THE UNIVERSITY
of EDINBURGH

3D Data Fusion by Depth Refinement and Pose Recovery

Can Pu

Doctor of Philosophy
Institute of Perception, Action and Behaviour
School of Informatics
University of Edinburgh
2019

Abstract

Refining depth maps from different sources to obtain a refined depth map, and aligning the rigid point clouds from different views, are two core techniques. Existing depth fusion algorithms do not provide a general framework to obtain a highly accurate depth map. Furthermore, existing rigid point cloud registration algorithms do not always align noisy point clouds robustly and accurately, especially when there are many outliers and large occlusions. In this thesis, we present a general depth fusion framework based on supervised, semi-supervised, and unsupervised adversarial network approaches. We show that the refined depth maps are more accurate than the source depth maps by depth fusion. We develop a new rigid point cloud registration algorithm by aligning two uncertainty-based Gaussian mixture models, which represent the structures of the two point clouds. We show that we can register rigid point clouds more accurately over a larger range of perturbations. Subsequently, the new supervised depth fusion algorithm and new rigid point cloud registration algorithm are integrated into the ROS system of a real gardening robot (called TrimBot) for practical usage in real environments. All the proposed algorithms have been evaluated on multiple existing datasets to show their superiority compared to prior work in the field.

Lay Summary

The target of this thesis is to help a robot see the 3D world better. It focuses on how to make a robot see the 3D structure precisely and how to localize a moving robot accurately in the 3D world. Two original techniques were developed to solve the two problems above. The first technique enables us to combine depth information from two sensors to get more accurate depth information. We developed a general framework that could be used for different types of sensors, and could also use different amounts of initial information from the user for training. The second technique is an improved method for aligning two different sets of rigidly connected 3D points, such as getting one by scanning an object or scene from two different viewpoints. The proposed algorithm is robust to strong noise, many false 3D points, and large amounts of data that are missing because of occlusion. These two techniques were adapted to a prototype robot called TrimBot successfully and achieved a good performance in the real environment. The proposed methods in this thesis can be widely used in virtual & augmented reality, autonomous driving, robotics etc. to facilitate the related research or product development.

Acknowledgements

I thank my supervisor Robert Fisher for his detailed supervision, which has cultivated me to be an expert in 3D data fusion. I thank my families and friends for their understanding and help. I thank TrimBot2020 project [Grant Agreement No. 688007, URL: http://trimbot2020.webhosting.rug.nl/] from the European Union Horizon 2020 programme for funding me through my whole PHD study. I thank all the colleagues around me and those people who helped me. I thank the University of Edinburgh for providing me with a good atmosphere to conduct the research.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified. The work published within this thesis has been published in the following peer reviewed articles with attribution and contribution as follows:

Can Pu, Nanbo Li, Radim Tylecek, and Bob Fisher. Dugma: Dynamic uncertainty based gaussian mixture alignment. In 2018 International Conference on 3D Vision (3DV), pages 766-774. IEEE, 2018. (**oral presentation**)

Can Pu, Runzi Song, Radim Tylecek, Nanbo Li, and Robert B Fisher. Sdf-man: Semi-supervised disparity fusion with multi-scale adversarial networks. Remote Sensing, 11(5):487, 2019.

Can Pu and Robert B Fisher. UDFNET: Unsupervised Dispairity Fusion with Adversarial Networks. In 2019 26th IEEE International Conference on Image Processing (ICIP). IEEE, 2019. (**oral presentation**)

Contribution:

- Pu: developed theory, implemented theory, ran experiments, wrote papers;
- Song: helped with experiments; (8 hours' work on Sdf-man)
- Tylecek: helped with experiments; (8 hours' work on DUGMA and Sdf-man)
- Li: helped with experiments; (8 hours'work on DUGMA and Sdf-man)
- Fisher: helped with design of theory, experiments and writing of the papers.

(*Can Pu*)

Table of Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

Refining initial depth maps from the same view point and fusing depth maps from different views are core techniques, which are important in many areas such as 3D reconstruction, robot navigation, and augmented & virtual reality. However, there are many difficult problems (eg: low accuracy and robustness) involved in making depth fusion (Figure 1.1) and point cloud registration (Figure 1.2) work robustly and accurately in real environments, which is the research motivation for this proposed thesis.

In this chapter, an overview of the whole thesis is provided. First, the research objectives are introduced, followed by a description of existing problems in the 3D fusion area. Subsequently, the original contributions to solving each problem are stated, and finally, a structural overview of the whole thesis is presented.

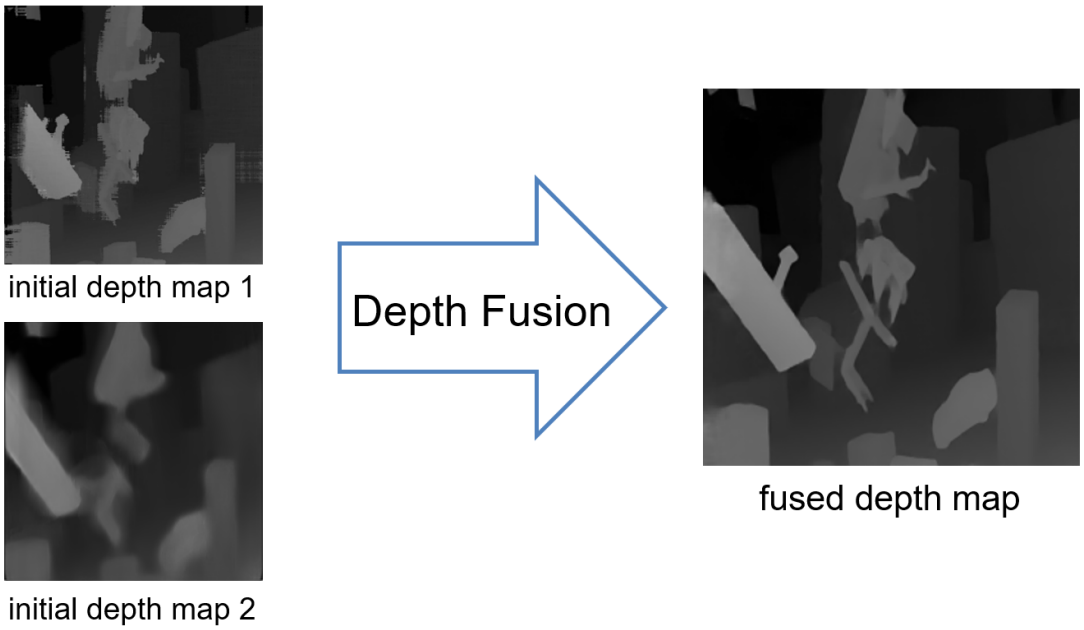


Figure 1.1: The figure shows the concept of depth fusion. The input to the depth fusion algorithm ('Depth Fusion') is some initial depth maps ('initial depth map 1', 'initial depth map 2'). After depth fusion, a more accurate fused depth map ('fused depth map') is output.

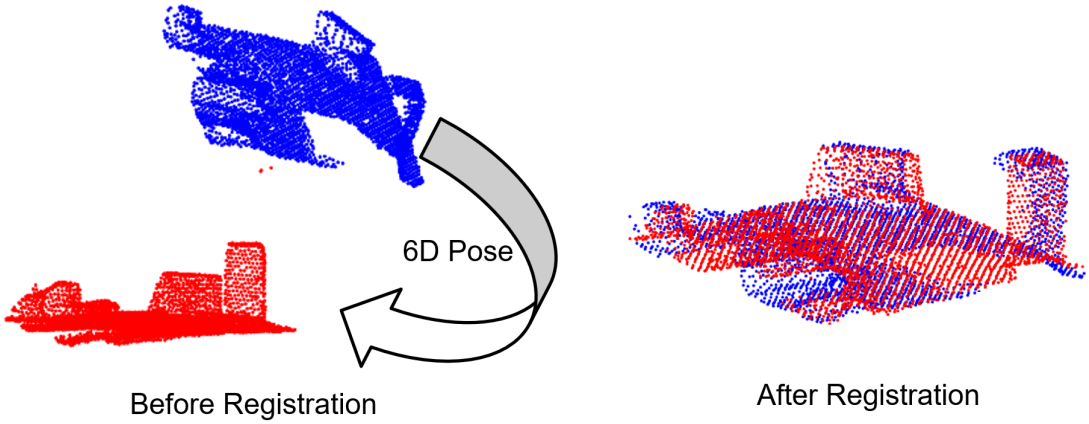


Figure 1.2: The figure shows the concept of point cloud registration. The input to the point cloud registration algorithm is two point clouds (shown in 'Before Registration') in different views. After registration, a relative 6D pose is output, which can transform one point cloud to make both point clouds overlap well (shown in 'After Registration').

1.1 Thesis Objectives

There are currently many depth acquisition methods. Compared with active vision (such as Time of flight (ToF), Lidar, and Radar), passive vision (such as stereo vision

and monocular depth estimation) is cheaper and denser, but coarser. Every method has its own advantages and disadvantages. For example, Lidar is accurate, but expensive and sensitive to humidity; ToF is cheaper but sensitive to infrared light; and stereo vision is cheaper but sensitive to textureless areas or repetitive texture areas. To obtain an accurate depth map[1] robustly, fusing depth maps from multiple sources is a good solution considering cost and performance. Accordingly, depth fusion is the first objective under the assumption that the initial depth inputs are from the same view at the same time for any scene or from the same view at any time for the static scene.

The point cloud from a cheap sensor (such as stereo vision cameras [82, 107]) is usually noisy. Obtaining the relative 6D pose between different views is required in many areas, such as 3D reconstruction and robot navigation. Developing a robust rigid point cloud registration is necessary, especially when there are many outliers, large occlusions, and strong noise. Accordingly, rigid noisy point cloud alignment is the second objective under the assumption that the two point clouds have partial overlapping areas and the scene is rigid.

