baseline algorithm. As shown in Figure 5-4, the only difference between this model and the baseline model is that this model super-resolves the depth images before other processing is applied. Essentially, the increase of the resolution of the final 3D object model is achieved in two stages. In the first stage, the depth image resolution is increased to result in denser partial 3D point clouds. In the later stage, the 3D object point cloud is meshed to result in smooth presentations. With this two-stage approach, we also expect it can achieve better performance than the baseline algorithm.

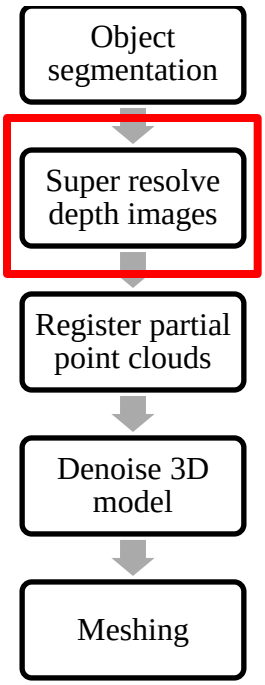


Figure 5-4. The flowchart for super-resolving the depth images. The initial depth super-resolution of the depth images is the difference between the baseline method and this method.

5.2.3 *Denoising followed by super-resolving depth images before the baseline algorithm*

In the previous method, the depth images are super-resolved in order to improve their resolution. However, since the depth images are very noisy and contain holes where there is no data available, directly super-resolving the depth images could introduce many artifacts and worsen the holes and noises. It may be better to denoise the depth images first before performing the super-resolution. To investigate this issue, the depth images are denoised using the method we proposed in Chapter 4, then super-resolved using the algorithm in [37] by a factor of 2 as shown in Figure 5-5. The color images are also super-resolved using bicubic interpolation by a factor of 2.

(a) (b)

Figure 5-5. (a) Depth image (b) the corresponding depth image first denoised using [36] and then super-resolved by a factor of 2 using [37].

Then, the partial 3D point clouds are registered to get the complete 3D object model. Finally, the object model is denoised and meshed as in the baseline algorithm. As shown in Figure 5-6 the only difference between this method and the baseline method is that this method denoises and super-resolved the depth images before other processings are applied.

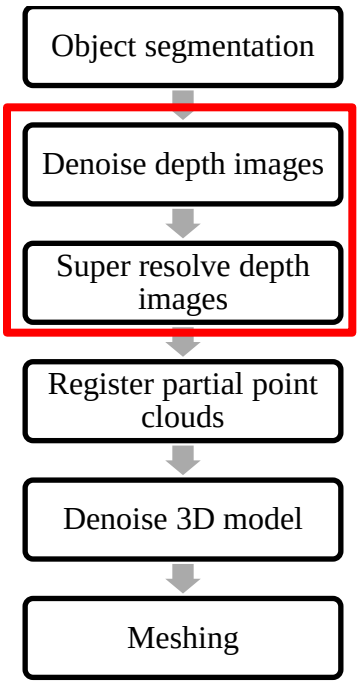


Figure 5-6. The flowchart for this method. The initial depth denoising and super-resolution of the depth images is the difference between the baseline method and this method.

5.3 Experimental Results

We use objects, as shown in Figure 5-7, which were investigated in the previous chapters for comparison. Simulation results are shown in Table 5-1 to Table 5-4. In the tables, "A" is the baseline 3D modeling method, and "B" is the altered 3D modeling method. We use our proposed evaluation method described in Chapter 3 to evaluate the performance of each method. Each 3D modeling process is run 5 times with a different number of input RGB-D image pairs. The average RMSE errors between the resultant 3D models and the ground truth models are reported. Since the visual differences between the models are hard to qualitatively evaluate and are subjective, we report the quantitative results. The quantitative results help us to objectively compare different methods.

Figure 5-7. Objects tested under various denoising and super-resolution conditions.

First, the experiment outlined in Section 5.2.1 was performed. The depth images were denoised and later subject to the 3D modeling process. The comparisons between the resultant models and the ground-truth models are tabulated in Table 5-1.

Table 5-1. Comparing the results from the baseline 3D object modeling process (A) vs the denoising depth images followed by the baseline 3D object modeling process (B).

Number of frames			75	60	45	30	15
Average RMSE (mm)	Biscuit Box	A	0.44	0.59	0.61	0.79	0.98
		B	0.40	0.53	0.56	0.74	0.91
		% improvement	9.1	10.2	8.2	6.3	7.1
	Nut Container	A	0.49	0.59	0.67	0.95	1.23
		B	0.43	0.55	0.64	0.89	1.15
		% improvement	12.2	6.8	4.5	6.3	6.5
	House Model	A	0.45	0.57	0.71	0.89	1.14
		B	0.41	0.53	0.67	0.84	1.10
		% improvement	8.9	7.0	5.6	5.6	3.5
	Mug	A	0.51	0.73	0.85	0.99	1.35
		B	0.46	0.69	0.80	0.95	1.28
		% improvement	9.8	5.5	5.9	4.0	5.2

From Table 5-1, it can be seen that denoising the depth images before further processing could improve the results by about 6.9% on average. Denoising aids in improving the 3D model when

more frames are used. Good quality depth denoising would help better registration of partial point clouds. Thereby leading to better 3D models.

In order to evaluate the effect of different denoising methods on 3D models, we used the denoising methods outlined in [38] and [39]. The Guide [38], JBF [39], and our [36] algorithms were used to denoise depth maps. These depth maps were used in the 3D modeling process described in Chapter 3 to obtain 3D models B_Guide, B_JBF, and B_ours, respectively. These experiments were performed with 45 RGB-D frames to create the 3D models and the results are given in Table 5-2.

Table 5-2. Comparing the effect of different depth denoising methods on the resultant 3D models by using 45 RGB-D frames to create 3D models.

Model	Average RMSE (mm)			
	Biscuit Box	Nut Container	House Model	Mug
A	0.61	0.67	0.71	0.85
B_Guide	0.575	0.653	0.69	0.816
B_JBF	0.571	0.648	0.68	0.807
B_ours	0.56	0.64	0.67	0.80

In Chapter 4, it was shown that denoising methods Guide [38] and JBF [39] blur object boundaries in depth images. In Table 5-2 we see the effect of this depth blurring in 3D. The 3D models created with object boundary blurred depth images are less accurate than models created with depth images where object boundaries are not blurred.

Next, the experiment outlined in Section 5.2.2 was performed. The RGB and depth images were super-resolved by a factor of 2 and used in the 3D modeling process to create object models. The comparisons between the resultant models and the ground-truth models are tabulated in Table 5-3.

Table 5-3. Comparing the results from the baseline 3D object modeling process (A) vs the depth superresolution followed by the baseline 3D object modeling process (B).

Number of frames			75	60	45	30	15
Average RMSE (mm)	Biscuit Box	A	0.44	0.59	0.61	0.79	0.98
		B	0.4	0.52	0.54	0.73	0.89
		% improvement	9.1	11.9	11.5	7.6	9.2
	Nut Container	A	0.49	0.59	0.67	0.95	1.23
		B	0.42	0.54	0.63	0.87	1.14
		% improvement	14.3	8.5	6.0	8.4	7.3
	House Model	A	0.45	0.57	0.71	0.89	1.14
		B	0.41	0.52	0.65	0.82	1.07
		% improvement	8.9	8.8	8.5	7.9	6.1
	Mug	A	0.51	0.73	0.85	0.99	1.35
		B	0.45	0.68	0.78	0.93	1.25
		% improvement	11.8	6.8	8.2	6.1	7.4

From Table 5-3, it can be observed that performing super-resolution on the depth images before further processing can improve the results by about 8.7% on average. Super-resolution helps in the point cloud registration phase, thus leading to improved 3D object models.

Finally, the experiment outlined in Section 5.2.3 was performed. First, the depth images were denoised. Then RGB and depth images were super-resolved by a factor of 2 and subsequently used in the 3D modeling process to create object models. The comparisons between the resultant models and the ground-truth models are tabulated in Table 5-4.

Table 5-4. Comparing the results from the baseline 3D object modeling process (A) vs the depth denoising and super-resolution followed by the baseline 3D object modeling process (B).

Number of frames			75	60	45	30	15
Average RMSE (mm)	Biscuit Box	A	0.44	0.59	0.61	0.79	0.98
		B	0.39	0.52	0.53	0.71	0.87
		% Improvement	11.4	11.9	13.1	10.1	11.2
	Nut Container	A	0.49	0.59	0.67	0.95	1.23
		B	0.42	0.53	0.6	0.85	1.12
		% Improvement	14.3	10.2	10.4	10.5	8.9
	House Model	A	0.45	0.57	0.71	0.89	1.14
		B	0.4	0.5	0.62	0.8	1.06
		% Improvement	11.1	12.3	12.7	10.1	7.0
	Mug	A	0.51	0.73	0.85	0.99	1.35
		B	0.44	0.67	0.76	0.91	1.22
		% Improvement	13.7	8.2	10.6	8.1	9.6

It can be observed that performing both denoising and super-resolution on the depth images before further processing can achieve the best results, achieving an improvement of about 10.77% on average. The results are better than performing either denoising or super-resolving the depth images. However, note that the improvements are not additive. Denoising removes artifacts and

holes in the depth image. When super-resolution is performed after denoising, it increases the resolution of a clean depth image. None of the undesirable artifacts are worsened by the super-resolution operation. For this reason, the 3D models are improved by using both denoising and super-resolution.

5.4 Discussion and Future Work

Denoising the depth images before the baseline 3D object modeling process provides improvements to the model. Applying super-resolution to the baseline increases the improvement. Applying denoising and then super-resolution to the depth images gives the best performance. However, the improvement is only incremental, not the sum of the individual approaches.

This work explored the effect of using our denoising algorithm in [36] to improve the 3D object modeling process. Depth denoising methods proposed in [39] and [38] were also used to denoise depth images and then used in the 3D modeling process. As shown in Chapter 4, these depth denoising methods blur object boundaries. These blurred object boundaries would negatively impact the 3D partial point clouds and the registration of 3D partial point clouds. Results presented here suggest that using denoising methods which do not blur object boundaries, such as [36] are more suitable for 3D model modeling process rather than other methods.

In the super-resolution experiments performed in this chapter, the resolution was increased by a factor of 2. One could also experiment with higher factors of super-resolution and evaluate its impact on the modeling process. Higher factors of super-resolution are bound to produce some artifacts, so these artifacts may affect the 3D models. Super-resolution is a very active field with many proposed methods. Another possible future work is to evaluate these super-resolution techniques for depth and color images on the 3D modeling process.

While software techniques can be fruitful, it is inevitable to consider the possibility of improving the sensor to create noise-free and higher resolution depth images. Better sensor data with improved software techniques would be very powerful to bring 3D modeling based applications to everyday life.

5.5 CONCLUSION

In this chapter, we investigate the effects of applying denoising and super-resolution to the depth images in the process of 3D object modeling. We show that both denoising and super-resolving the depth images before further processing can result in improved 3D object models, and combining denoising and super-resolution on the depth images can achieve the best results. Further work can be performed by finding the optimal super-resolution factors and investigating the impact of other super-resolution methods on 3D object modeling.

Chapter 6. CONCLUSION AND FUTURE WORK

6.1 CONCLUSION

The main contribution of this dissertation is to provide a framework for building a high-quality 3D model of an object with only a limited number of views and to develop new techniques to improve the quality of such models.

In Chapter 2 we provide a brief review of RGB-D cameras available in the market and a detailed overview of Kinect V1 and V2 cameras which were primarily used in this work. We also reviewed published work in 3D object modeling. KinectFusion and CopyMe3D provide state-of-the-art results in 3D modeling. Some limitations of these techniques and the premise of this dissertation were laid out in that chapter.

When the only relatively small number of views (RGB and depth images) are available, the overlapped areas between frames from different views can be rather limited, and thus, the denoising and registration techniques used in KinectFusion or CopyMe3D may not be as effective. Also, these methods may fail when objects lack salient structural features. In Chapter 3, we propose a method to build a high-quality 3D model of an object with a limited number of available views of the object. We developed a complete 3D object model construction process with automatic object segmentation, pairwise registration, global alignment, model denoising, and texturing, and studied the effects of these functions on the constructed 3D object models. We also developed a process for objective performance evaluation of the constructed 3D object models. We collected laser scan data as the ground truth using a Roland Picza LPX-600 Laser Scanner to compare to the 3D models created by the proposed process and other state-of-the-art methods. We show that the

proposed approach can result in much better convergence and overall quality compared to KinectFusion, especially when the number of available views is limited.

One of the limitations of RGB-D cameras is that its depth image contains holes, noises, and inaccurate depth measurements. The accuracy of various RGB-D related applications, such as 3D modeling, suffers from these depth image errors. In Chapter 4, we propose a solution to the depth image denoising problem by estimating depth edges that correspond to the object boundaries and using them as priors in the hole filling process. The proposed algorithm first extracts an initial depth edge image from the raw depth image which is noisy due to the inaccuracies present in the raw depth image. This initial edge image is refined by using the edges from the color images. The clean depth edge image is used to fill the holes in the depth image by using an edge and color aware adaptive trilateral filter. The algorithm improves the accuracy of depth images near object boundaries, preserves object boundaries in depth images, and denoise depth images in homogeneous regions. Compared to previously reported approaches, our proposed approach focuses on recovering clean depth edges corresponding to the true object boundaries. The proposed method exhibits quantitative and qualitative improvements over the current state-of-the-art methods.

The depth images obtained from commercially available economical RGB-D cameras are limited by the noise and the relatively low resolution which cause the partial 3D point cloud from one frame to be relatively sparse and error-prone. These limitations affect the quality of the 3D object models created using these depth images. In Chapter 5, we investigate the impact of denoising and super-resolving the depth images before 3D registration on the quality of the resultant 3D object models. While depth processing could improve the depth images, it could inadvertently introduce artifacts in the 3D models. This is especially of concern when only a limited number of frames are

used to produce 3D object models. In Chapter 5, we show that applying denoising and super-resolution on the depth images before aligning partial 3D point clouds, as in the process described in Chapter 3, can improve the overall quality of the 3D object models. We also show that combining denoising and super-resolution on depth images can achieve the best results.

Finally, we conclude this dissertation in this chapter by providing a summary of the work and some directions for future work.

6.2 FUTURE WORK

In this section, we discuss some possible future work and directions to extend the work provided in this dissertation.

6.2.1 *Using multiple RGB-D cameras*

In this work, we primarily used one Kinect device to capture multiple frames of an object in different views. Alternatively, one could use multiple devices to simultaneously capture multiple views of an object. If too many devices are used, then IR light from adjacent cameras may cause interference. This would lead to even poorer quality depth images and pose harder challenges for 3D object modeling. A primary challenge of using multiple devices is that all the devices should be calibrated. Other challenges are the cost and infrastructure needed and the time to set up the system. But once these challenges are met, the time to capture data to produce the 3D models would be significantly reduced. Deformable objects are objects which may distort while capturing data. Since the data can be recorded quickly when using multiple RGB-D cameras, deformable objects may also be reconstructed using this system.

Initially, the RGB-D data for this work was captured using a Kinect V1 and then later a Kinect V2 device was used. The techniques developed in this work are applicable to any RGB-D platform. To extend this work, the project can be transferred to another RGB-D platform, such as Intel RealSense depth cameras. Moving to a new platform may bring with it new engineering challenges. For example, Intel RealSense R200 [40] and D400 [41] series cameras use stereoscopic IR to produce depth images. These depth images have slightly different noise characteristics than depth images produced by IR structured light (Kinect V1) or IR time-of-flight (Kinect V2) techniques. New depth denoising techniques may be needed to achieve optimal depth denoising performance.

6.2.2 *Improved partial point cloud registration*

One of the critical methods of the 3D modeling algorithm is the partial point cloud registration technique. A possible work is to evaluate and extend the state-of-the-art point cloud registration techniques such as the algorithm presented in [42]. Convolutional neural networks and its variants are producing state of the art results in many areas of computer vision [43]. One could use a Siamese network architecture to feed in two depth images along with ground truth 6 degree-of-freedom (DOF) transformation to design a method to learn transformations between successive frames.

The lack of large-scale RGB-D datasets is an obstacle for pushing the realm of point cloud registration and 3D object modeling using RGB-D data. Large-scale datasets are needed to comprehensively evaluate and compare methods. Also, deep learning methods need a lot of labeled data to build models. Another challenge is to obtain high-quality ground truth for the training set. One possible solution to circumvent both problems is to use synthetic data. With advances in

computer graphics, one could generate a large volume of realistic 3D models [44]. These models could be artificially, yet within physical limits, transformed. Now the ground truth 6DOF is known. The depth images corresponding to the original 3D model and transformed 3D model can be obtained by projecting the models in different views. With this large-scale synthetic dataset, many techniques for improving registration of point clouds could be explored.

Since the end goal is to test and use on real RGB-D data, a smaller dataset of RGB-D frames of 3D objects could be collected. Good quality ground truth data can be obtained by using crowdsourced mechanisms, such as Amazon Mechanical Turk (AMT) to gather, annotate, and correct data. But one must be objective when designing tasks for the AMT workers and must also consider a human error in performing such tasks. Using domain adaption techniques, models learned on synthetic data could be transferred to work with real RGB-D data.

6.2.3 *Data-driven depth denoising*

The proposed depth denoising algorithm may not perform well on depth images of small structures with intricate depth patterns, such as statues. There are many CAD model libraries of complex statues available on the internet. Given this data, one could use a data-driven approach to tackle this problem. It would be interesting to see how a fully convolutional network [45] could handle complex depth denoising.

6.2.4 *Comparing effect of super-resolution techniques on 3D models*

3D data is sparse. In this work, we super-resolve depth images by a factor of 2 using algorithm [37]. The usefulness of super-resolution could be further investigated by super-resolving the depth

images by a higher factor and by varying the parameters in the algorithm. Furthermore, one could investigate different super-resolution methods for increasing the resolution of the RGB and depth images. The impact of these methods could be evaluated using the 3D model comparison strategy presented in Chapter 3.

APPENDIX A

MALARIA DIAGNOSIS AND QUANTITATION USING CONVOLUTIONAL NEURAL NETWORKS [46]

ABSTRACT

The optical microscope remains a widely-used tool for diagnosis and quantitation of malaria. An automated system that can match the performance of well-trained technicians is motivated by a shortage of trained microscopists. We have developed a computer vision system that leverages deep learning to identify malaria parasites in micrographs of standard, field-prepared thick blood films. The prototype application diagnoses *Plasmodium falciparum* with sufficient accuracy to achieve competency level 1 in the World Health Organization external competency assessment and quantitates with sufficient accuracy for use in drug resistance studies. A suite of new computer vision techniques—global white balance, adaptive nonlinear grayscale, and a novel augmentation scheme—underpin the system's state-of-the-art performance. We outline a rich, global training set; describe the algorithm in detail; argue for patient-level performance metrics for the evaluation of automated diagnosis methods, and provide results for *Plasmodium falciparum*.

A.1 INTRODUCTION

Automated detection of malaria in field-prepared blood films is a challenging computer vision task with potential benefit for millions of people. Half of the world's population are at risk of

contracting malaria, with an estimated 212 million cases in 2015 [47]. A majority of the 429,000 deaths from malaria in 2015, mostly of young children, are attributable to P. falciparum. Four other Plasmodium species—*P. vivax*, *P. ovale*, *P. malarie*, and, rarely, *P. knowlesi*—also infect humans [48].

Microscopy continues to be regarded as a standard for malaria diagnosis and quantitation [49], in part because it can be used to detect other infectious diseases [50], has low incremental cost, is widely available, can measure parasite density, and can identify malaria species. Microscopy can detect low-density infections if enough blood is scanned, but this is time-consuming, difficult, and tedious due to the low density and small size of parasites as well as the abundance of similar non-parasite objects, as illustrated in Figure A-1. To be effective, microscopy needs well-trained staff for consistent slide preparation and examination. In areas with poor quality control, microscopy can produce inaccurate results [51] resulting in inappropriate treatment.

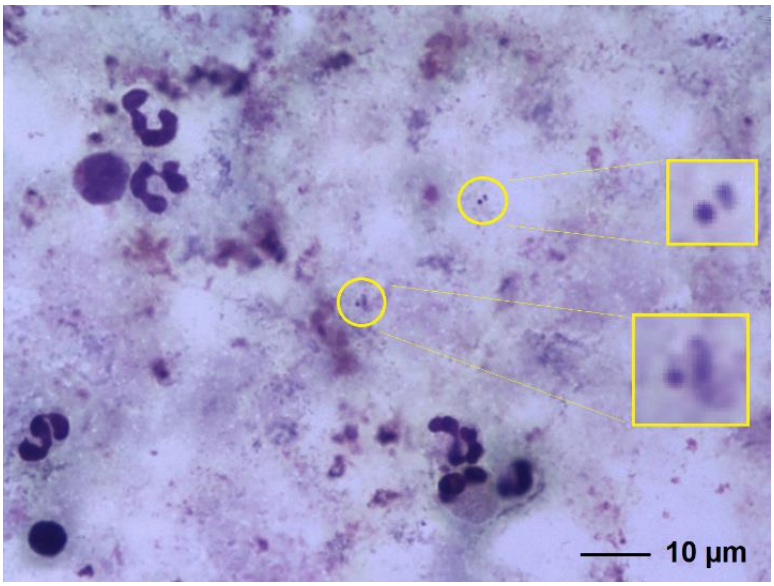


Figure A-1. Typical thick film microscope image. This field-of-view image contains only two parasites indicated by yellow circles with enlargements at the right.