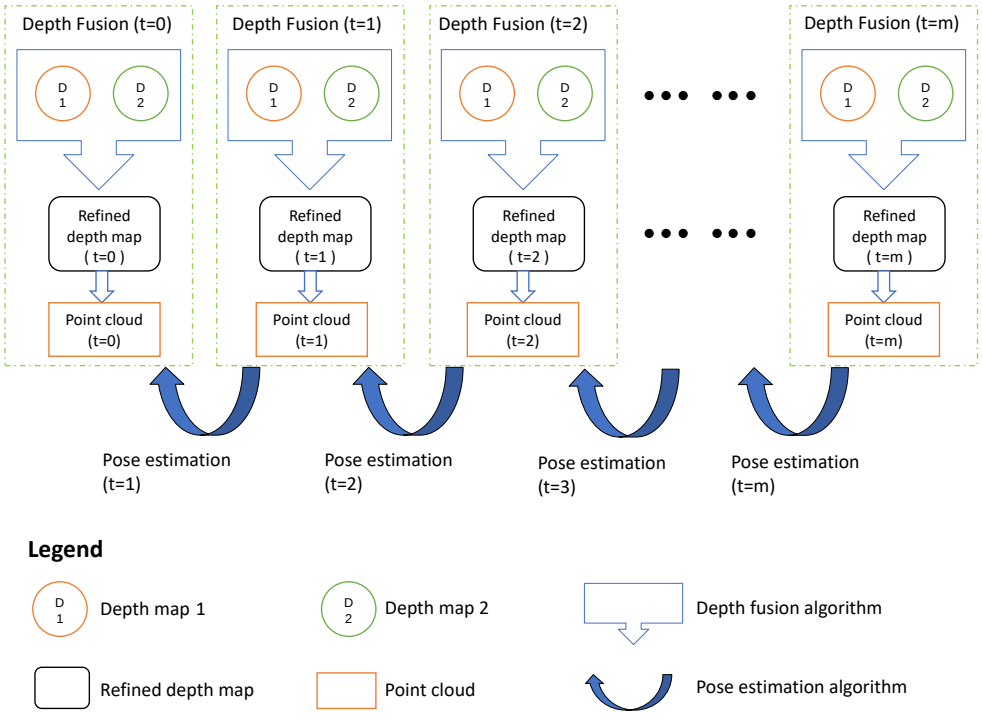


Figure 1.3: This figure shows the overall pipeline of the work presented in this thesis. See text for a description of the diagram.

Figure 1.3 shows the main fusion pipeline. Regarding time t, two depth acquisition

[1]The description of 'depth map' in Wikipedia [2]: an image or image channel that contains information relating to the distance of the surfaces of scene objects from a viewpoint.

methods are used to obtain two initial depth maps $D1$ and $D2$ in the same view. The proposed depth fusion methods are used to fuse these two initial depth maps to obtain a refined depth map at time t. Subsequently, the refined depth map is converted into the corresponding point cloud. Finally, the relative 6D pose between time t and time $t+1$ can be recovered by the proposed 6D pose estimation method. The whole system is integrated into the ROS system [123] to ensure each component in this pipeline runs simultaneously. Accordingly, integration of the whole pipeline in the ROS system to make it work on a real application is the third target.

The research in this thesis can be adapted to many existed physical systems containing multiple sensors, such as a robot called TrimBot2020 (see Figure A.2 in Appendix A) and an autonomous driving vehicle from AutoVision project [65] (See Figure 1.4). One motivation for the proposed research is to promote industrial and economic development.

Figure 1.4: AutoVision physical platform [65]. **Left**: The autonomous driving vehicle. **Right**: A close-up of the multi-sensor system.

1.2 Problem Statement

Depth fusion from multiple sources [14, 30, 38, 52, 94, 104, 110, 114] has been a difficult problem for many years. Different sensors use different principles to estimate depth information, which results in different uncertainty estimates [22, 107]. Given various uncertainty estimates for each sensor, there is no common depth fusion framework to realise different kinds of depth fusion. Additionally, it is hard to obtain extremely accurate uncertainty information, which limits the accuracy of the refined depth map. Although there are existing depth fusion methods [110, 114] based on deep learning that do not require uncertainty information, all of them neglect the contribution of the auxiliary image information (such as intensity and gradient images). The first

end-to-end adversarial network has been developed to solve the depth fusion problem.

Noisy point cloud alignment has been a difficult problem for decades. Outliers, strong noise, and large occlusions easily mislead the rigid point cloud registration process. Methods [32, 116, 134, 146, 156, 169] based on the iterative closest point (ICP) algorithm [23] cannot eliminate these problems, because of their underlying assumptions. They assume that point-to-point correspondences exist, but in real cases this is not true. Methods [26, 47, 86, 162] based on feature matching suffer from the effects of extreme noise conditions, which misleads the feature extraction. Regarding the methods [27, 28, 35, 74, 76, 89, 103, 138, 142, 167] based on probability alignment, they failed to represent the point cloud in a probability model considering the physical uncertainty from the real sensors. Further, the registration convergence basin is narrow if the extremely time-consuming branch and bound optimisation method [28, 89, 156] is not used. In this thesis an uncertainty-based GMM (Gaussian Mixture Model) has been developed for the first time to represent the real structure of the point cloud and the proposed registration algorithm obtains a wider convergence basin by convolving the two GMM (representing moving point cloud and static point cloud) in the whole 3D space.

Presently, many algorithms (eg: [29, 97, 166]) based on deep learning work robustly and accurately on labelled datasets, but perform poorly in real environments. Making the proposed algorithms generalise to work in the real environments, and integrating them in the ROS system to make each component interact with each other simultaneously, is required to power practical applications. The methods presented here have been adapted for use on a real outdoors gardening robot called Trimbot.

1.3 Original Contributions

The original contributions contain three parts: theory and algorithms; public datasets; code and data. They will be presented as follows:

1.3.1 Theory and Algorithms

Regarding depth fusion from multiple sources, this is the first time an end-to-end general framework has been developed for different depth fusion tasks (such as ToF-stereo fusion, monocular-stereo fusion, stereo-stereo fusion, and stereo-Lidar fusion). The general framework is realised by training an adversarial network, which can learn the

disparity Markov Random Field of the real environment more successfully. This is the first proposition of supervised, semi-supervised, and unsupervised depth fusion methods by adding the auxiliary image-based information (such as intensity and gradient images).

For supervised depth fusion in Chapter 3:

1. Improved fusion accuracy by using a network that learns the disparity relationships among pixels from examples without any prior knowledge.
2. Increased robustness by fusing intensity and gradient information as well as depth data.
3. Proposed a common network methodology allowing different kinds of sensor fusion without requiring detailed knowledge of the performance of each sensor.

For semi-supervised depth fusion in Chapter 3:

1. Reduced the labeled data requirements drastically by using the proposed semi-supervised strategy.

For unsupervised depth fusion in Chapter 4:

1. An efficient unsupervised strategy by combining global disparity initialisation and local refinement.
2. An indirect method using an adversarial network to force the disparity Markov Random Field of the refined disparity map to be close to the real disparity field.
3. An unsupervised end-to-end uncertainty-based pipeline to fuse any disparity input.

Regarding rigid point cloud alignment, this is the first time an uncertainty-based Gaussian mixture model has been used to describe the point cloud's uncertainty information. By designing a more robust energy function, the convergence area is successfully widened.

For rigid point cloud alignment in Chapter 5:

1. Incorporation of the invariant 3D uncertainty distribution information (represented by a Gaussian function with a physical covariance) into the dynamic registration;

2. A bridge to make the two point clouds interact with each other by creating a novel point proximity weight term;

3. A more robust energy function and efficient approximation to the optimisation step that greatly reduces computational complexity.

1.3.2 Datasets

Besides the contributions in the proposed algorithm mentioned above, four datasets have been created (URL: `https://github.com/Canpu999/DUGMA/tree/master/dataset_all`) for the rigid point cloud registration.

Datasets for Point Cloud Registration in Chapter 5:

1. '2D FISH MODEL': The synthetic dataset for 2D point cloud registration is generated from the Gatorbait100 database using different sampling rates. It is used in Chapter B.1 and some samples from the dataset are in Figure B.1.

2. '2D SHAPE MODEL': The synthetic dataset for 2D point cloud registration is generated from 100 different objects to test sensitivity to the shape. It is used in Chapter B.2 and some samples from the dataset are in Figure B.8.

3. '3D SHAPE': The real dataset is generated from a real garden in the Trimbot2020 project. It is used in Chapter 5.3.1 and (d)(e)(f) in Figure 5.1 are three samples from the dataset.

4. 'Kinect Application': The real dataset containing 30 different scenes is generated from two Kinect V2 sensors. It is used in Chapter 5.3.2 and one sample from the dataset is in Figure 5.8.

1.3.3 Code and Data

The corresponding code and data in this thesis have been (or will be) released to the public to accelerate progress in related disciplines.

Code and Data in This Thesis:

- The code and data in Chapter 3 (for depth fusion) and Chapter 6 (for 3D reconstruction of the real garden) have been submitted to project Trimbot2020 and will be released at the end of project Trimbot2020 (URL: `http://trimbot2020.webhosting.rug.nl/`). The ground truth data in the real garden can be used for training and evaluation of depth estimation and 6D pose recovery algorithms.

- All the experimental results in Chapter 4 (for depth fusion) have been released to URL: `https://github.com/Canpu999/UDFNet`. Readers can check the proposed algorithm's performance on each sample.
- The code and data in Chapter 5 (for 3D point cloud registration) has been released in URL: `https://github.com/Canpu999/DUGMA`. The included datasets could help develop and evaluate the 3D point cloud registration algorithms.

1.4 Thesis Overview

Chapter 1 provides an overview of the whole thesis, including the objectives, existing problems, proposed contributions, and structural overview of the thesis. Chapter 2 introduces the literature and background in depth acquisition, depth fusion, rigid point cloud registration and generative adversarial networks. Chapter 3 introduces the proposed supervised & semi-supervised depth fusion algorithm. Chapter 4 introduces the proposed unsupervised depth fusion. Chapter 5 introduces the rigid point cloud registration. Chapter 6 introduces the fusion pipeline in the ROS system and real performance of the corresponding algorithms in a real garden. Chapter 7 presents a conclusion about the whole thesis. Appendix A is a description of the Trimbot2020 Garden Dataset. Appendix B shows the performance of the proposed algorithm [118] in 2D point cloud registration.

Chapter 2

Literature Review and Background

In this chapter, the relevant literature and background are reviewed. As the pipeline in the previous chapter describes, different depth acquisition methods are used to obtain the depth maps for the scene. Multiple depth maps are fused from different sources to refine the initial depth maps, convert the refined depth map to a 3D point cloud, and use the point cloud registration algorithm to obtain the relative 6D pose. The literature review and background are introduced along with that pipeline. Given that the research questions in this thesis are depth fusion and rigid point cloud registration, only the depth fusion and rigid point cloud registration are introduced in more detail and the rest are described briefly in this chapter. The latest progress in four areas, which are relevant to this research study, is introduced in the following order: depth acquisition (Section 2.1, providing initial depth maps as input to research question 'depth fusion'), depth fusion (Section 2.2, the first research question in this thesis), rigid point cloud registration (Section 2.3, the second research question in this thesis) and generative adversarial networks (Section 2.4, providing basic network frameworks to research question 'depth fusion'). Partial content is from the published papers [117, 118, 119].

2.1 Depth Acquisition

To obtain the initial depth maps for the depth fusion, a variety of algorithms or sensors can be chosen. However, depth acquisition methods are not the research target in this thesis but will provide the initial input depth data to the depth fusion network in Chapter 3 and Chapter 4. In the following section, monocular depth estimation, stereo vision, and some other depth algorithms or sensors are introduced briefly.