In addition, evolving drug resistance is an increasing concern. The World Health Organization (WHO) encourages regular monitoring of antimalarial efficacy in malaria-endemic countries [52]. Microscopy remains the most field-practical tool to accurately monitor response to therapy, due to its capacity for accurate quantitation, since parasite clearance rates are the most commonly used measure of drug efficacy [53]. But quantitation is a time- and labor-intensive measurement requiring the reading of many blood films [54].

A major difficulty with using microscopy in drug efficacy monitoring is the shortage of trained experts in regions where malaria is endemic [55]. Therefore, the development of a computer vision system to aid in malaria diagnosis and quantitation is an appealing research goal, both because of the difficulty of the task and the high potential benefit. In addition, it is an attractive target for application of convolutional neural networks (CNNs), which have shown success in other image classification tasks [56, 57, 58, 59]. Before addressing automated malaria diagnosis via computer, we present a brief overview of malaria blood film microscopy.

A.1.1 *Blood film microscopy*

Two types of blood films are used to diagnose malaria: thick film and thin film. Here, we will mainly be concerned with thick films because they provide a sufficient volume of blood to enable reliable diagnosis of low parasite density infections [60]. We have also developed a thin film module, which will be presented in a subsequent publication.

The thick film is prepared by placing a drop of blood (about 2 μL) on a slide and using the corner of another slide to spread the drop in a circular pattern to ~1.2 cm diameter. The slide is then dried

and stained with a Romanovsky-type stain, e.g. Giemsa [48], then rinsed and dried again. Giemsa results in DNA (e.g. nuclei) staining purple and RNA (e.g. cytoplasm) staining blue.

We confine our discussion here to *P. falciparum*. The most commonly found parasite stages in *P. falciparum* positive blood films are ring forms (immature trophozoites). In Figure A-2, a number of examples are shown in finer detail. The small, round, purple disk, found in most of the thumbnails, is the nucleus of the parasite; the wispy blue-gray shape in close proximity is the cytoplasm. Later stage trophozoites (Figure A-2, lower-left) do not have a clear, round nucleus and distinct cytoplasm. Note the variety in the appearance of trophozoites. All these must be recognized as *P. falciparum* parasites and must be distinguished from non-parasites.

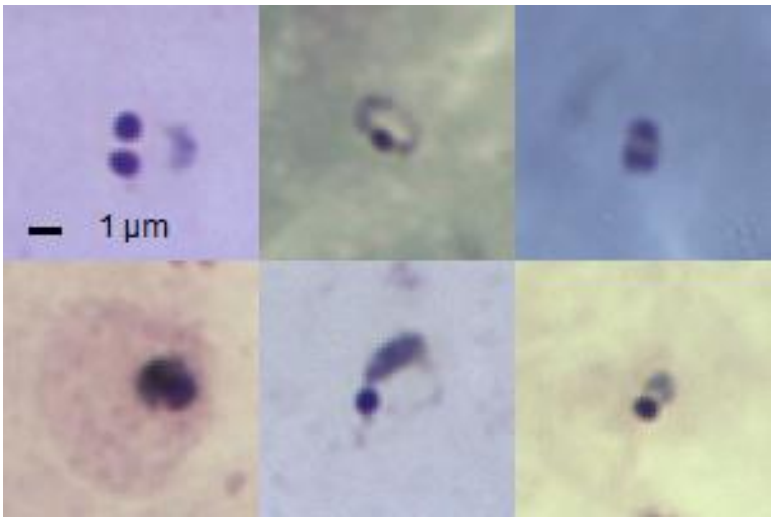


Figure A-2. Ring form *P. falciparum* malaria parasites.

Uninfected normal human red blood cells (RBCs), which are lysed during staining of the thick film, contain neither DNA nor RNA and do not appear dark blue or purple; thus Giemsa provides good contrast between parasites and background. Nevertheless, interpretation of thick films is challenging because of a noisy and variable background. Example fields-of-view (FoVs) are

shown in Figure A-1 and Figure A-3. Note the white blood cells (WBCs) in Figure A-3, whose nuclei are stained similarly to the parasite nuclei in Figure A-2. Various non-parasite components of the thick film can also absorb stain, creating artifacts that may mimic parasites, and the stain itself can self-aggregate. Collectively, artifacts and objects that are difficult to distinguish from parasites are termed distractors, examples of which can be seen in Figure A-1, Figure A-3 and Figure A-4.

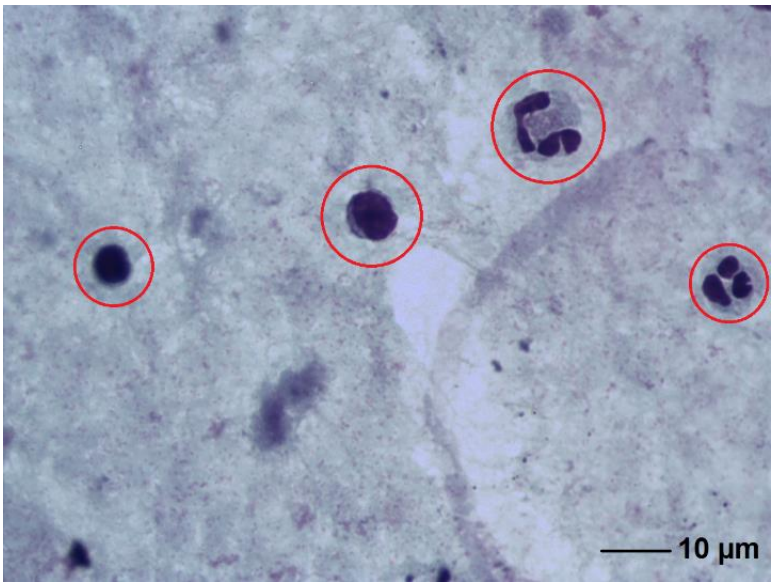


Figure A-3. An FoV image of a negative sample, i.e. with no malaria parasites. WBCs are indicated with red circles.

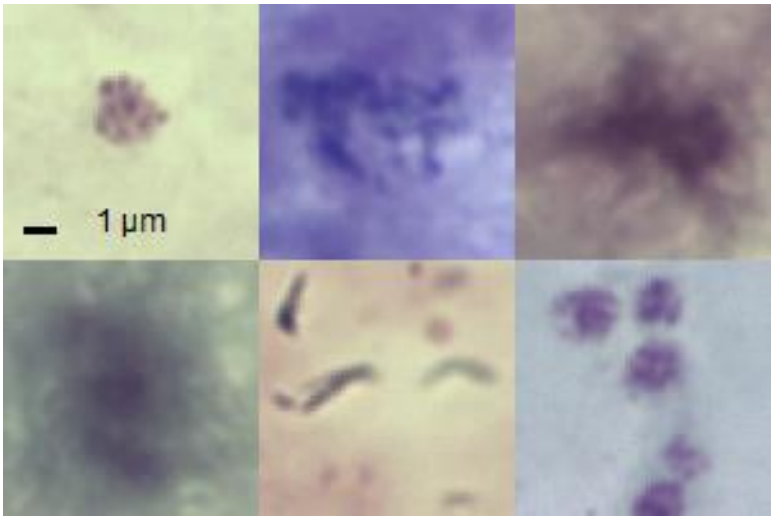


Figure A-4. Examples of distractors. The objects in the upper left and lower right corner are platelets.

In this application, images of blood films are acquired with a digital scanning microscope. The nucleus of the trophozoite can be as small as $1\ \mu m$ in diameter, and other components used to detect malaria parasites and identify species can have features smaller than $250\ nm$, which is close to the optical limit of resolution. To resolve features of this size, a high numerical aperture (NA) oil immersion objective, for example $100 \times\ NA = 1.2$, is required. At these high numerical apertures, the depth of field is on the order of $0.5\ \mu m$. To detect parasites throughout the depth of the thick film, and to ensure all fields of view are in focus, images must be captured at multiple focal planes spanning the entire depth of the blood film. An FoV refers to a stack of images centered at a particular x, y location in the slide, at one or more focal depths z. To achieve reliable detection at low parasitemia levels, WHO recommends inspecting 100 thick film FoVs before declaring a sample negative [48]. (This pertains to a manual $100 \times$ microscope, whose FoVs tend to be larger than automated microscope FoVs.)

A.1.2 *Related Work*

Several proposals for the computer-automated reading of malaria blood films (thin or thick) have appeared in the literature in the past few years [61, 62, 63, 64, 65, 66]. Many of these studies have not presented realistic assessments of the field-effectiveness of their algorithms, due to a lack of emphasis on patient-level metrics. Full reviews of these publications may be found in [67, 68]; here we merely offer a brief overview of a few of these proposals. In Section A.3.2, we present a summary analysis of key metrics for all of these systems and a comparison to ours.

An automated system for the diagnosis of malaria from thick blood smears is proposed in [61]. They crop random, overlapping patches from good images (discarding out-of-focus images) representing blood smears of 133 patients. Patches containing a parasite (based on expert annotation) are marked as positive and the remaining patches as negative. Traditional feature engineering and an ensemble decision-tree classifier form the core of their system. The classifier applied to the test set achieves an area-under-the-curve (AUC) metric of 0.97. The authors evaluate this result purely on the object level, reporting that it achieves 20% recall at a precision of 90%. There is no reference to parasitemia level. (The significance of parasitemia in relation to sensitivity and specificity is discussed in Section A.2.6). Random assignment of images to the train and test sets implies that these sets were not disjoint at the patient level and therefore, the reported metrics are not predictive of actual field performance (see Section A.2.1).

Among the prior automated malaria diagnosis systems considered here, [63] uniquely does not use Romanovsky staining. Rather it uses a cartridge that accepts a sample of blood and automatically creates a film stained with fluorescent Acridine Orange (AO) [69]. Automatic slide creation and AO staining do hold some advantages, but a reluctance to adopt a new, disposable cartridge and

reliance on a fluorescence microscope may prove barriers to field acceptance. The authors do provide patient-level metrics and report a limit of detection of tens of parasites per μL of blood. The quantitation results shown, however, are inadequate for use in drug efficacy studies.

An automated malaria diagnosis system that works on thick film microscope images is described in [66]. They train the system on a dataset consisting of 27 *P. falciparum* positive and 36 negative blood samples. The test set consists of 24 *P. falciparum* positive and 20 negative samples. They do provide patient-level metrics and report achieving WHO competence level 1 diagnosis accuracy, albeit at a parasite density of $300\ p/\mu L$. They allude to the use of CNNs for feature extraction, but the results they report use traditional feature engineering (morphological, shape, color, texture, and Haar-like features).

Our system is intended for use under field conditions. Thus, the system has the following requirements and characteristics: (1) accepts standard field-prepared, Giemsa slides; (2) is robust to moderate variability in slide quality; (3) scans a sufficient volume of blood, approximately $0.1\ \mu L$, ~300 FoVs; (4) scans at multiple focal planes; (5) has high patient-level sensitivity and specificity at low parasitemia—approaching $100\ p/\mu L$; (6) has accurate quantitation in the parasitemia range of $200\ 200{,}000\ p/\mu L$; and (7) has high object-level sensitivity and specificity. Our system has a resolution of 11.36 pixels/μm and each FoV is 1280 × 960 pixels. We scan at 9 focus levels, $0.3\ \mu m$ apart, and thus 300 FoVs amounts to ~2.5 gigapixels. The system is trained on a large and diverse set of images, where the test set is disjoint from the training set at the patient level. We report patient-level diagnosis and quantitation results (as opposed to merely object-level classification), which are the most important metrics for the system's intended use-cases. Our system achieves WHO competence level 1 [70] for *P. falciparum* diagnosis (Section A.3.1.1) and

P. falciparum quantitation accuracy sufficient to be used for drug resistance studies (Section A.3.1.2). We now describe the dataset and methods in detail.