2.1.1 Monocular Depth Estimation

Monocular depth estimation refers to an algorithm that obtains the depth map using only one camera. In 2014, Eigen et al. [46] used a neural network with two components (one for coarse depth map generation, and the other for the refinement) to regress on the depth directly. It is hard to generalise, because it is impossible to obtain the depth value using only one image from a physics aspect. Obtaining a range of 2D images in different views (such as by using one moving camera) to recover the 3D structure is feasible, which is called structure from motion (SFM) in the computer vision field, or visual simultaneous localisation and mapping (SLAM) in robotics. Figure 2.1 shows the epipolar constraint as used by SFM.

Based on the theory of SFM, many algorithms using a monocular camera have been proposed in the last several years, such as [49, 50, 102, 131, 147]. Readers are directed to a recent survey [128]. Besides the classical algorithms mentioned above, many algorithms based on deep learning have also emerged. For example, in 2017, Ummenhofer et al. [143] framed structure from motion as a supervised learning problem. A monocular camera was used to input two successive images into the neural network, which consisted of three sub encoder-decoder networks. The depth map and egomotion were output. More recently, Huang et al. [71] inputted multiple posed images and produced a set of plane-sweep volumes to obtain a high-quality disparity map. Similarly, Zhou et al. [166] accumulated relevant information in a cost volume, which was centered at the current predicted depth map. The key frame image and cost volume were combined by the mapping network to update the depth estimate, to make full use of image-based priors and previous depth estimates. In the latest work, Wang et al. [147] proposed a framework to estimate the depth map and visual odometry simultaneously by using a monocular camera. They adopted convolutional long short-term memory units to convey temporal information from previous views into the current view for depth and visual odometry estimation, which created a loss

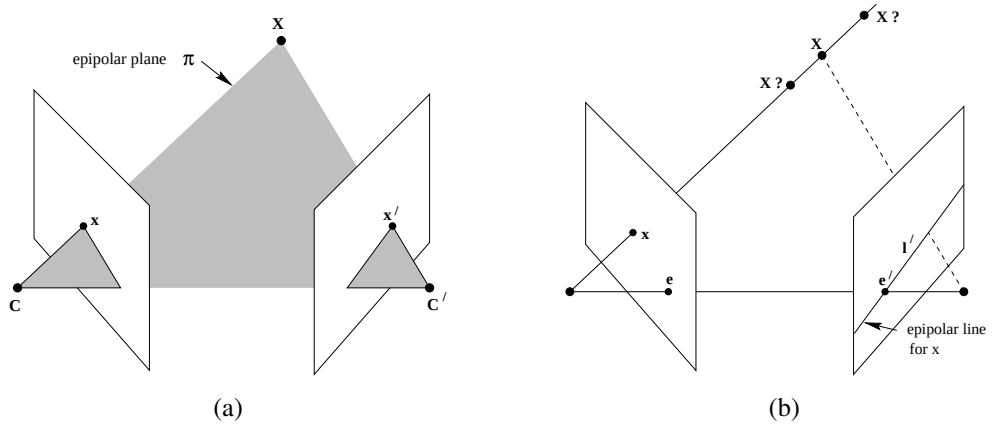


Figure 2.1: Point correspondence geometry from [64]. (a) The two image planes corresponding to camera centre C and C' are from one moving camera or two fixed cameras. X denotes a 3D point in the 3D space. The plane which contains the two camera centres (C, C') and the 3D point X is called the epipolar plane π. x and x' are the 2D points on two image planes when projecting X to the corresponding view. (b) A ray from the camera centre C and x is imaged as an epipolar line l' on the image plane corresponding to camera centre C'. After finding the 2D point x' corresponding to x on the epipolar line l', the 3D point X can be obtained by triangulation. e or e' is called an epipole, which is the intersection point between the image plane and the line containing two camera centres.

term by image reprojection from multiple views. The optical flow from different frames were used to impose a forward-backward flow-consistency constraint when training the network. The proposed method achieved the state-of-art performance in this area.

Except for the methods based on supervised learning mentioned above, some unsupervised learning methods also achieved good performance. For example, Godard et al. [59] proposed the use of left and right intensity consistency to obtain the depth map unsupervisedly. More specifically, they adopted the geometric constraint by using one pair of rectified stereo images when training. They inputted the left image from the left stereo vision camera into the neural network, and obtained the disparity map on the left and right view. They treated the disparity maps as a hidden layer and used them to reconstruct the left and right intensity images. Further, by forcing the reconstructed intensity images to be similar to the real intensity images from the stereo vision cameras, they realised unsupervised training without the requirement of the ground truth depth data, but with the requirement of the real intensity images. The pipeline for this is

presented in Figure 2.2. Subsequently, some similar work (but with different training strategy or neural network structure) has appeared, such as [31, 112, 152]. In the latest work, Chen et al. [31] leveraged semantic understanding to improve the monocular depth estimation effectively. Their proposed network took the RGB image as input and encoded it into a scene representation. Along with the introduced identity layer t ($t = 0$ indexes semantic meaning and $t = 1$ depth estimation), the scene representation above can be decoded into the semantic or depth output. They used supervised learning to train the semantic part and used the similar strategy in Monodepth [59] to train the depth part by treating the depth output as a hidden layer and reconstructing the intensity image in the other view to get the intensity loss from the left-right intensity consistency. Intensity information in images is sensitive to scene lighting and camera parameters. In order to be more robust, they reconstructed the semantic image in the other view and adopted the left-right semantics consistency as one of the loss terms to refine the depth output. A semantics-guided disparity smoothness term was also proposed to do the smoothing. In unsupervised monocular depth estimation area, it achieved the state-of-art performance.

2.1.2 Stereo Vision

Stereo vision algorithms can be regarded as a specific special case of known relative camera pose. Stereo vision has a long history, and can be classified into two main categories: methods based on classical analysis, and methods based on deep learning. The most famous classical stereo vision algorithm is the semi-global matching method [67], which achieves pixelwise matching by using mutual information and a global smoothness approximation. Its stereo vision pipeline could be decomposed into three key steps: feature extraction, matching cost aggregation and disparity estimation. With the emergence of deep learning technique, some researchers tried to replace some key steps in the classical pipeline with deep neural nets. For example, In the feature extraction stage, MC-CNN [161] used a convolutional neural network to learn the similarity between two small image patches from the left and right view to design the matching cost. After obtaining the stereo matching cost, the post-processing pipeline similar to [67] was used. In the latest work, Nie et al. [106] proposed a unary feature descriptor encapsulating convolutional features into a more discriminative representation for feature matching. For matching cost aggregation, GC-net [81] replaced the matching cost aggregation step by using 3D convolutions to incorporate contextual

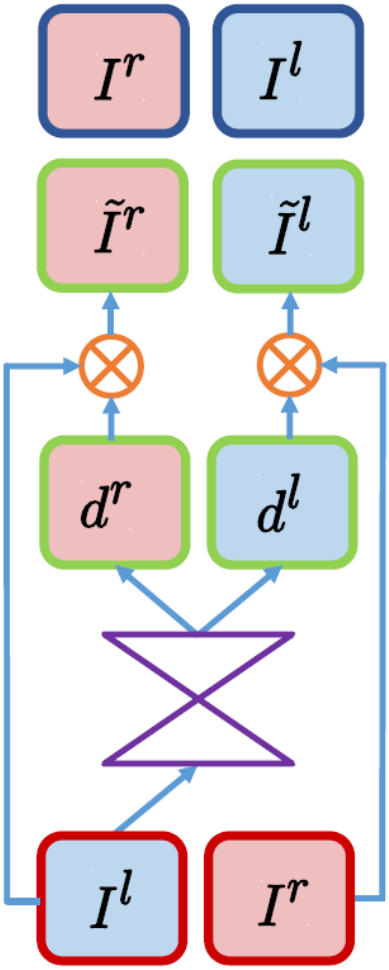


Figure 2.2: Unsupervised disparity estimation pipeline from [59]. I^l and d^r are the left intensity image and corresponding disparity map. I^r and d^l are the right intensity image and corresponding disparity map. $\tilde{I}^r$ and $\tilde{I}^l$ are the reconstructed right and left intensity image. The input to the neural network is I^l and the output is d^r and d^l.

information over the cost volume. A fully differentiable soft argmin function was used to regress sub-pixel disparity values from the disparity cost volume. However, using 3D convolutional layer is memory-consuming and computationally costly because it has cubic memory/computational complexity. In the latest work, GA-Net [163] used two neural net layers to replace the 3D convolutional layer. One is for approximation of the semi-global matching and the other is for refining the thin structures by following a traditional cost filtering strategy. It achieved the state-of-art results.

There are stereo vision methods with deep neural nets, which completely replace the whole classical pipelines from another point of view. In 2015, Dosovitskiy et al. [44] initially designed an end-to-end convolutional neural network to obtain the optical

flow (a disparity map can be regarded as a type of special optical flow) between two images. It provided two simple neural network architectures. FlowNetSimple used one encoder and one decoder. FlowNetCorr used one encoder for each input image and concatenated the output of the two encoders together to feed into the common decoder. Subsequently, many stereo vision algorithms based on deep learning emerged, such as Dispnet [97], PSMNet [29], iResnet [88] and MADNet [141]. Besides the stereo vision algorithms above aiming at accuracy improvement, some latest work [115, 140] started to concentrate on the domain generation of stereo vision algorithms, which make the proposed stereo vision algorithms work better in the new environment. Some latest work [155] started to concentrate on high-res stereo computing with limited computing/memory resources by searching for correspondences incrementally from coarse to fine.

2.1.3 Other Depth Estimation Methods

Besides the previously mentioned passive methods for obtaining scene depth, there are many active methods to reconstruct the 3D structure of the scene. These include ToF sensors [125], structured-light 3D scanners [57], light detection and ranging (Lidar) [66], radio detection and ranging (Radar) [137], and ultrasonic sensors [80]. Given that the speed of light is a constant, the ToF sensor [11] can estimate the distance between the camera and the targets by calculating the round trip time of the projected light. The light signal can be from an LED (which is sensitive to infrared light) or from Lidar (which is sensitive to its own temperature, environment, and noise). The structured-light 3D scanner [10] measures the 3D structure with the help of a camera system and projected light patterns. Lidar [4] is a method of measuring the distance in 3D space by using pulsed laser light and measuring the reflected pulses. Subsequently, the return wavelengths and time could be used to predict the 3D structure. Lidar is predominantly used to produce high-resolution maps. Radar [8] is a system that uses radio waves to predict the velocity, distance, angle, and range of an object. An ultrasonic sensor [12] is a type of acoustic sensor, which calculates the distance by sending a signal and receiving an echo.

Different depth sensors have their own advantages and disadvantages. However, it is difficult to make a single sensor work in all general cases. Multiple depth sensor fusion is a new trend to make the system reliable enough for many applications, such as autonomous driving and robotics.

2.2 Depth Fusion

To obtain a reliable and accurate depth map, fusing multiple depth maps from different sources is an alternative, which is the first research target in this thesis. Currently there are two main categories: classical methods and methods based on deep learning.

2.2.1 The Classical Methods

The majority of fusion work [14, 15, 30, 38, 52, 94] shares the same pipeline architecture, which initially estimates the uncertainty of each pixel, and refines the depth map based on those confidence maps. For example, Dal Mutto et al. [38] used the IR frequency of a ToF sensor to estimate depth map pixelwise uncertainty, and used the similarity of image patches in the stereo images to estimate the confidence of pixels in the stereo depth map. A MAP-MRF framework refined the depth map. Subsequently, Marin et al. [94] also utilised sensor physical properties to estimate the pixelwise confidence for the ToF depth map, and used an empirical model based on the global and local cost of stereo matching to calculate the confidence map for the stereo vision sensor. The extended LC (Locally Consistent) technique was used to fuse the depth maps, based on each confidence map. To obtain a more accurate confidence map for fusion, Agresti et al. [14] used a simple convolutional neural network for uncertainty estimation, and used the LC technique from [94] for the fusion. More recently, Chen et al. [30] proposed the use of edge-selective joint filtering to realise variational fusion of depth maps from ToF and a stereo vision camera. They also estimated the pixelwise confidence of the depth maps first using a Gaussian function, and upsampled the low-resolution ToF depth map by using edge selective joint filtering. Horizontal and vertical discontinuity maps were extracted from the depth maps. Based on the discontinuity and confidence maps, they fused the two depth maps to obtain a final refined depth map. In the latest work, Agresti et al. [15] extended the approach in [14] to realize stereo-ToF depth fusion and achieved the state-of-art performance in the classical depth fusion area. Figure 2.3 shows the flowchart of their proposed method. The ToF data (ToF depth and ToF amplitude) from ToF sensor is projected to the right view of the stereo setting in order to have the same reference coordinates. The projected ToF data is still low-resolution and upsampled to the resolution of the stereo images in the right view by a combination of bilateral filtering and segmentation clues. The SGM in OpenCV [6] was adopted to do the stereo calculation. The intensity difference between the right intensity image and reconstructed right image from left image and disparity map was calculated as one

uncertainty source. Then, interpolated ToF disparity, amplitude and left-right intensity difference, stereo disparity were fed into an ad-hoc CNN to calculate the confidence of each disparity pixel. An extended LC (Local Consistence) framework was used to do disparity fusion considering the confidence information and initial disparity maps. They learned the parameters in the CNN using a synthetic dataset and their experiments showed it could generalize well to a real experimental setting when using a Kinect V2 and ZED camera [107].

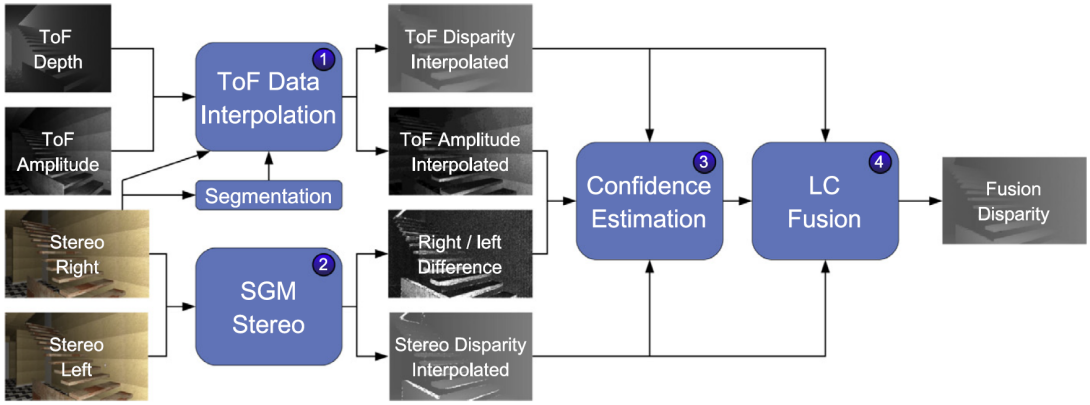


Figure 2.3: The pipeline of the proposed method in [15].

2.2.1.1 Uncertainty Estimation

Estimating the confidence of the depth map for stereo vision has a long history, and can be traced back to the 1990s [78, 96]. In 1991, Kanade and Okutomi [78] assumed that the disparity of all the points in the windows (which were centered on the corresponding points in the left and right image) obeyed a Gaussian distribution. The scalar variance of the point in that window would increase with the distance between it and the center point. Given an initial disparity map, they successfully used a statistical model, which represented the expectation and variance of disparity of points over the window for the stereo matching algorithm. However, the initial disparity map would affect the matching points in the left and right image, which in turn would ultimately influence the reliability of the model. Additionally, if only the distribution of disparity is considered, the distribution of the corresponding 3D points will be distributed in one dimension along a ray. Alternatively, from another angle, Matthies et al. [96] considered corresponding points in left and right images as normally distributed random vectors. However, given that the triangulation is non-linear, the realistic 3-D distribution is non-Gaussian. To simplify the process, they adopted 3-D Gaussian distributions to approximate the

triangulation error, and used a Jacobian matrix to calculate the covariance of 3D points. However, they neglected the higher order moments, which meant that these expressions did not hold exactly. Given quantisation error, the reconstructed 3D point from triangulation stands for a 3D region in the 3D space (See Figure 2.4), which is called a 3D cell with hexahedral shape. In 2015, Freundlich et al. [55] made an assumption that the points were evenly distributed in the 3D space. By decomposing the 3D cell into 12 tetrahedra, they managed to deduce the covariance matrix to estimate the uncertainty of a reconstructed point, by computing the second moments of a uniform distribution in the corresponding 3D cell.

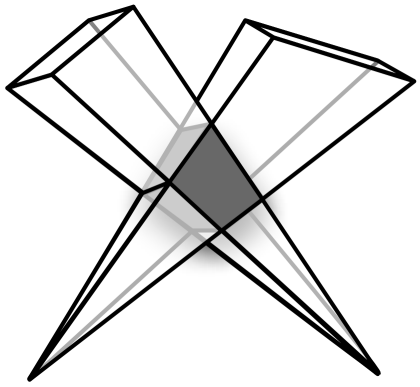


Figure 2.4: Illustration of a 3D cell from [55]. The grey 3D region is called a 3D cell.

Some researchers started to investigate an empirical approach, which also achieved a good result. For example, in 2009, Wedel et al. [151] used semi-global matching (SGM) to estimate disparity for stereo vision, and showed that the smaller the cost ratio (the current estimate cost:neighbouring cost), the more reliable the disparity estimate. They mapped the inverse of the surface slope to the variance of that disparity (See Figure 2.5). However, this did not work well in periodic pattern areas.

In 2015, Dal Mutto et al. [38] tried to derive the uncertainty for SGM from the similarity between the left centre pixel and its neighbours, between right corresponding centre pixels and their neighbours, and between the corresponding pair. They used segmentation to avoid errors in the discontinuity areas. However, the research lacked experiments dealing with correspondences in periodic patterns and textureless areas. In 2016, Marin et al. [94] utilised a more complex empirical equation that depended on both the relationship between local and global costs, and the properties of the local cost function. They managed to represent the uncertainty of their stereo results, which was used to eliminate error matches on periodic patterns and in textureless areas. More recently, Agresti et al. [14] created a synthetic dataset called ‘SYNTH3’ for

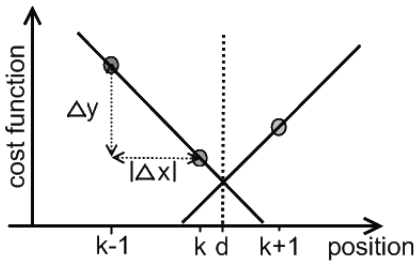


Figure 2.5: The slope as uncertainty measurement from [151]. The Y-axis represents the accumulated cost for a certain pixel, and the X- axis represents the disparity hypothesis. Term k is an integer, and stands for the disparity estimation, whose accumulated cost is the global minimum. Term d represents the disparity estimate in subpixels, which will be computed by a subsequent fit of a symmetric equiangular function in the cost volume. If the absolute slope ($\left|\frac{\Delta y}{\Delta x}\right|$) is greater, the uncertainty will be lower, and the reverse is also true.

ToF-stereo depth fusion, and used a simple convolution neural network to learn the uncertainty of ToF and the stereo vision disparity map supervisedly. In the latest work, LAF-Net [83] proposed a method to estimate the confidence of an initial disparity map from a stereo vision algorithm. They inputted three modalities (color image, initial disparity, matching cost) into their proposed network called LAF-Net. The LAF-Net consisted of four sub networks: feature extraction network for feature extraction, attention inference network for the joint usage of the tri-modal confidence features, scale inference network for extracting the confidence features in optimal receptive fields, recursive confidence refinement network for the confidence refinement. Their experiments showed it achieved the state-of-art performance for confidence estimation of disparity maps from stereo vision algorithms on the Middlebury Stereo Datasets [130] and the Kitti2015 dataset [98].

Compared to the depth uncertainty from a stereo vision algorithm, the confidence map for depth from the ToF sensor is easier to estimate. The principle of the ToF sensor can be referred to in [160]. The confidence measurement model of a ToF sensor can be built easily. For example, Marin et al. [94] used both the geometric and radiometric properties of the scene to estimate the confidence map for the ToF sensor. Regarding the metric from radiometric property, they considered the relationship between the intensity and amplitude of the received ToF signal. The other metric from the geometric property considered the local depth variance. They assumed that the two metrics were independent. The final confidence of the depth map from the ToF sensor was the product

of these two metrics. More confidence measurements for the ToF sensor can be referred to in [14, 30, 38, 94, 160]. Microsoft Kinect V2 is one kind of ToF sensor and is appealing for its low cost. Nguyen et al. [105] used the distance and angle between the Kinect sensor and observed surface to estimate both axial and lateral noise distributions. In the latest work, Giancola et al. [58] investigated uncertainty characterization of Kinect V2 by transmitted TOF signal evaluation, stability and distribution normality discussion, range measurement at pixel and sensor scales. They succeeded in giving the state-of-art uncertainty description based on the physical experiment testing.

In conclusion, the classical depth fusion methods have two issues limiting the accuracy of the refined disparity map: (1) Estimating the confidence map for each type of sensor accurately is difficult and renders the system unstable. (2) Accurately modelling the complex disparity relationship among neighbouring pixels in random scenes is challenging. However, the methods based on deep learning can avoid these problems (to some extent).