A.2 METHOD

Our data processing pipeline consists of a number of modules, each designed with the above requirements in mind. The preprocessing module (Section A.2.2) implements a new sample-level global white balance method. The candidate object detection module (Section A.2.3) processes multiple focal planes for each FoV (image *z*-stacks) and is based on a novel adaptive nonlinear grayscale intensity image. The feature extraction module (Section A.2.4) incorporates CNNs and introduces a new gamma-transform color augmentation scheme. The classification module (Section A.2.5) is designed to allow the system to adapt to local variations, e.g. in slide preparation. Finally, the disposition module (Section A.2.6) computes patient-level diagnosis and quantification (a multiple-instance learning problem) employing a learning algorithm calibrated on the statistics of the validation set.

A.2.1 *Data*

Large numbers and a great variety of images are needed for training the rich deep learning models in our computer vision system. Diversity in the training and testing data contributes to system robustness under heterogeneous field conditions. And because the patient is the atomic unit for diagnosis, samples from a wide variety of patients and labs are essential to validate diagnostic effectiveness. Some relevant statistics of our malaria blood film library are shown in Table A-1.

The models in our system are trained on image patches of individual objects (parasites and distractors) from a subset of patients in the library; the system is tested against objects from a disjoint subset of patients. (See Section A.3 for details on numbers of patients in each subset.) Disjoint training and testing sets at the patient level enable realistic estimates of field performance.

Table A-1. Summary of thick film malaria database.

Blood samples	1,452
Fields-of-view	5,707,947
Parasite objects	956,531
Countries of origin	12

A.2.2 *Pre-processing*

Histologically stained microscope slides typically display color variation within a slide, between slides of different blood specimens, and between different technicians, laboratories, clinics, and regions. Color variation can result from differences in stain pH, age and purity of stain, duration of the staining procedure, and sensor settings; overall slide hue can range from blue to green to pink to golden. Figure A-1 through Figure A-4 illustrate a small fraction of the variability in quality, color, and presentation that is typical of field samples. Uncorrected, color variation may degrade system performance.

White balancing techniques may be used to compensate for some, but not all, of the color variation. Traditional white balancing involves the scaling of red, green, and blue (RGB) pixel values based on the mean color of the brightest pixels in each image individually, which can result in color

distortion and exaggerated intra-slide color differences. Our white balancing technique pools the pixels from all FoVs and computes a global color balance affine transform for each blood sample.

A.2.3 *Detection*

The detection module generates object proposals—potential parasites to be subsequently scored as parasite or distractor by a classifier. To achieve the target limit of detection, some ~300 FoVs need to be processed by the algorithm, due to the Poisson statistics of rare object distributions. To keep the runtime within reasonable limits (roughly 20 minutes on a standard quad-core laptop), the computational complexity of the detection algorithm should be as low as possible.

Most generic object detection methods, such as R-CNN [71], YoLo [72], deformable parts model [73], and selective search [74] are either too complex, too insensitive, or too slow for malaria detection on large numbers of FoVs at multiple focal planes. The deformable parts model performs an exhaustive search using a support vector machine (SVM) on a histogram of oriented gradients (HOG) feature pyramid. Selective search uses segmentation on multiple color spaces based on a greedy hierarchical grouping of graphs. This leads ~10K detections per image, which would drastically slow down our framework. The processing flow in R-CNN (and its variants Fast R-CNN [75] and Faster R-CNN [76]) consists of region proposals, followed by classification, followed by post-processing to refine the bounding boxes and eliminate duplicate detections. These complex pipelines are slow. While YoLo processes 45 frames per second, it fails to detect small objects and objects appearing in clusters, which negatively impacts quantitation performance. These shortcomings render these methods unsuitable for malaria parasite detection.

Leveraging domain-specific information allows the design of a specialized detection scheme that out-performs more general methods. As mentioned previously, Giemsa-stained microscope images provide good contrast between deep purple nuclei and background. Thus, malaria parasite nuclei, along with white blood cells, are among the darkest objects in the images; a dark threshold applied to a grayscale intensity image may act as a simple and effective initial detector for malaria parasites.

While this simple detector has high sensitivity, its precision is low: many dark distractors are also detected, which degrades low parasitemia performance because of excessive false positive detections. To enhance the object-level specificity of the detector, we introduce two innovations: adaptive grayscale intensity and dynamic local thresholding. The standard grayscale intensity is a linear combination of red, green, and blue pixel values that approximate the human-perceived luminance [77], but does not necessarily provide the best separation between parasites and background. Machine learning techniques may be used to compute a more optimal projection vector.

We make use of the above-noted similarity in color between parasite nuclei and WBC nuclei. The latter are relatively easy to detect and classify at high precision because they are large and contrast strongly with the background. In a first pass through the FoV images, we segment WBC candidates using a dark threshold tied to grayscale intensity statistics. Morphological and clustering operations further filter individual WBC candidates, which are then classified with a Gaussian-kernel SVM [78]. The segmentation of WBCs enables the collection of RGB color statistics for WBC pixels and a random sampling of background pixels. Machine learning techniques are then used to compute the optimal projection in RGB space that will separate WBC pixels from background pixels.

The resulting projection, which varies by blood sample, is called the adaptive grayscale intensity. It provides higher precision for parasite detection compared to the standard grayscale intensity. Performance may be further enhanced by adding non-linear terms to the predictor, similar in spirit to polynomial regression. For example, the predictor may be augmented from the linear $\xi = [R, G, B]^T$, to the 2nd order polynomial predictor:

$$\xi = [R, G, B, R^2, G^2, B^2, RG, RB, BG]^T \tag{A.1}$$

More flexible non-linear terms, such as rational functions of the RGB components, may be included, as in the following 12-dimensional non-linear form:

$$\xi = \begin{bmatrix} R, G, B, R^2, B^2, RG, \dots \\ \frac{R}{G+\epsilon}, \frac{R}{B+\epsilon}, \frac{G}{B+\epsilon}, \frac{R+B}{G+\epsilon}, \dots \\ \frac{B-G}{R+G+B+\epsilon}, \frac{G}{R+G+B+\epsilon} \end{bmatrix}^T \tag{A.2}$$

where ϵ is a small constant added to the denominators to prevent overflow. Because of collinearity between the individual components of the predictor, we use regularized regression, such as ridge regression [79], lasso [80], or partial least-squares regression (PLSR) [81]. PLSR with 1 PLS component has performed the best in our experiments. Figure A-5 shows a comparison of the detection free-response ROC curves (FROC) [82] using the standard grayscale image vs. the adaptive grayscale image based on the 12-component non-linear predictor of Equation (A.2). At

98% sensitivity, the adaptive nonlinear grayscale image detects 35% fewer false positives than standard grayscale.

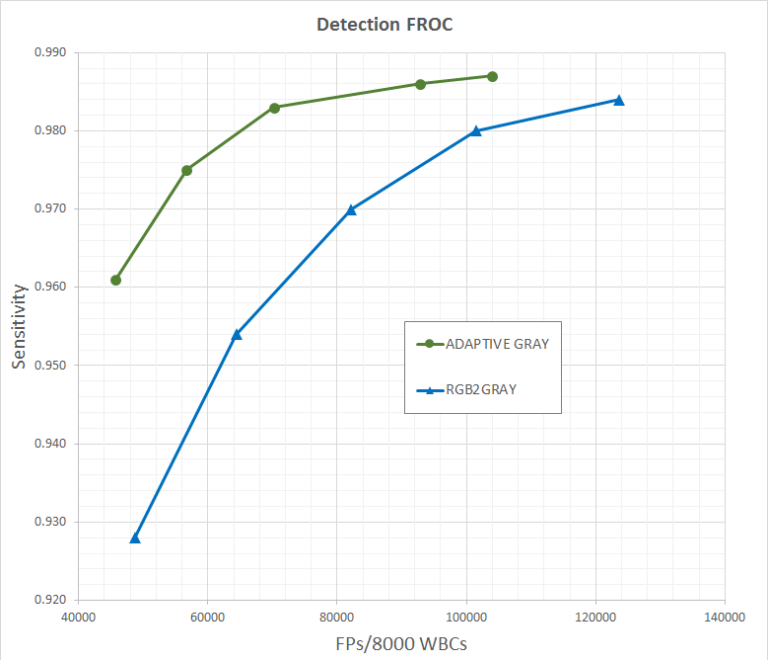


Figure A-5. FROC curves for *P. falciparum* detection based on standard grayscale vs. adaptive grayscale images.

We now address the question of thresholding. Both of the popular thresholding methods [83, 84] assume bimodal intensity histograms. This does not hold when the target objects occupy a negligible fraction of the pixels, as is the case with malaria parasites. Furthermore, a fixed threshold across all FoVs entails a compromise between sensitivity and false positive rate. Adaptive thresholding per FoV is better but still involves compromise because, typically, there are both noisy regions and quiet regions in a single FoV.

Dynamic local (i.e. pixel-wise) thresholding provides the best performance compared to either static or FoV-wise adaptive thresholding. Our thresholding scheme estimates the local noise floor using a large-kernel median filter. WBC pixels (which were detected and classified in the first

pass) are replaced with the median FoV pixel value to prevent WBCs from desensitizing the local threshold.

The adaptive grayscale intensity image is thresholded—pixel-wise—resulting in a binary image that is processed to detect connected-component blobs. These are treated as candidate objects. Since multiple blobs may be detected in a given *z*-level for the same object, distance-based clustering is used to associate nearby detected blobs with a single candidate object. In addition, the same object will frequently be detected at multiple *z*-levels of an FoV. The best-focused version of the object (i.e. *z*-level image patch with the highest Brenner focus score [85]) is selected. The output of the detector is a list of bounding boxes and thumbnails representing the candidate objects.

Notwithstanding the use of adaptive nonlinear grayscale intensity and dynamic local thresholding, many dark distractors are still detected. For low parasitemia samples, these distractors can overwhelm the number of parasites. To eliminate a large number of obvious distractors, we train a Gaussian-kernel SVM based on low-computational cost geometric, color, gradient, contrast, and texture attributes extracted from thumbnails of the candidate objects, using the database of annotated parasites as ground truth. The classifier achieves AUC of about 0.90 for both the training and validation sets. The distractor filter threshold is tuned to keep the object-level sensitivity at 95% for training and 90% for validation (and holdout). We have subsequently employed a random forest classifier [86] for the distractor filter with equivalent or better results.