2.2.2 Methods Based on Deep Learning

The remaining methods are based on deep learning. Some researchers [14, 135] estimated the confidence maps for different sensors using deep learning methods, and incorporated the confidence as weights into the classical pipeline to refine the disparity map. However, these methods treated the confidence maps as an intermediate result, and they did not train the neural network to conduct the fusion from end to end directly. In 2016, Poggi and Mattoccia [114] first proposed an end-to-end neural network to conduct the disparity fusion. They selected the best disparity value for each pixel (from the several algorithms) by formulating depth fusion as a multi-labeling deep network classification problem. More specifically, given a set M of depth maps from m depth acquisition methods for the same scene at the same time, the depth fusion target is to obtain a more accurate depth map D_F by combining the initial input depth maps D_k, $k \in M$. As for the value of each pixel in the refined depth map D_F, it will be from the corresponding pixel in one of the initial depth maps D_k. This problem was framed as a multi-label classification problem. In order to get the value of one pixel in the refined map, they extracted one square patch centered on the corresponding pixel in each of the m initial depth maps respectively. The size of the square patch is N and they inputted this 3D patch with the size $N \times N \times m$ into their neural network and outputted the probability that each depth acquisition method has a correct depth estimation. Figure 2.6 shows

their network architecture.

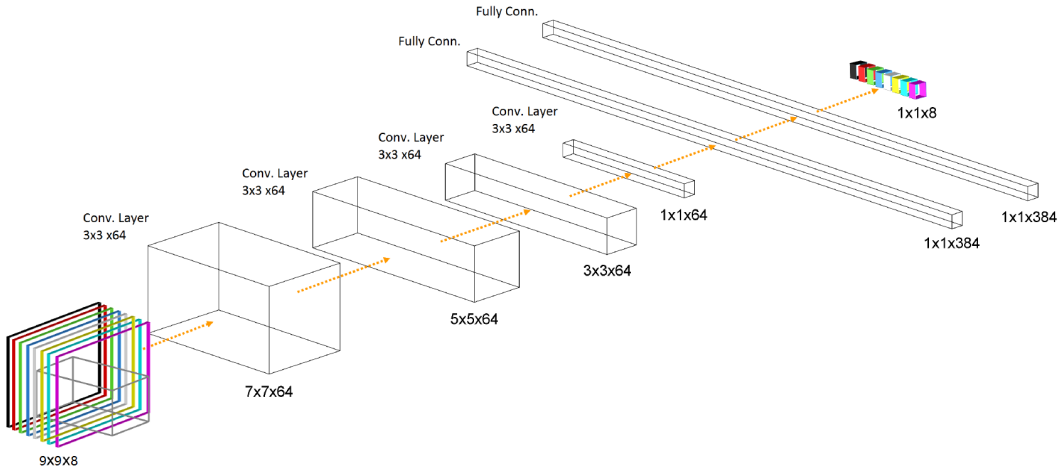


Figure 2.6: Network architecture of DSF [114]. In their paper, they used $m = 8$ depth acquisition methods to provide initial depth maps and the size of square patch is $N = 9$. There are four convolutional layers and in each layer there are $F = 64$ convolutional kernels with size 3×3 and the stride is 1. No padding was applied. Then two fully-connected layers appear, followed by ReLU (Rectifier Linear Unit) activators. In each fully-connected layer, there are 384 neurons. After the final fully-connected layer, the sigmoid function is added to get the output. The output vector is the probability of each depth acquisition method having the correct depth estimation.

Completely different from our methods in Chapter 3 and Chapter 4, each run of their neural network during inference could only assign one value to one pixel in the refined map. If they want to get the whole refined depth map with resolution $a \times b$, they need to run their neural network for $a \times b$ times. As for the proposed methods in this thesis, the whole initial depth inputs were input into the proposed network during inference and the proposed network only needs to be run once to output the whole refined map. As for the total inference time for one refined map on Kitti2015 dataset using GTX1080Ti, ours is about 0.01 second and theirs is about 10 seconds. As for the training loss function, they used the Binary Cross Entropy loss function (BCE) and Stochastic Gradient Descent (SGD) to optimize it. Equation 2.1 shows their loss function.

$$BCE(o,t) = -\frac{1}{n}\sum_{i\in B}\sum_{k\in M}\{t[i][k]log(o[i][k]) + (1-t[i][k])log(1-o[i][k])\}. \quad (2.1)$$

In Equation 2.1, o is the output of the neural network. t is the binary indicator on each sample i of the mini-Batch B. n is the normalization factor and the rest follow

the definitions in the former description in this part. By framing the depth fusion as a multi-label classification problem, they refined the initial depth maps to some extent. However, the method only used the disparity maps from the sensors, and neglected other associated image information (such as intensities and gradients). This approach did not exploit the real disparity relationship among adjacent pixels. Given this limitation, it could not achieve high accuracy for the final refined depth map. Chapter 3 took auxiliary information from other domains (such as intensity and gradient) into consideration, and incorporated this information into the energy function of the refiner network, making the refiner network generalise to a new scene easily (but with similar accuracy). More specifically, it adopted adversarial training by using two neural networks, a refiner, and a discriminator network. The refiner took the initial depth maps and auxiliary information as input, and outputted the refined depth map. The real depth map and refined depth map were input to the discriminator, to discriminate whether the input was real or fake. This forced the disparity relationship in the refined disparity map closer to the real case. It also proposed semi-supervised learning to reduce the requirement of ground truth data. Both of their supervised and semi-supervised methods could adapt to different kinds of depth fusion, such as monocular-stereo fusion, ToF-stereo fusion, and stereo-stereo fusion. It achieved the state-of-art performance in the general depth fusion area. Meanwhile, a later work [16] (in an area related to depth fusion) proposed to use a semi-supervised method to realize ToF depth denoising with adversarial networks. In their supervised part, they used a coarse-to-fine CNN (similar to a generator in a GAN) to produce the refined ToF depth map, which was trained on synthetic data with ground truth. Parallelly, an adversarial learning strategy was used to conduct an unsupervised domain adaptation to bridge the gap between synthetic and real data. The synthetic and real data were used simultaneously during adversarial training, enabling a better generalization into the real world.

Additionally, Lidar-stereo depth fusion [33, 92, 109, 111], has become increasingly popular in the past several years. In 2016, Maddern [92] proposed a probabilistic fusion approach by treating the Lidar points as support points, which worked robustly only in the area with Lidar information. More recently, Park et al. [109] proposed to do Lidar-stereo fusion using a supervised convolutional neural network. They input the Lidar information and depth from SGM [67] and used the Lidar information and the disparity map from MC-CNN [161] as the ground truth. By setting the weights of the Lidar term and stereo vision term in the energy function, they forced the output of their neural network get close to the Lidar point if the Lidar information is valid

and, otherwise, close to the MC-CNN. In the latest work [111], Park et al. [109] extended their previous work [109] by incorporating an online calibration network. The online calibration network aligned the input sensor coordinate systems by correcting the initial extrinsic parameters and formulated the calibration process in the depth domain. Then the depth fusion network in [109] was used to conduct Lidar-stereo depth fusion. All these methods above based on deep learning are supervised, and they need ground truth depth data to train (more or less). However, ground truth data is difficult to acquire, and expensive. In the latest work, Cheng et al. [33] proposed a first unsupervised Lidar-stereo fusion network, which could be trained without ground truth depth data. They exploited depth consistency between Lidar and stereo vision, photometric consistency between stereo images to build a training loss to realize unsupervised training. Meanwhile, Chapter 4 also presents the development of a fully unsupervised method for the disparity fusion problem with adversarial netowrks for the first time, by building a mathematical model based on multi-modal information to train the network.

Besides the methods which fuse multiple depth maps from different sources, some different (but similar) work has emerged recently, such as depth completion [48, 75, 122, 136, 157]. In [48, 75, 122, 136, 157], a sparse depth map and an RGB image were both input into a neural network. By using the auxiliary information from the RGB images, they obtained a dense and accurate depth map based on the initial sparse depth map. The neural network completed the missing depth data in the initial sparse depth map. However, all these methods are currently supervised, and obtaining the fully dense ground truth in real environments is difficult and expensive.

In conclusion, using the complementary advantage of each depth acquisition method can achieve a highly accurate and reliable depth map. However, there is a trade-off between performance and cost. The performance refers to accuracy and robustness, and the cost refers to more computation resources and additional time complexity. Currently, the methods based on deep learning have become the main trend because of their universality and performance. Both of the proposed depth fusion methods in Chapter 3 and Chapter 4 are developed based on deep learning.

2.3 Rigid Point Cloud Registration

After obtaining a refined depth map with the depth fusion algorithm, the depth map can be projected into 3D space using the intrinsic camera parameters to form a 3D point

cloud in the current view. To obtain the relative sensor pose between different views, rigid point cloud registration is a method of realising that aim, which is the second research target in this thesis. Regarding the rigid point cloud registration algorithms, there are currently three main categories: registration based on ICP (Iterative Closest Point) [20, 23, 32, 116, 134, 145, 146, 149, 156, 159, 168, 169], registration based on feature matching [26, 42, 47, 86, 158, 162, 164], and registration based on probability alignment [27, 28, 35, 45, 74, 76, 89, 100, 103, 138, 142, 167].

2.3.1 Registration Based on ICP

In 1992, Besl and McKay [23] first introduced the iterative closest point (ICP) algorithm, to compute the rigid transformation between two point clouds by minimising the Euclidean distance between the corresponding points. Its pipeline is listed in Algorithm 1. X is the static point cloud and Y is the moving point cloud. ICP updates the transformation matrix in an iterative style by using an EM optimization algorithm [40]. Step 3 finds the point x_k closest to the predicted position $\mathbf{R}y_j + \mathbf{t}$. Step 4 updates the $\mathbf{R}$ and $\mathbf{t}$ given the correspondence.

Algorithm 1 Standard ICP Algorithm [23].

Input: Two point clouds $X = \{x_i\}$, $i = 1..M$; $Y = \{y_j\}$, $j = 1..N$;
initial transformation $\mathbf{R} = \mathbf{I}$, $\mathbf{t} = \mathbf{0}$;

Output: $\mathbf{R}$ and $\mathbf{t}$, which aligns the moving point cloud Y to the static point cloud X;

1: **procedure** (Repeat until convergence:)
2: **procedure** (Iteration from j=1..N)
3: $x_k \leftarrow \mathbf{R}y_j + \mathbf{t}$ ▷ *Correspondence matching*
4: $(\mathbf{R}, \mathbf{t}) \leftarrow \arg\min_{\mathbf{R},\mathbf{t}} \mathcal{L} = \sum_{j=1}^{N} (\mathbf{R}y_j + \mathbf{t} - x_k)^2$

Since then, a large number of variants have been proposed, and the reader is directed to the survey in [116]. To enhance robustness against noise, Segal et al. proposed the Generalized-ICP [134] in 2009, which considered the probability distribution of each point. They used a 'plane-to-plane' strategy (unlike the point-to-plane ICP [32]) by using a probabilistic framework to describe the planar surface structure in both point clouds locally. However, the correspondence search was still binary, like a standard ICP [23], which limited its performance. To be robust to occlusion and small partial overlap, researchers [169] built bilateral correspondence using bidirectional distances. Later, they [168] extended their previous work [169] by incorporating the hard and

soft assignments to improve registration accuracy further. To widen the convergence zone, GO-ICP [156] used a branch-and-bound method to avoid getting stuck in the local minimum. More recently, Vongkulbhisal et al. [145] used extended discriminative optimization to learn a sequence of update steps instead, which searched the global parameter space to avoid the local minima. Given the semantic information that could be used as a prior when searching for correspondences between the two point clouds to improve the final registration accuracy, Zaganidis et al. [159] extended the work in Generalized-ICP [134] further by incorporating semantic information of the point clouds, which were from the PointNet [121]. Later, Aoki et al. [20] adapted the point cloud representation from PointNet [121] into the Lucas & Kanade (LK) algorithm [91] by using a single trainable recurrent deep neural network to realize 3D point cloud registration. Wang et al. [146] adapted the idea of ICP into their pose prediction neural network to obtain the 6D pose of the objects in RGB-D images. More specifically, when they trained their pose prediction neural network, they used the minimum distance as the cost function between the corresponding points in the ground truth 3D model (similar to the static point cloud in ICP) and estimated 3D model (similar to the moving point cloud in ICP). The difference here is that they used an end-to-end neural network to obtain the 6D pose directly, rather than by using an iterative style. In the latest work, DCP [149] revisited the standard ICP [23] from a deep learning perspective. First they embedded the input point clouds as rigid-invariant/permutation representations to help identify point correspondence. An attention module was used to estimate a soft matching between the two input point clouds. The final rigid transformation was calculated by a differentiable singular value decomposition layer. This algorithm achieved the state-of-art performance in the rigid point cloud registration area based on ICP.

Exact point-to-point correspondences seldom exist, and the correspondence definition (two points have the minimum distance) is coarse, which makes it difficult for the classical ICP family to achieve accurate results compared with other types of method. However, the methods based on ICP usually have lower time complexity.

2.3.2 Registration Based on Feature Matching

The second class of alignment approaches is feature-based methods, which first extract and match local descriptors (e.g. FPFH [127], SHOT [139]) from two point clouds and estimate the relative pose using random sampling such as [127], RANSAC [53], and

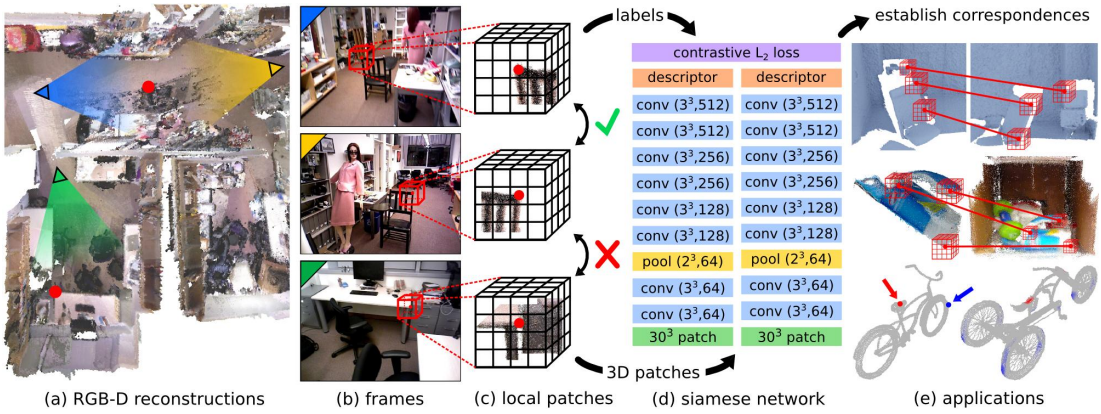


Figure 2.7: 3DMatch Overview [162]. (a) The RGB-D reconstruction algorithm was used to build the global 3D model with an RGB-D sensor. (b) From different views, they extracted local RGB-D patches and their correspondence labels. (c) They converted the local 3D patches into a volumetric representation and used the non-matching and matching pairs as the training dataset. (d) The non-matching and matching pairs were input into a siamese network for training. (e) After training, the geometric descriptor can be used in different applications, such as 3d reconstruction, model alignment, and surface alignment. The graph is from its project website [7].

Hough transforms [153]. Recently, Zeng et al. [162] used a siamese neural network to learn a local patch descriptor in a volumetric grid, to establish the correspondences between two point clouds. Figure 2.7 shows the overview of 3DMatch [162]. Similarly, Elbaz et al. [47] used a deep neural network auto-encoder to encode local 3D geometric structures instead of traditional descriptors. Lei et al. [86] proposed a fast descriptor based on eigenvalues and normals computed from multiple scales to extract the local structure of the point clouds, and then recovered the transformation from matches. The registration estimation from 3D keypoint correspondences is significantly affected by false correspondences and outliers. Bustos and Chin [26] developed a new preprocessing method called guaranteed outlier removal (based on geometric operations), which rejected false correspondence in the globally optimal solution and reduced the input point cloud to a smaller size. Given the population decline of the outliers, the optimisation could converge faster, but with the same high accuracy. More recently, researchers [42, 158, 164] paid more attention to 3D feature extraction based on deep learning and recovered the 6D pose based on 3D feature matching. Yew and Lee [158] proposed 3DFeat-Net to learn 3D feature descriptor and detector for rigid point cloud registration. Similar to 3DMatch [162], they also used a siamese architecture in their

proposed algorithm. They designed a triplet loss considering the 3D point saliency and individual descriptor similarities to train the network. The output of 3DFeat-Net is a set of local descriptor vectors. Meanwhile, Zhang et al. [164] used two 3D convolutional neural networks to extract 3D deep features. In order to extract 3D deep features for key points, they set corresponding and non-corresponding points to train their neural network, by minimizing the feature distance between corresponding pairs and maximizing the feature distance between non-corresponding pairs. In the latest work, Deng et al. [42] thought that a good feature for 3D point cloud matching should include not only the information for correspondence establishment but also a direct rigid transformation estimation. They proposed an end-to-end algorithm for 3D local feature extraction. They started by augmenting PPF-FoldNet [41] with a learned orientation. Then they decoupled the 3D structure from 6D pose via their pose-variant orientation learning. Finally a hypothesize-and-verify scheme was proposed to find the optimal alignment. This algorithm achieved the state-of-art performance in this area.

Although researchers have been developing this type of method, intrinsic disadvantages in this class have limited its extremely high accuracy. For example, these methods are sensitive to noisy point clouds. Additionally, the density of the point cloud influences the extraction of local descriptors, and even makes them completely break down if the density is too low. Compared with this type of method, rigid point cloud registration based on probability alignment can be superior.

2.3.3 Registration Based on Probability Alignment

Aligning probabilistic models that represent the structure of the point cloud can effectively mitigate the problems from the other two previously mentioned classes. One key factor for an accurate and robust registration is the data representation used. Since 2003, many approaches based on a variety of probabilistic models have been explored to represent the structure of the point cloud, such as Robust Point Matching [35], Kernel Correlation [142], and Coherent Point Drift [103]. In the coherent point drift algorithm, Myronenko and Song [103] regarded one point cloud as a GMM and the other point cloud as the given data points. They solved the rigid registration problem by maximizing the GMM posterior probability for the given data points. More specifically, the fixed point cloud is defined as $X_{D\times N} = \{x_1, x_2, ..., x_N\}$ and the moving point cloud $Y_{D\times M} = \{y_1, y_2, ..., y_M\}$ (D is the dimension of the point cloud, N and M are the number of the points in point cloud X, Y respectively). The points in point cloud Y are consid-

ered as the GMM centroids and the points in point cloud X as the data points generated from the GMM. The corresponding probability density function of the GMM is:

$$p(x) = \sum_{m=1}^{M+1} P(m)p(x|m) \tag{2.2}$$

In Equation 2.2, $p(x|m) = \frac{1}{(2\pi\sigma^2)^{D/2}} exp^{-\frac{||x-y_m||^2}{2\sigma^2}}$. Given the noise and outliers, an extra uniform distribution $p(x|M+1) = \frac{1}{N}$ is added and its weight is assigned as w, $0 \leq w \leq 1$. All the covariances σ^2 for all the points in point cloud Y are assumed as isotropic and equal. The membership probability of each point in point cloud Y is assumed as equal $P(m) = \frac{1}{M}$. Equation 2.2 could be rewritten as:

$$p(x) = w\frac{1}{N} + (1-w)\sum_{m=1}^{M} \frac{1}{M} p(x|m) \tag{2.3}$$

They assumed that each data point is independent and identically distributed. The rigid point cloud registration problem can be cast into maximum likelihood estimation $L(R,t,\sigma^2) = \prod_{i=1}^{N} p(x_i)$, R is the rotation and t is the translation. It is equal to minimizing the $E(R,t,\sigma^2) = -log(L(R,t,\sigma^2))$, which can rewritten as:

$$E(R,t,\sigma^2) = -\sum_{n=1}^{N} log\{w\frac{1}{N} + (1-w)\sum_{m=1}^{M} \frac{1}{M} p(x|m)\} \tag{2.4}$$

Given that it is rigid point cloud registration, the underlying constraint is $R^T R = 1$, $det(R) = 1$. Then they used EM algorithm [40] to optimize the energy function in Equation 2.4 to get the optimal rotation and translation.

In 2011, GMMREG [76] used two GMM (Gaussian Mixture Model) with the same isotropic covariances for each point, and minimised the L_2 distance between the two GMM to obtain the transformation. The cost function is:

$$E_{L_2}(R,t) = \int \{GMM(X) - GMM(T(Y,R,t))\}^2 dx \tag{2.5}$$

In Equation 2.5, $GMM(*)$ represents the Gaussian mixture density built from point cloud $*$. $T(Y,R,t)$ represents transforming point cloud Y with rotation R and translation t. A non-stochastic gradient-free optimization method, for example, Powell's method, was used to optimize the cost function to obtain the transformation parameters in the rigid point cloud registration. However, the GMM that represented the fixed point cloud was regarded as invariant in this algorithm; hence, it could not receive the current registration state from the other point cloud.