A.2.4 *Feature extraction*

We extract features on those candidate objects that survive the distractor filter. The recent widespread adoption of CNNs for feature extraction and classification has led to notable breakthroughs in performance for various computer vision tasks [56, 57, 58, 59]. We employ CNNs using the Caffe Deep Learning Framework [87]. We have experimented with a few CNN architectures, including AlexNet [57], VGG [88], and GoogLeNet [89]. These networks were designed for 1000-category vision problems such as ILSVRC [90], and generally, lead to overfitting on our binary classification problem. We, therefore, use reduced versions, optimizing generalization performance by adjusting numbers of filters and layers in VGG and numbers of filters and inception modules in GoogLeNet.

For the results shown in Section A.3, we used a 9 layer VGG architecture (6 convolutional + 2 fully-connected + 1 output). When VGG is used as a feature extractor, the output of 2nd fully connected layer (after dropout) is used as the feature vector. This reduced VGG achieves about 93% accuracy on a validation set for *P. falciparum* vs. distractor. The VGG results are markedly better than those achieved with AlexNet. Subsequent experiments with a reduced GoogLeNet architecture performed roughly equivalent to the reduced VGG.

Training the CNN entails augmentation of data to avoid overfitting. Three different kinds of augmentation are employed. The individual object thumbnails are flipped and rotated in 90° increments which give 8× augmentation. (Smaller angles are avoided to prevent loss of resolution.) Random positional shifts of ± 5 pixels and random augmentation of individual RGB color channels are also performed. Initially, we employed the color augmentation approach described in [57] but found the resulting colors unrealistic. We opted instead for random gamma correction of individual

color channels, which gave more realistic blood smear microscopy image colors as well as improved performance. The number of augmentations used depends on the type of object and is anywhere from 16-64×.

The CNN is trained using equal numbers of (augmented) parasites and distractors and the following Caffe settings: batchsize=128, base_lr=0.001, lr_policy="inv", power=1, gamma=10-4, momentum=0.9, and weight_decay=10-5.

A.2.5 *Classification*

One approach is to use the CNN as both feature extractor and classifier. Another option is to use the CNN as feature extractor and a different algorithm as an external classifier. The first choice has some advantages, including simplicity, speed, and the fact that the CNN is trained with a large (augmented) number of thumbnails. The second option provides more flexibility in responding to new distractor types or sample preparations discovered in the field. Transfer learning [91, 92, 93] assures us that a universal CNN feature extractor, trained on a broad set of samples available in-house, can provide discriminative features in most field settings, while an external classifier can be fine-tuned to local conditions. Initially, the CNN and external classifier are trained on the same in-house samples.

We use logistic regression [94] as the external classifier for two reasons. First, logistic regression mimics the CNN's fully-connected + SoftMax output. Second, the software package [95] implements a robust, large-scale learning algorithm for logistic regression based on stochastic gradient descent. We used this architecture for the results of Section A.3.

A.2.6 *Patient-level disposition*

In object classification tasks (e.g. ILSVRC [90]), a sample is a single image and the endpoint is object classification accuracy. With malaria diagnosis, a sample is a blood film. For each blood film, the system must process hundreds of FoVs and about 10 focal planes per FoV; and it must detect and classify thousands of object thumbnails. The ultimate goal is to diagnose the patient; metrics of success must reflect this goal. Because object identification is only an intermediate goal, good object-level performance is necessary but not sufficient to assure strong performance on patients. Object-level results are relevant only insofar as they affect patient-level accuracy. Thus we develop and emphasize patient-level methods and metrics.

Our system counts the number of detected objects (which include true positives (TP) and false positives (FP)), then diagnoses the patient according to whether this count exceeds some threshold. For patient-level diagnosis, the figure-of-merit (FoM) is the estimated limit-of-detection (LoD) in parasites/μL at fixed specificity (e.g. 95%). This determines whether the system can correctly diagnose low-parasitemia (and healthy) patients.

Consider the following patient-level quantities:

$$\begin{array}{ll} p & actual\ number\ of\ parasites\ per\ \mu L \\ q & suspected\ number\ parasites\ per\ \mu L \\ t & number\ of\ true\ positives\ per\ \mu L \\ f & number\ of\ false\ positives\ per\ \mu L \\ s & object\text{-}level\ sensitivity \end{array} \tag{A.3}$$

The following relations hold:

$$t = p \cdot s \tag{A.4}$$

$$q = t + f \tag{A.5}$$

Substituting Equation (A.4) into Equation (A.5), and solving for p, we obtain:

$$p = (q - f)/s \tag{A.6}$$

Thus, we can estimate parasitemia, p, if we can estimate f and s. We know the ground truth for the validation set, so we can estimate s on the positive samples in the validation set as follows: $\hat{s} = \text{mean}(s)$. We can estimate f on the negative validation set because every suspected parasite is a false positive object. The estimate of f is the threshold $\hat{f}$ on the number of suspected parasites/μL. Let us assume f is Gaussian-distributed at the patient level. If we set the threshold $\hat{f} = \text{mean}(f)$, half of negative patients will be diagnosed as positive. To get 95% patient-level specificity, we must use a larger threshold:

$$\hat{f} = \text{mean}(f) + 1.65 \cdot std(f) \tag{A.7}$$

The mean and standard deviation by patient are taken over the negative validation set, and sensitivity variation is ignored for ease of calculation. Using this threshold $\hat{f}$, we will obtain 95% sensitivity for positive patients when the parasitemia is greater than the following LoD:

$$LoD = 3.3 \cdot std(f)/\hat{s} \tag{A.8}$$

The numerator in Equation (A.8) captures the algorithm's variance in FP rate by patient, while the denominator accounts for the inefficiency of parasite detection. For example, if sensitivity is 50%, and $\mathrm{std}(f) = 80\ p/\mu L$, then a clean slide with many fewer false positives than usual must contain $> 264\ p/\mu L$ to get a positive-object count that exceeds the threshold. This implies that the critical FoMs for estimating LoD are $\mathrm{mean}(s)$ and $\mathrm{std}(f)$ by patient.

A.3 RESULTS

In this section, we report results in two ways, patient-level, and object-level. First, we give patient-level diagnosis accuracy on various low parasitemia holdout sets (Figure A-6), and quantitation results on holdout sets with a range of parasitemias (Figure A-7). Second, we present the object-level metrics that support the patient-level results. We also provide a table that compares our algorithm with various others in the literature. Key metrics include the number of patients and estimated LoD.

A.3.1 *Patient-level results*

Our algorithm was trained on a set of 78 positive and 31 negative patients. Hyperparameters for diagnosis and quantitation (such as $\mathrm{mean}(f)$, $\mathrm{std}(f)$, and $\mathrm{mean}(s)$) were calculated on a validation set of 54 positive and 32 negative patients. Each sample consisted of 324 FoVs ($\sim 0.1\ \mu L$ of blood). Target patient-level specificity was set to 95% on the negative validation set.

A.3.1.1 *Diagnosis*

The algorithm was applied to four holdout sets. Each set contained 20 negative and 10-12 low-parasitemia *P. falciparum* positive slides. Two of the sets were *P. falciparum* diagnosis portions of official WHO55 evaluation sets. The other two sets were ersatz WHO-type sets from different malaria-endemic regions. The low parasitemia samples are important to clinical use-cases. Figure A-6 shows diagnosis results by patient and parasitemia. Specificity on the negative slides in each of the holdout sets was ≥ 90%. We present diagnoses by parasitemia to infer empirical LoD because patient-level sensitivity is a meaningful metric only in association with specificity and parasitemia. These results indicate an effective LoD $\sim 100\ p/\mu L$ at 90% specificity.

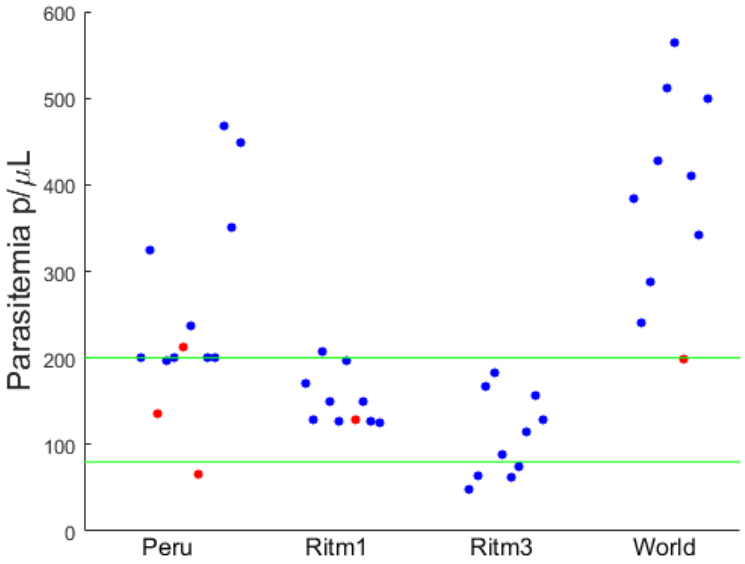


Figure A-6. Diagnosis by parasitemia for 4 holdout sets. Each dot represents a positive patient. Blue dot: correct diagnosis, red dot: false negative. Green lines indicates parasitemia range used for WHO evaluation (90% sensitivity @ 90% specificity = competency level 1).

A.3.1.2 *Quantitation*

Accurate parasite quantitation is important for case management—parasite density can indicate the severity of the infection [48]—and for generating accurate parasite clearance curves [53] in antimalarial efficacy studies [54]. To assess quantitation accuracy, the algorithm was applied to a holdout set of 45 positive *P. falciparum* patients from various regions of the world. Results are shown in Figure A-7. The ± 25% error lines represent a range that allows the calculation of the log slope of clearance curves with error ≲10% for antimalarial efficacy studies. The results indicate that quantitation is sufficiently accurate for parasitemia $> 1000\ p/\mu L$, but that estimates are high for parasitemia $< 1000\ p/\mu L$.

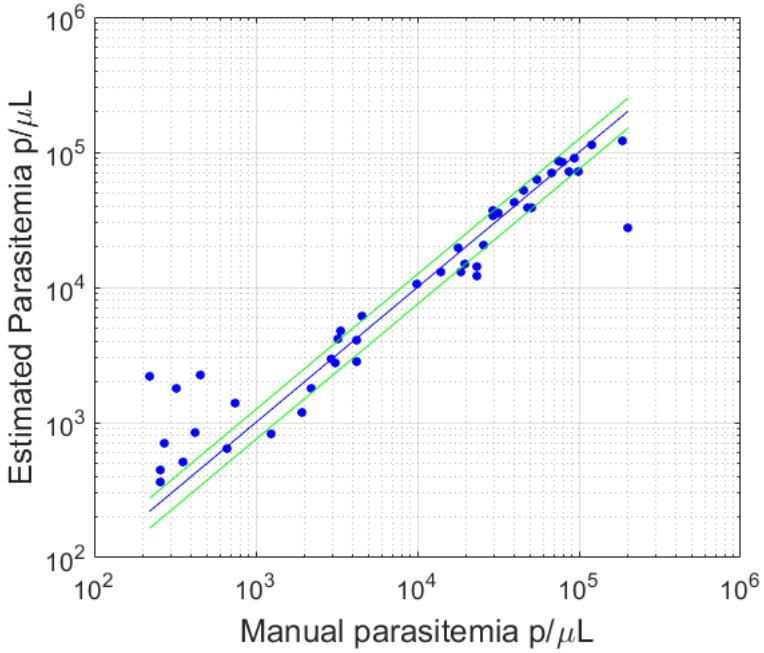


Figure A-7. Parasitemia quantitation performance on holdout sets. Green lines indicate ± 25% range.