In 2014, Zhou et al. [167] proposed to use the Student's-t mixture model to represent the point cloud in the registration algorithm. In 2015, Campbell et al. [27] used a support vector machine (SVM) to learn and construct SVGM (a GMM with non-uniform weights) to represent the point cloud. In the following year, Campbell et al. [28] used SVGM [27] and the architecture in GMMREG [76] to obtain the globally-optimal transformation using a branch and bound approach in order not to be vulnerable to local minimum. Recently, Straub et al. [138] used a Dirichlet process Gaussian mixture (DP-GMM) and a Dirichlet process von Mises-Fisher mixture (DP-vMF-MM) to represent the geometric information of the point cloud. The mathematical probabilistic model used to represent the point cloud has become considerably complex. Pu et al. [118] used the physical 3D uncertainty distribution from real sensors to construct a simple uncertainty-based GMM to represent the structure of the point cloud, which fitted the real surface geometry better. They convolved two uncertainty-based GMM in the whole 3D space to build the energy function, and used the EM algorithm [40] to optimise the objective function. Lawin et al. [74] modeled the underlying structure of the scene as a latent probability distribution, and used the EM algorithm [40] to infer the registration parameters by minimising the Kullback-Leibler divergence. Given that they used the underlying structure of the scene, their algorithm could handle point clouds with different densities robustly. However, algorithms based on the EM algorithm [40] are local algorithms, which may not obtain the global optimal solution. To avoid local minimum, most existing global registration algorithms ([28, 156]) use branch and bound optimisation, which usually suffers a high time complexity. To speed up the branch and bound algorithm, Liu et al. [89] proposed to decouple the optimisation of rotation and translation using rotation invariant features. That is, they would optimise the translation parameters first. Subsequently, based on the estimated global optimal translation, the global rotation would be calculated later. Their experiments showed that both speed and accuracy were improved compared with the counterparts. More recently, Eckart et al. [45] used a Hierarchical Gaussian Mixture to build a top-down multi-scale representation of the point cloud. They used the constructed representation through a novel optimization criterion to conduct data association between spatial subsets of the point clouds. The optimization criterion was based on principal component analysis and approximated the true maximum likelihood estimation well. The EM algorithm [40] was adopted to search for the optimal solution finally. This algorithm achieved the state-of-art performance. In the latest work, Min and Meng [100] used Gaussian mixture models and Fisher distribution mixture models to represent the observed position set

and normal vector set of the point cloud respectively. Additionally, they regarded the hybrid mixture models as inhomogeneous. They also used an EM algorithm [40] as the optimization algorithm in their proposed algorithm. However, their experiments failed to show significant registration improvements.

However, compared with the other two classes of rigid point cloud registration, the methods in this class usually require more running time to finish the registration but could achieve a higher accuracy against strong noise, density change, numerous outliers and large occlusion.

2.4 Generative Adversarial Network

Since the adversarial networks (similar to GAN) are used in Chapter 3 and Chapter 4, GAN progress is introduced here briefly although GAN theory is not the research target in this thesis. The GAN was first proposed by Goodfellow et al. [61], who trained two neural networks (generator and discriminator) simultaneously, to make the distribution of the output from the generator approximate the real data distribution by a minimax two-player strategy. Instead of explicitly calculating the probability of each sample x in the real data distribution P_{data}, a GAN uses the generator G to generate samples from the generator distribution P_G by mapping a random noise variable $z \sim P_{noise}$ to a generated sample $G(z)$. The sample x from the real data distribution and the generated sample $G(z)$ from the generator distribution P_G are fed into the discriminator D to discriminate whether the input is true (from the real data distribution) or fake (from the generator distribution). The minimax two-player strategy could be cast into the following expression (Equation 2.6):

$$\min_G \max_D V(D,G) = \mathop{\mathbb{E}}_{x\sim P_{data}} \Big[\log(D(x))\Big] + \mathop{\mathbb{E}}_{z\sim P_{noise}} \Big[\log(1-D(G(z)))\Big] \tag{2.6}$$

Later, many variants based on the original GAN model [61] were developed. The readers could be directed to the latest surveys [36, 108, 150]. In the following, the architecture-variant GAN, loss-variant GAN and GAN application on depth domain will be introduced briefly.

2.4.1 Architecture-variant GAN

Given the original GAN [61] adopted fully-connected neural networks in the discriminator and generator, its architecture could only applied to some simple images (eg:

images in MNIST [85]). In order to obtain a high-resolution image from low resolution, LAP-GAN [43] incorporated a cascade of CNNs into a Laplacian pyramid framework to achieve high-resolution outputs. Meanwhile, DCGAN [124] proposed to use a deconvolutional neural network architecture for the generator first to output the high resolution images. BEGAN [165] viewed the discriminator as an energy function, which allowed to use various loss functions and architectures including the binary classifier with logistic output. They used an auto-encoder architecture for the discriminator for the first time. In order to capture information in a large receptive field but not to sacrifice computational efficiency, SAGAN [144] incorporated a self-attention mechanism into the generator and discriminator architectures, which could learn global, long-range dependencies when generating images. PROGAN [79] proposed to grow both the generator and discriminator progressively when training progressively. The training started from a low resolution input and the network architecture added new layers progressively, which made the GAN model fine details increasingly. In the latest work, BigGAN [25] extended the architecture in SAGAN [144], by introducing two general architectural changes to improve scalability. A orthogonal regularization to the generator was used to improve conditioning. It achieved the state-of-art performance about the fidelity and variety of the generated samples.

2.4.2 Loss-variant GAN

Compared with JS divergence in the original GAN [61], the Wasserstein distance is linear and could better represent distance even when the real and generated model distributions do not overlap. Some researchers [21, 62] used the Wasserstein distance to measure the distance between the model distribution and the real distribution, which reduced the difficulty of training the GAN significantly. It also reduced mode collapse (the phenomenon where the generator learns only a partial distribution rather than the full real data distribution) to some extent. Meanwhile, Metz et al. [99] tried to solve mode collapse and unstable optimization in a GAN through unrolling optimization of the discriminator objective during training. Mao et al. [93] found that the sigmoid cross entropy function could lead to the vanishing gradients dilemma when training the adversarial networks. They proposed to use the least squares loss function for the discriminator to solve that problem, which yielded minimizing the Person χ^2 divergence. Recently, Miyato et al. [101] proposed a weight normalization method called spectral

normalization, which could stabilize the discriminator training and is computationally light and easy to incorporate. In the latest work, Jolicoeur-Martineau [77] proposed to use 'relativistic discriminator' in GAN rather than the standard discriminators (eg: discriminators in [21, 61, 62, 93, 99]) which output the probability that the input data is real. The 'relativistic discriminator' should estimate that probability that the real data is more realistic than a randomly generated fake data. When using the 'relativistic discriminator', it could achieve more stable training and generate better samples than the counterparts with the standard discriminators.

2.4.3 GAN Application on Depth Domain

The GAN family of methods have been applied to many problems in different areas and the following introduction will concentrate on the problems related to the depth domain [16, 18, 19, 37, 63, 73, 113, 120], which is close to the research target 'depth fusion' in this thesis. Isola et al. [73] trained a GAN to translate between image domains, which can be also used to transfer the initial disparity maps from several sensors into a refined disparity map. However, the design proposed in [73] neglects information useful for disparity fusion, which limits the accuracy of the refined disparity map. More recently, Kumar et al. [37] used adversarial networks to do depth estimation and pose prediction from input monocular video sequences. In the generator, they used two sub networks, one for depth estimation of the target image, one for pose estimation between the target image and another view. Then the RGB information in another view could be projected into the view of the target image using the estimated depth and pose to form a fake RGB image. Then the fake reconstructed image and real image were fed into the discriminator to make the discriminator judge whether the input was real or fake. Lore et al. [63] used two cascaded GAN for the supervised depth estimation. The generated depth map from the generator and the ground truth were fed into the discriminator to judge. More recently, some unsupervised work in depth estimation has emerged. Pilzer et al. [113] used the similar architecture in CycleGAN [170] for stereo vision. They inputted a pair of stereo images and used two GAN to estimate the depth on the left and right view. After the depth estimation, they reconstructed the left and right image respectively. Finally, they fed the fake images and corresponding real images on the left or right view into the two discriminators to judge. The used the intensity difference between the reconstructed image and real image as the training loss,

which avoided the requirement of the ground truth depth data and realized unsupervised training. The similar unsupervised training strategy for depth estimation could be found in more work [18, 19, 120]. However, there are no work based on adversarial networks for depth fusion currently. Chapter 3 and Chapter 4 will investigate depth fusion based on adversarial networks.

2.5 Conclusion

In this chapter, the related literature and background have been reviewed. First, different kinds of depth acquisition methods and their corresponding principles were briefly reviewed, which is not the research target in this thesis but will provide input depth data for the proposed methods in Chapter 3 and Chapter 4. Then the history and current progress of depth fusion was investigated, which is the first research target. Subsequently, different types of rigid point cloud registration algorithms were introduced, which is the second research target. Finally, a short introduction about the progress of generative adversarial networks was presented, which is not the research target in this thesis but will provide some theoretical support for the proposed depth fusion part. In the next chapter, a supervised and semi-supervised depth fusion method will be developed by fusing the initial depth maps from the existing depth acquisition methods.

Chapter 3

Supervised & Semi-supervised Depth Fusion from Multiple Sources

Refining raw disparity maps from different algorithms to exploit their complementary advantages remains challenging. Uncertainty estimation and complex disparity relationships among pixels limit the accuracy and robustness of existing methods, and there is no standard method for fusion of different kinds of depth data. In this chapter, a new method to fuse disparity maps from different sources is introduced, while incorporating supplementary information (such as intensity, and gradient images) into a refiner network to refine raw disparity inputs more successfully. A discriminator network classifies disparities from different receptive fields and scales. Further, assuming a Markov Random Field[1] for the refined disparity map produces better estimates of the true disparity distribution. Both fully supervised and semi-supervised versions of the algorithm are proposed. The approach includes a more robust loss function to inpaint invalid disparity values, and requires much less labeled data to train in the semi-supervised learning mode. The algorithm can be generalised to fuse depths from different kinds of depth sources. The experiments explored different fusion opportunities: stereo-monocular fusion, stereo-ToF fusion, and stereo-stereo fusion. The experiments show the superiority of the proposed algorithm compared with the most recent algorithms on public synthetic datasets (Scene Flow, SYNTH3, and our synthetic garden dataset) and real datasets (Kitti2015 dataset and Trimbot2020 Garden dataset). The results presented in this chapter were achieved during October 2018. The majority of the content in this chapter is from the paper [119].

[1] 'Markov random field' description in Wikipedia [5]: a set of random variables having a Markov property (the memoryless property of a stochastic process) described by an undirected graph.

3.1 Introduction

With recent improvements in depth sensing devices, depth information is now easily accessible. In a stereo camera pair depth and disparity are interchangeable measures:

$$depth = \frac{focal\_length \times baseline}{disparity}. \tag{3.1}$$

In Equation 3.1, $focal\_length$ and $baseline$ are the focal length and baseline parameters of the stereo camera setting. When data is from a sensor like a time of flight sensor, the depths can be converted into disparities using a constant focal length and baseline from a stereo camera setting. However, each sensor has its own advantages and disadvantages, with the result that no algorithm can perform accurately and robustly in all general scenes. For example, active illumination devices such as ToF (Time of Flight) sensors and structured light cameras [160] estimate the depth information accurately regardless of the scene content but struggle on low reflective surfaces or outdoors. Stereo vision algorithms [29, 67, 68, 69, 97] work better outdoors and perform accurately on high texture areas but behave poorly in repetitive or textureless regions. Monocular depth estimation algorithms based on deep learning (eg: [59]) work robustly in textureless areas but tend to produce blurry depth edges if they use a smoothing term in the objective function. Fusing multiple depth maps from different kinds of algorithms or devices and utilizing their complementary strengths to get more accurate depth information is a valuable technique for various applications.