A.3.2 *Object-level results*

We now describe object-level metrics supporting the patient-level results described in Section A.3.1.1. Table A-2 shows the confusion matrix for the classification of candidate objects from the validation set via the CNN classifier. The confusion matrix was generated by setting the classifier threshold to 0.6. At this setting, the following object-level performance metrics obtain sensitivity 91.6%, specificity 94.1%, precision 89.7%.

Table A-2. Confusion matrix for CNN on validation set.

	Positive	**Negative**
Parasite objects	33,438	3,064
Distractor objects	3,849	61,405

We compare our results with others in the literature in Table A-3. Although the most relevant comparison between algorithms is at the patient level, due to the general lack of patient-level information in the publications, we primarily compare object-level results. We also predict patient-level results based on object-level metrics: the final column in Table A-3 is a projected LoD at the patient level (estimated via Equation (A.8) at 95% specificity; when $\mathrm{std}(f)$ is not available, then $\mathrm{mean}(f)$ is a useful indicator of the variance in FP rate). Note that several of the methods were proposed as decision-support rather than standalone systems, and for these methods lower specificity may be tolerable.

Table A-3. Metrics of manual and automated malaria diagnosis algorithms.

Algorithm	**Film type**	**# Patients training set**	**µL blood/ patient**	**mean(f) FP/µL**	**std(f) FP/µL estimated**	**mean(s) % estimated**	**LoD p/µL estimated**
WHO level 1 microscopist	thick		0.03-0.07				100
Quinn *et al*. [15]	thick	133 †	0.06	1200 ‡	240	20 ‡	4800
Rosado *et al*. [16]	thick	6	0.04	6470	1294	78	6640
Vink *et al*. [17]	thin AO §	> 22	0.47	< 7	< 2	75	30 ∂
Linder *et al*. [18]	thin	44	0.05	5000	1000	85	9400
Díaz *et al*. [19]	thin	5	0.005	15000	3000	94	12800
Delahunt *et al*. [20]	thick	93	0.1	93	70	20	267
Ours	thick	195	0.1	12	12 *	43 *	112

† Train and validation set not separated by patient.
‡ Assumes 90% precision, 20% recall per authors' suggestion.
§ Uses non-standard Acridine-Orange staining cartridge and fluorescence microscope.
∂ Results from a field trial with 70 positive, 16 negative patients, 84% patient specificity.
* Actual (not estimated).

A.4 CONCLUSION

This appendix describes a CNN-based malaria detection algorithm, the first (to our knowledge) that applies CNN models with sufficient training and validation data and patient-level accuracy to meet two key use-cases of the automated malaria problem: clinical diagnosis down to 100 p/µL; and *P. falciparum* quantitation for drug-resistance studies. The system reads thick film blood slides prepared with Giemsa stain according to current field norms, which is a minimum requirement for the above use-cases. Multiple field evaluations to further test the system are currently underway.

Our internal tests indicate that the system has thus far achieved malaria diagnosis accuracy sufficient to attain competence level 1 in the WHO external competency assessment of malaria microscopists for *P. falciparum*, which means that it performs on a par with well-trained microscopists for this species. It is still the case that highly-trained microscopists can out-perform automated systems. While the algorithm shows robustness to wide variation in field-prepared samples, it can fail when confronted with novel slide preparations or artifacts to which it was not

exposed. This tendency can be ameliorated as the newly encountered material is classified and added to its library via updates.

Our system can also be used for computer-assisted malaria diagnosis since it outputs an array of thumbnails of the most suspicious (i.e. highest scoring) objects. In this mode, the machine reduces the workload of the user by pre-scanning the slide and presenting the most relevant objects for review. In initial field usage, this mode of operation may allow time for stakeholders to gain confidence in the system's capabilities and robustness. Regardless of usage mode, the thumbnails are always available for confirmation and review in case of unusual findings.

The new computer vision methods we have introduced are relevant to applications in automated medical diagnosis via microscopy, sonography, and radiology, as well as problems dealing with rare-object detection. These applications are important areas of computer vision research.

A.5 ACKNOWLEDGEMENTS

The authors gratefully acknowledge the Bill and Melinda Gates Foundation Trust for their sponsorship through Intellectual Ventures' Global Good Fund.

.

VITA

Mayoore Selvarasa Jaiswal

mayoore@uw.edu
https://mayoore.github.io/

EDUCATION

University of Washington, Seattle, WA
Ph.D. in Electrical Engineering Expected Feb 2018
Dissertation: Constructing high-quality 3D object models using RGB-D cameras

Université de Montréal, Montreal, Quebec, Canada
Deep Learning and Reinforcement Learning Summer School June - July 2017

University of Washington, Seattle, WA
M.S in Electrical Engineering Dec. 2012
Coursework: *Electromagnetic Theory and Applications, Electromagnetic Computations and Applications I & II, Microwave Engineering, Fundamentals of Wireless Communication, Computer-Communication Networks*

Anna University, Chennai, India
B.E in Electronics & Communication Engineering, "First Class with Distinction" Apr. 2009

PUBLICATIONS

- H. Hosseini, B. Xiao, **M. Jaiswal**, R. Poovendran, "Assessing Capability of Convolutional Neural Networks in Generalizing Shapes", Under Review
- B. Wilson, M. Horning, C. Mehanian, C. Delahunt, L. Hu, **M. Jaiswal**, S. McGuire, D. Bell, "The Challenges and Prospects of Microscopy and Biophotonics in Low Resource Settings", OSA Biophotonics Congress: Biomedical Optics, April 2018
- **M. Jaiswal**, M. Horning, L. Hu, Y. Ben-Or, C. Champlin, D. Levitz, "Characterization of cervigram image sharpness using multiple self-referenced measurements and random forest classifiers", Optics and Biophotonics in Low Resource Settings, SPIE Photonics West, January 2018
- H. Hosseini, B. Xiao, **M. Jaiswal**, R. Poovendran, "On the Limitation of Convolutional Neural Networks in Recognizing Negative Images", 16th IEEE International Conference On Machine Learning And Applications (ICMLA), December 2017
- C. Mehanian, **M. Jaiswal**, C. Delahunt et. al., "Computer-Automated Malaria Diagnosis and Quantitation Using Convolutional Neural Networks", Bioimage Computing International Conference on Computer Vision (ICCV), October 2017
- **M. Jaiswal**, C. Mehanian, C. Delahunt, L. Hu, C. Thompson, M. Horning and C. Champlin, "Automated Diagnosis and Quantitation of *Plasmodium falciparum* in Thin Film Blood Slide Microscopy Images using Artificial Intelligence", Global Grand Challenges Summit, July 2017

- **M. Jaiswal**, Y.Y. Wang, and M.T. Sun, "Object Boundary Based Denoising for Depth Images", International Conference on Image Analysis and Recognition (ICIAR), July 2017
- M. Horning, Y. Ben-Or, **M. Jaiswal**, C. Champlin, N. Gachuhi, L. Hu, Y. Rosenberg and D. Levitz, "A Digital Framing Ring for Stabilizing Cervix Location in Digital Cervicography Images", IFCPC2017 World Congress for Cervical Pathology and Colposcopy, April 2017
- **M. Jaiswal**, J. Xie, and M.T. Sun, "3D Object Modeling with a Kinect Camera," Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC), December 2014

POSTER PRESENTATION

- **M. Jaiswal**, M.T. Sun, "Constructing High-Quality 3D Object Models using RGB-D Cameras", iREDEFINE ECE Workshop, March 2017
- **M. Jaiswal**, C. Delahunt, C. Mehanian, C. Thompson, L. Hu, M. Horning, S. McGuire and C. Champlin, "Hunting Malaria in the Wild with Machine Learning and Computer Vision", University of Washington Data Science Poster Session, February 2017

INVITED TALKS

- "Importance of Internships and Strategy to Get One!", Grace Hopper Celebration of Women in Computing (GHC), Orlando, FL, October 2017
- "Get out of your own way!", Panelist, GHC, Orlando, FL, October 2017
- "Internships Can Break the Ceiling and We'll Show You How to Get Them", Women in Science and Engineering (WiSE) Conference, University of Washington, Seattle, WA, February 2017
- "Where are the people like me? Examining the Factors that Impact the Cracking of the Ceiling", Women in Science and Engineering Conference, University of Washington, Seattle, WA, February 2017
- "Challenges in 3D Object Modeling", AI With The Best, September 2016

RESEARCH EXPERIENCE

***Research Assistant*, Information Processing Lab** Jan. 2013 – Present
University of Washington, Seattle, WA

- Research innovative methods to improve the quality of the 3D models such as de-noising and adaptive voxels
- Developed a 3D modeling algorithm using limited number of frames of color and depth data from a Kinect to model household objects
- Proposed a methodology to evaluate the models
- Developed an algorithm to de-noise noisy depth images collected from a Kinect camera
- Mentored 1 undergraduate student and 3 master's students on independent projects

***Algorithms Intern*, Intellectual Ventures Lab,** Bellevue, WA June 2015 – Present

- Designed an algorithm to classify cancer in cervicography images captured from a mobile phone camera
- Developed an automated algorithm to evaluate focus of digital cervicography images
- Developed algorithms to quantify Malaria parasites in blood samples using multiple stages of machine learning, convolutional neural networks (CNN) and computer vision techniques

- Designed and implemented graphical user interfaces to assist biologists to annotate blood image data
- Deployed algorithm in multiple field trials in Malaria endemic areas
- Maintained a large-scale database of 1 million blood sample image data

TEACHING EXPERIENCE

Teaching Assistant, **University of Washington,** Seattle, WA Dec. 2010 – June 2012 & Dec. 2012 – Dec. 2015

- Served as Teaching Assistant for undergraduate and graduate courses at the Electrical Engineering department for 13 quarters
- **Courses:** *Introduction to Electrical Engineering, Continuous Time Linear Systems, Introduction to Digital Circuits and Systems, Discrete-Time Linear Systems, Introduction to Digital Imaging Systems, Microwave Engineering, and Digital Image Processing*
- Instructed lab sections, graded exams, labs, and homework, and provided feedback on homework and lab
- Took initiative to hold additional office hours, solicit mid-term evaluations, and lead test review sessions

Study Section Instructor, **Math Science Upward Bound Outreach Program** July 2013 – Aug. 2013

University of Washington, Seattle, WA

- Coordinated with the director and seminar speakers to develop and deliver study section activities to support and extend the lecture topics, engage students, and encourage active learning
- Helped students from under-represented minority groups in STEM, and educationally and economically disadvantaged backgrounds, to understand the content of the seminars and develop good study skills, led them in hands-on activities and work on group projects, and accompanied students on lab and facilities tours

Math/Science Tutor, **Edmonds Community College,** Lynnwood, WA July 2009 – Aug. 2010
Math Tutor, **Shoreline Community College,** Shoreline, Jan 2010 – Aug. 2010

- Assessed students' gaps in comprehension and adapted instructions to the learning style of the student
- Taught material that included GED, beginning through advanced algebra, calculus, and multivariable calculus

INTERNSHIP EXPERIENCE

System Performance Co-op, **Verizon Wireless,** Bellevue WA June 2012 – Nov. 2012

- Collected and analyzed cluster drive data
- Used Verizon Wireless proprietary and non-proprietary tools to monitor and troubleshoot the CDMA, EV-DO and LTE network performance

Graduate Intern, **Network Technology, T-Mobile USA,** Bellevue WA June 2011 – Sept. 2011

- Performed parametric (2G & 3G), protocol and data throughput conformance testing on mobile devices

- Analyzed test cases in Quality Center and assisted in test case clean-up process

LEADERSHIP EXPERIENCE

- ***Corporate Relations Officer,*** **Institute of Electrical and Electronics Engineers (IEEE) University of Washington Student Chapter,** Seattle, WA, Apr. 2017 - Present
- ***Leader,*** **PEERs (Promoting Equity in Engineering Relationships), University of Washington,** Seattle, WA, Jan. 2014 – Present
- ***Founding Member,*** **Husky Student Experience Student Advisory Council**, **University of Washington,** Seattle, WA, Sept. 2014 – June 2017
- ***Workshop Lead,*** **Why You Need Internships and How to Get Them, Electrical Engineering, University of Washington,** Seattle, WA, June – Oct 2016
- ***Founding Member,*** **College of Engineering Student Advisory Council**, **University of Washington,** Seattle, WA, Oct. 2014 – June 2016
- ***Member,*** **Husky Leadership Initiative Project Committee**, **University of Washington,** Seattle, WA, Sept. 2015 – June 2016

ACADEMIC VOLUNTEER EXPERIENCE

- Reviewer, Medical Image Computing and Computer Assisted Interventions Conference (MICCAI), 2017-2018
- Reviewer, IEEE International Conference on Multimedia and Expo (ICME), 2016 - 2018
- Reviewer, Journal of Visual Communication and Image Representation, June 2016 - present
- Conference volunteer, IEEE International Conference on Multimedia and Expo (ICME), 2016
- Undergraduate admissions review committee member, Electrical Engineering, University of Washington, 2015

COMMUNITY SERVICE

- Participant, Healthcare Hackathon, University of Washington, Seattle, August 2016
- Panelist, Science Brief, WiSE Conference, University of Washington, Seattle, February 2015-16
- Volunteered for Engineering Discovery Days, an outreach program for elementary through high school students in the Seattle area organized by UW College of Engineering, explaining the concept of the exhibits to the students and helped them perform the experiments, 2011 – 2017
- Volunteer note-taker for Disability Student Services at University of Washington, Sept. 2012 – June 2013

AWARDS

- Husky 100, University of Washington, 2017
- Grace Hopper Celebration of Women in Computing Student Scholarship, 2017
- Graduate School Fund for Excellence and Innovation (GSFEI) Travel Award, 2017
- Graduate and Professional Student Senate (GPSS) Travel Grant, 2017
- Global Grand Challenges Summit Student Travel Grant, 2017
- iREDEFINE ECE (Improving the Diversity of Faculty in Electrical and Computer Engineering) Fellow, 2017
- Storytelling Fellow, UW Research Commons, 2017

- Eugene Beebe Scholarship, UW Graduate Opportunity Program (GO-MAP), 2015
- UW EE department scholarship to attend the Grace Hopper Conference, 2015
- Paul C. Leach Fellowship, 2013
- Best outgoing student award, 2009
- Overall college topper award for scoring the highest grade in 2nd year, 2007

BIBLIOGRAPHY

[1] R. A. Newcombe, A. J. Davison, S. Izadi, P. Kohli, O. Hilliges, J. Shotton, D. Molyneaux, S. Hodges, D. Kim and A. Fitzgibbon, "KinectFustion: real-time 3D reconstruction and interaction using a moving depth camera," *IEEE International Symposium on Mixed and Augmented Reality*, pp. 127-136, 201.

[2] H. Roth and M. Vona, "Moving Volume KinectFusion," in *Proceedings of the British Machine Vision Conference*, 2012.

[3] T. Whelan, M. Kaess, M. Fallon, H. Johannsson, J. Leonard and J. McDonald, "Kintinuous: Spatially extended KinectFusion," in *RSS Workshop on RGB-D: Advanced Reasoning with Depth Cameras*, Sydney, 2012.

[4] Y. Chen and G. Medioni, "Object Modeling by Registration of Multiple Range Images," in *IEEE International Conference on Robotics and Automation*, 1991.

[5] P. Besl and N. McKay, "A method for registration of 3D shapes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, pp. 239-256, 1992.

[6] Z. Zhang, "Iterative point matching for registration of free-form curves and surfaces," *International Journal of Computer Vision*, vol. 13, no. 2, pp. 119-152, 1994.

[7] J. Sturm, E. Bylow, F. Kahl and D. Cremers, "CopyMe3D: Scanning and Printing Persons in 3D," in *German Conference on Pattern Recognition (GCPR)*, 2013.

[8] D. G. Lowe, "Distinctive Image Features from Scale-Invariant Keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91-110, 2004.

[9] R. Inomata, K. Terabayashi, K. Umeda and G. Godin, "Registration of 3D Geometric Model and Color Images using SIFT and Range Intensity Images," in *International Symposium on Advances in Visual Computing*, 2011.

[10] B. Curless and M. Levoy, "A volumetric method for building complex models for range images," in *Proc. SIGGRAPH 96*, 1996.

[11] H. Pfister, M. Zwicker, J. van Baar and M. Gross, "Surfels: Surface Elements as Rendering Primitives," in *Proc. SIGGRAPH 2000*, 2000.

[12] J. Xie, Y. F. Hsu, R. Feris and M. T. Sun, "Fine Registration of 3D Point Clouds with ICP Using an RGB-D Camera," in *IEEE International Symposium on Circuits and Systems(ISCAS)*, 2013.

[13] F. Lu and E. Milios, "Global Consistent Range Scan Alignment for Environment Mapping," *Autonomous Robots*, vol. 4, pp. 333-349, 1997.

[14] J. Sprickerhof, A. Nuchter, K. Lingemann and J. Hertzberg, "A Heuristic Loop Closing Technique for Large-Scale 6D SLAM," *Journal for Control, Measurement, Electronics, Computing and Communications, Special Issue with selected papers from the European Conference on Mobile Robots 2009.*, 2011.

[15] K. Shoemake, "Animating rotation with quaternion curves," in *ACM SIGGRAPH*, 1985.

[16] M. Alexa, J. Behr, D. Cohen, S. Fleishman, D. Levin and C. T. Silva, "Computing and Rendering Point Set Surfaces," in *IEEE Transactions on Visualization and Computer Graphics*, 2003.

[17] X. Lin, "A New Kinect Depth Image Refinement Approach by Joint Trilateral Filter and Adaptive Joint Bilateral Filter," in *(under submission)*.

[18] H. Avron, A. Sharf, C. Greif and D. Cohen, "L1-sparse Reconstruction of Sharpe Point Set Surfaces," *ACM Transactions on Graphics*, vol. 29, 2010.

[19] M. Isenburg, Y. Liu, J. Shewchuk and J. Snoeyink, "Streaming Computation of Delaunay Triangulations," in *SIGGRAPH*, 2006.

[20] "http://www.rolanddg.co.uk," [Online]. Available: http://www.rolanddg.co.uk/files/>PX_DS_brochure.pdf. [Accessed 25 3 2014].

[21] K. He, J. Sun and X. Tang, "Guided image filtering," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 35, no. 6, pp. 1397-1409, 2013.

[22] J. Lu, K. Shi, D. Min, L. Lin and M. N. Do, "Cross-based local multipoint filtering," in *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, 2012.

[23] L. Xu, O. C. Au, W. Sun, Y. Li and J. Li, "Hybrid plane fitting for depth estimation," in *Signal & Information Processing Association Annual Summit and Conference (APSIPA ASC)*, 2012.

[24] K. Matsumoto, F. De Sorbier, and H. Saito, "Plane fitting and depth variance based upsampling for noisy depth map from 3D-ToF cameras in real-time," in *SciTePress*, 2015.

[25] J. Yang, X. Ye, K. Li and C. Hou, "Depth recovery using an adaptive color-guided auto-regressive model," in *Computer Vision -- ECCV 2012*, 2012.

[26] M. Camplani and L. Salgado, "Efficient spatiotemporal hole filling strategy for Kinect depth maps," in *IS&T SPIE Electronic Imaging*, International Society for Optics and Photonics, 2012.

[27] J. Liu, X. Gong and J. Liu, "Guided inpainting and filtering for Kinect depth maps," in *Pattern Recognition (ICPR), 2012 21st International Conference on*, 2012.

[28] Z. Wang, J. Hu, S. Wang and T. Lu, "Trilateral constrained sparse representation for Kinect depth hole filling," *Pattern Recognition Letters*, vol. 65, pp. 95-102, 2015.

[29] J. Canny, "A computational approach to edge detection," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 6, pp. 679-698, 1986.

[30] J. Xie, R. S. Feris, and M.-T. Sun, "Edge-Guided Single Depth Image Super-Resolution," *IEEE Transactions on Image Processing*, vol. 25, no. 1, pp. 428-438, 2016.

[31] H. Hirschmuller and D. Scharstein, "Evaluation of cost functions for stereo matching," in *Computer Vision and Pattern Recognition*, 2007.

[32] O. Riemens, O. Gangwal, B. Barenbrug and R.-P. Berretty, "Multistep joint bilateral depth upsampling," in *IS & T SPIE Electronic Imaging*, International Society for Optics and Photonics, 2009.

[33] Z. Wang, A. C. Bovik, H. R. Sheikh and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *Image Processing, IEEE Transactions on*, vol. 13, no. 4, pp. 600--612, 2004.

[34] S. Izadi, D. Kim, O. Hilliges, D. Molyneaux, R. Newcombe, P. Kohli, J. Shotton, S. Hodges, D. Freeman and A. Davison, "KinctFusion: real-time 3D reconstruction and interaction using a moving depth camera," in *Proceedings of the 24th annual ACM symposium on User interface software and technology*, 2011.

[35] M. Jaiswal, J. Xie, and M.-T. Sun, "3D object modeling with a Kinect camera," in *Signal and Information Processing Association Annual Summit and Conference (APSIPA), 2014 Asia-Pacific*, Siem Reap, Cambodia, Dec. 2014.

[36] M. Jaiswal, Y.-Y. Wang and M.-T. Sun, "Object Boundary Based Denoising for Depth Images," in *14th International Conference on Image Analysis and Recognition ICIAR 2017*, Montreal, Canada, 2017.

[37] J. Xie, R. S. Feris, and M.-T. Sun, "Edge-guided single depth image super resolution.," *IEEE Transactions on Image Processing*, vol. 25, no. 1, pp. 428-438, 2016.

[38] K. He, J. Sun and X. Tang, "Guided image filtering," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 6, pp. 1397-1409, 2013.

[39] A. Riemens, O. Gangwal, B. Barenbrug and R.-P. Berretty, "Multistep join bilateral depth upsampling," in *International Society for Optics and Photonics - Visual communications and image processing*, 2009.

[40] Intel Corporation, "software.intel.com," [Online]. Available: https://software.intel.com/en-us/articles/realsense-r200-camera. [Accessed 15 November 2017].

[41] Intel Corporation, "software.intel.com," [Online]. Available: https://software.intel.com/en-us/realsense/d400. [Accessed 15 November 2017].

[42] J. Park, Q.-Y. Zhou and V. Koltun, "Colored Point Cloud Registration Revisited," in *The IEEE International Conference on Computer Vision (ICCV)*, Venice, Italy, 2017.

[43] Y. LeCun, Y. Bengio and G. Hinton, "Deep Learning," *Nature Research*, vol. 521, no. 7553, pp. 436-444, 2015.

[44] A. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, J. Xiao, L. Yi and F. Yu, "ShapeNet: An Information-Rich 3D Model Repository," arXiv:1512.03012 [cs.GR], 2015.

[45] J. Long, E. Shelhamer and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015.

[46] C. Mehanian, M. Jaiswal, C. Delahunt, C. Thompson, M. Horning, L. Hu, S. McGuire, T. Ostby, M. Mehanian, B. Wilson, C. Champlin, E. Long, S. Proux, D. Gamboa, P. Chiodini, J. Carter, M. Dhorda, D. Isboke, B. Ogutu, W. Oyibo, E. Villasis, K. M. Tun, C. Bachman and D. Bell, "Computer-Automated Malaria Diagnosis and Quantitation Using

Convolutional Neural Networks," in *IEEE Conference on Computer Vision and Pattern Recognition*, Venice, Italy, 2017.

[47] World Health Organization, World malaria report 2015, 2016.

[48] World Health Organization and Center for Disease Control, Basic Malaria Microscopy: Tutor's guide, World Health Organization, 2010.

[49] C. Wongsrichanalai, M. Barcus, S. Muth, A. Sutamihardja and W. Wernsdorfer, "A review of malaria diagnostic tools: microscopy and rapid diagnostic test (RDT)," *The American journal of tropical medicine and hygiene,* vol. 77, pp. 119-127, 2007.

[50] H. Albert, Y. Manabe, G. Lukyamuzi, P. Ademun, S. Mukkada, B. Nyesiga, M. Joloba, P. C and M. Perkins, "Performance of three LED-based fluorescence microscopy systems for detection of tuberculosis in Uganda," *PLoS One,* vol. 5, no. 12, 2010.

[51] D. Durrhelm, P. Becker, K. Billinghurst and A. Brink, "Diagnostic disagreement - the lessons learned from malaria diagnosis in Mpumalanga," *South African medical journal = Suid-Afrikaanse tydskrif vir geneeskunde,* vol. 87, no. 5, pp. 609-611, 1997.

[52] M. Gatton and Research Malaria Microscopy Standards Work Group, Microscopy for the detection, identification, and quantification of malaria parasites on stained thick and thin blood films in research settings, World Health Organization, 2015.

[53] N. White, "The parasite clearance curve," *Malaria Journal,* vol. 10, no. 1, 2011.

[54] World Health Organization, Methods for surveillance of antimalarial drug efficacy, Geneva: World Health Organization, 2009.

[55] S. Ashraf, A. Kao, C. Hugo, E. Christophel, B. Fatunmbi, J. Luchavez, K. Lilley and D. Bell, "Developing standards for malaria microscopy: external competency assessment for malaria microscopists in the Asia-Pacific," *Malaria Journal,* vol. 11, no. 1, 2012.

[56] Y. LeCun, B. Boser, J. Denker, D. Henderson, R. Howard, W. Hubbard and L. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural Computation,* vol. 1, no. 4, pp. 541-551, 1989.

[57] A. Krizhevsky, I. Sutskever, and G. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097-1105.

[58] D. Cire{\c{s}}an, A. Giusti, L. Gambardella and J. Schmidhuber, "Mitosis detection in breast cancer histology images with deep neural networks," in *International Conference on Medical Image Computing and Computer-assisted Intervention*, Springer, 2013, pp. 411-418.

[59] K. He, X. Zhang, S. Ren and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770-778.

[60] D. Warhurst and J. Williams, "Laboratory diagnosis of malaria," *Journal of clinical pathology,* vol. 49, no. 7, 1996.

[61] J. Quinn, A. Andama, I. Munabi and F. Kiwanuka, "Automated blood smear analysis for mobile malaria diagnosis," *Mobile Point-of-Care Monitors and Diagnostic Device Design,* vol. 31, pp. 115-132, 2014.

[62] L. Rosado, J. Da Costa, D. Elias and J. Cardoso, "Automated detection of malaria parasites on thick blood smears via mobile devices," *Proceedings of Computer Science,* vol. 90, no. Elsevier, pp. 138-144, 2016.

[63] J. Vink, M. Laubscher, R. Vlutters, K. Silamut, R. Maude, M. Hasan and G. Haan, "An automatic vision-based malaria diagnosis system," *Journal of Microscopy,* vol. 250, no. 3, pp. 166-178, 2013.

[64] N. Linder, R. Turkki, M. Walliander, A. M{\aa}rtensson, V. Diwan, E. Rahtu, M. Pietik{\"a}inen, M. Lundin and J. Lundin, "A malaria diagnostic tool based on computer vision screening and visualization of Plasmodium falciparum candidate areas in digitized blood smears," *PLoS One,* vol. 9, no. 8, 2014.

[65] G. D{\'\i}az, F. Gonz{\'a}lez and E. Romero, "A semi-automatic method for quantification and classification of erythrocytes infected with malaria parasites in microscopic images," *Journal of Biomedical Informatics,* vol. 42, no. 2, pp. 296-307, 2009.

[66] C. Delahunt, C. Mehanian, L. Hu, S. McGuire, C. Champlin, M. Horning, B. Wilson and C. Thompson, "Automated microscopy and machine learning for expert-level malaria field diagnosis," in *Global Humanitarian Technology Conference (GHTC)*, IEEE, 2015, pp. 393-399.

[67] L. Rosado, J. Correia da Costa, D. Elias and J. Cardoso, "A review of automatic malaria parasites detection and segmentation in microscopic images," *Anti-Infective Agents,* vol. 14, no. 1, pp. 11-22, 2016.

[68] D. Das, R. Mukherjee and C. Chakraborty, "Computational microscopic imaging for malaria parasite detection: a systematic review," *Journal of microscopy,* vol. 260, no. 1, pp. 1-19, 2015.

[69] S. Stugger, "Fluorescence microscope examination of bacteria in soil," *Canadian journal of research,* vol. 26, no. 2, pp. 188-193, 1948.

[70] World Health Organization, Malaria microscopy quality assurance manual-version 2, World Health Organization, 2016.

[71] R. Girshick, J. Donahue, T. Darrel and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition,* 2014, pp. 580-587.

[72] J. Redmon, S. Divvala, R. Girshick and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 779-788.

[73] P. Felzenszwalb, R. Girshick, D. McAllester and D. Ramanan, "Object detection with discriminatively trained part-based models," *IEEE transactions on pattern analysis and machine intelligence,* vol. 32, no. 9, pp. 1627-1645, 2010.

[74] J. Uijlings, K. Van De Sande, T. Gevers and A. Smeulders, "Selective search for object recognition," *International Journal of computer vision,* vol. 104, no. 2, pp. 154-171, 2013.

[75] R. Girshick, "Fast r-CNN," in *Proceedings of the IEEE international conference on computer vision,* 2015, pp. 1440-1448.

[76] S. Ren, K. He, R. Girschick and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," in *Advances in neural information processing systems*, 2015, pp. 91-99.

[77] International Telecommunication Union, *Recommendation ITU-R BT601-7,* 2015.

[78] B. Boser, I. Guyon, and V. Vapnik, "A training algorithm for optimal margin classifiers," in *Proceedings of the fifth annual workshop on computational learning theory*, ACM, 1992, pp. 144-152.

[79] A. Ng, "Feature selection, L1 vs. L2 regularization and rotational invariance," in *Proceedings of the twenty-first international conference on machine learning*, ACM, 2004.

[80] R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the royal statistical society*, pp. 267-288, 1996.

[81] R. Rosipal and N. Kr{\"a}mer, "Overview and recent advances in partial least squares," *Lecture notes in computer science*, vol. 3940, 2006.

[82] C. Metz, "Evaluation of digital mammography by ROC analysis," *Excerpta Medica*, vol. 1119, pp. 61-68, 1996.

[83] N. Otsu, "A threshold selection method from gray-level histograms," *IEEE transactions on systems, man and cybernetics*, vol. 9, no. 1, pp. 62-66, 1979.

[84] J. Kittler and J. Illingworth, "Minimum error thresholding," *Pattern Recognition*, vol. 19, no. 1, pp. 41-47, 1986.

[85] J. Brenner, B. Dew, B. Horton, T. King, P. Neurath and W. Selles, "An automated microscope for cytologic research a preliminary evaluation," *Journal of histochemistry and cytochemistry*, vol. 24, no. 1, pp. 100-111, 1976.

[86] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5-32, 2001.

[87] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama and T. Darrell, "Caffe: a Convolutional architecture for fast feature embedding," in *Proceedings of the 22nd ACM international conference on multimedia*, 2014, pp. 675-678.

[88] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[89] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1-9.

[90] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein and F.-F. Li, "Imagenet large-scale visual recognition challenge," *International Journal of computer vision*, vol. 115, no. 3, pp. 211-252, 2015.

[91] A. Sharif Razavian, H. Azizpour, J. Sullivan and S. Carlsson, "CNN features off-the-shelf: an astounding baseline for recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, 2014, pp. 806-813.

[92] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng and T. Darrell, "Decaf: A deep convolutional activation feature for generic visual recognition," in *Internation conference on machine learning*, 2014, pp. 647-655.

[93] J. Yosinski, J. Clune, Y. Bengio and H. Lipson, "How transferable are features in deep neural networks?," in *Advances in neural information processing systems*, 2014, pp. 3320-3328.

[94] D. Cox, "The regression analysis of binary sequences," *Journal of the royal statistical society*, pp. 215-242, 1958.

[95] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang and C.-J. Lin, "LIBLINEAR: A library for large linear classification," *Journal of machine learning research*, vol. 9, no. 8, pp. 1871-1874, 2008.