The traditional pipeline for the majority of the fusion algorithms [14, 38, 52, 94, 104] is: (1) estimate disparities from the different sensors, (2) estimate associated confidence maps, and (3) apply a specific fusion algorithm based on the confidence maps to get a refined disparity map. This approach has three potential problems. Primarily, estimating the confidence maps for different sensors is a hard task with limited robustness and accuracy. Second, estimating the disparity relationship among pixels in a general scene is hard without prior knowledge. Finally, there is no common methodology for different kinds of depth fusion, such as stereo-stereo fusion, monocular-stereo fusion and stereo-ToF fusion. Researchers have designed different methods for different fusion tasks. The recent fusion method [114] based on end to end deep learning has provided a general solution to different kinds of fusion but has limited accuracy and robustness, in part due to not exploiting other associated information to help the network make judgments. It also did not exploit the disparity relationship among pixels.

In this chapter, an architecture similar to a Generative Adversarial Network (GAN) [61]

is proposed (generator is replaced by a refiner network without random noise input) to solve the problems listed above, by designing an effective network structure and a robust object function. In addition to the raw disparity maps the network input also includes other image information, i.e., the original intensity and gradient images (see Figure 3.1), in order to facilitate the selection of a more accurate disparity value from the input disparity images. This avoids having to design a manual confidence measure for different sensors and allows a common methodology for different kinds of sensor. To preserve and exploit the local information better, some successful ideas about local structure from Unet [126] and Densenet [70] have been used. To help the network refine the disparity maps accurately and robustly a novel objective function was designed. Gradient information is incorporated as a weight into the L_1 distance to force the disparity values at the edges to get closer to the ground truth. A smoothness term helps the network propagate the accurate disparity values at edges to adjacent areas, which inpaints regions with invalid disparity values. The Wasserstein distance [21, 62] replaced the Jensen-Shannon divergence [61] for GAN loss to reduce training difficulties and avoid mode collapse. With the discriminator network classifying input samples in different receptive fields and scales, the disparity Markov Random Field in the refined disparity map gives a better estimate of the real distribution.

We also propose a semi-supervised approach which trains the discriminator network to produce the refined disparity map using not only the labeled data but also unlabeled data along with the ground truth of the labeled data. It requires less labeled training data but still achieves accuracy similar to the proposed fully-supervised method or better performance when using the same amount of labeled data with additional unlabeled data compared with the supervised method, as shown in the experimental results.

Section 3.2 presents the new supervised and semi-supervised fusion models including the objective function and network structure. Section 3.3 presents the results of experiments conducted with synthetic and real data (Table 3.1) for stereo-monocular fusion, stereo-ToF fusion and stereo-stereo fusion.

The main contributions presented in this chapter are (Section 3.3 gives a more detailed comparison with [14, 29, 59, 67, 68, 69, 73, 94, 97, 114]):

- Improved fusion accuracy by using a network that learns the disparity relationships among pixels without any prior knowledge.
- Reduced the labeled data requirement drastically by using the proposed semi-supervised strategy.

- Increased robustness by fusing intensity and gradient information as well as depth data.
- Proposed a common network methodology allowing different kinds of sensor fusion without requiring detailed knowledge of the performance of each sensor.

3.2 Methodology

The proposed general framework (Figure 3.1) for disparity fusion and the new loss functions in the supervised and semi-supervised methods are introduced. These functions will make adversarial training simple and the refined disparity more accurate and robust. Finally, the end-to-end refiner (Figure 3.2) and discriminator (Figure 3.3) network structure are presented.

3.2.1 Framework

A method is developed that uses an adversarial network, which is similar to a GAN [61] but with raw disparity maps, gradient and intensity information as inputs instead of random noise. The refiner network R (similar to the generator G in a traditional GAN [61]) is trained to produce a refined disparity which cannot be distinguished (i.e. classified) as a "fake" by the discriminator D. Simultaneously, the discriminator D is trained to become better at distinguishing that the input from refiner R is fake and the input from the ground truth is real. By adopting a minimax two-player game strategy, the two neural networks $\{R, D\}$ make the output distribution from the refiner network approximate the real data distribution. The full system diagram is shown in Figure 3.1. A refiner network R is trained to map raw disparity maps (*disp1*, *disp2*) from two input algorithms to the ground truth based on associated image information (gradient, intensity). The refiner R tries to predict a refined disparity map close to the ground truth. The discriminator D tries to discriminate whether its input is fake (refined disparity from R) or real (real disparity from the ground truth). The refiner and discriminator can see both the supplementary information and initial disparity inputs simultaneously. Any number of disparity inputs or different information cues can be fused by concatenating them together directly as inputs. The two networks are updated alternately.

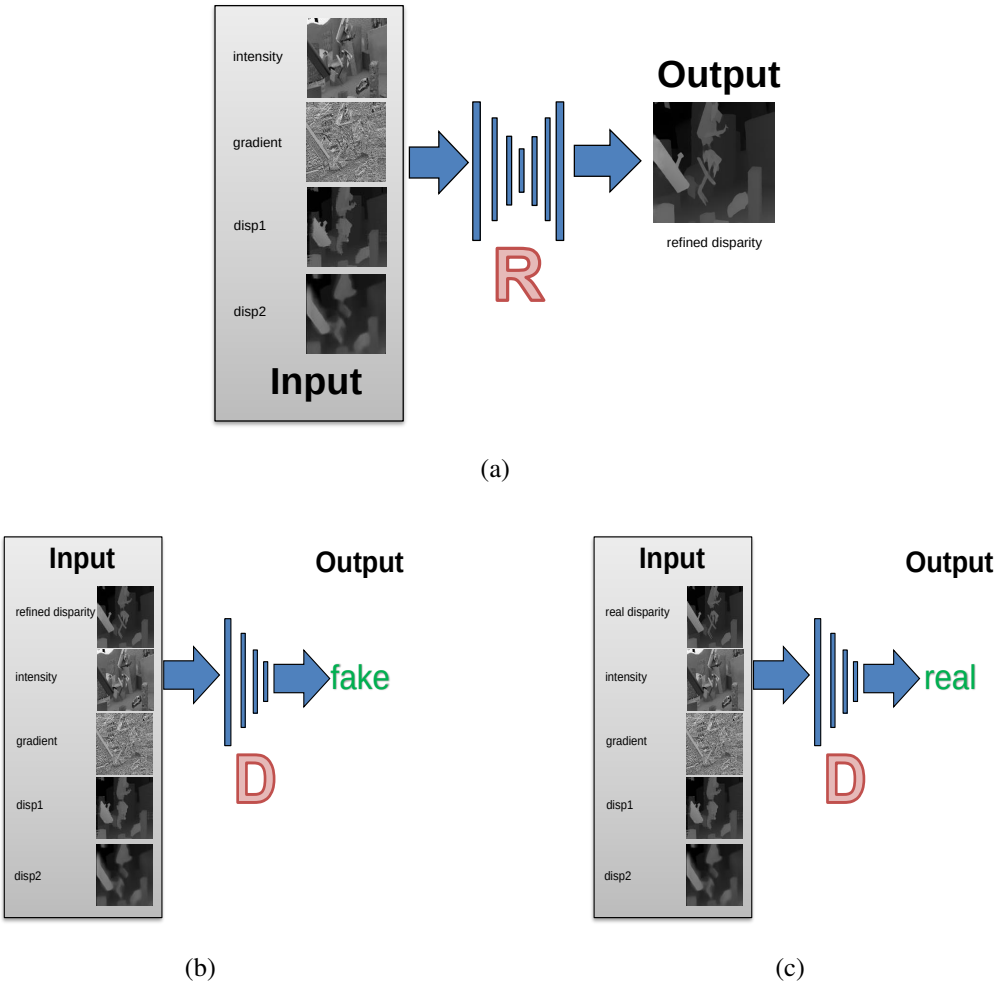


Figure 3.1: Overview of Sdf-man (the proposed method in this chapter). (**a**) Refiner: a network to refine initial disparity maps; (**b**) Negative examples: a discriminator network with refined disparity inputs; (**c**) Positive examples: a discriminator network with real disparity inputs.

3.2.2 Objective Function

To let the refiner produce a more accurate refined disparity map, the objective function is designed as follows:

(1) To encourage the disparity value of each pixel to approximate the ground truth and to avoid blur at scene edges (such as occurs with the Monodepth method [59]), a gradient-based L_1 distance training loss is used, which applies a larger weight to the disparity values at the scene edges:

$$\mathcal{L}_{L_1}(R) = \mathop{\mathbb{E}}_{x\sim P_{real},\tilde{x}\sim P_{refiner}} \left[\exp(\alpha|\nabla(I_l)|) \; ||x-\tilde{x}||_1 \right] \tag{3.2}$$

where R represents the refiner network. x is the ground truth and $\tilde{x}$ is the refined disparity

map from the refiner. P_{real} and $P_{refiner}$ represents the real disparity distribution from the ground truth and fake disparity distribution produced by the refiner. $\nabla(I_l)$ is the gradient (from Sobel operator) of the left intensity image in the scene because all the inputs and refined disparity map are from the left view. $\alpha \geq 0$ weights the gradient. $||\bullet||_1$ is the L_1 distance. The goal is to encourage disparity estimates near image edges (larger gradients) to get closer to the ground truth.

(2) A gradient-based smoothness term is added to propagate more reliable disparity values from image edges to the other areas in the image under the assumption that the disparity of neighboring pixels should be similar if their intensities are similar:

$$\mathcal{L}_{sm}(R) = \mathop{\mathbb{E}}_{u\in\tilde{x},v\in N(u),\tilde{x}\sim P_{refiner}} \Big[\exp(1-\beta|\nabla(I_l)_{uv}|)\ ||\tilde{x}_u - \tilde{x}_v||_1\Big] \tag{3.3}$$

where $\tilde{x}_u$ is the disparity value of a pixel u in the refined disparity map $\tilde{x}$ from the refiner. $\tilde{x}_v$ is the disparity value of a pixel v in the neighborhood $N(u)$ of pixel u. $\nabla(I_l)_{uv}$ is the gradient from pixel u to v in the left intensity image (the refined disparity map is produced on the left view). $\beta \geq 0$ is responsible for how close the disparities are if the intensities in the neighborhood are similar.

(3) The underlying assumption in $\mathcal{L}_{L_1}(R)$ is that the disparity relationship among pixels is independent. The disparity relationship in $\mathcal{L}_{sm}(R)$ is too simple to describe the real disparity relationship among neighbors in the real situation. To help the refiner produce a disparity map whose disparity Markov Random Field is closer to the real distribution, the proposed method inputs disparity maps from the refiner and the ground truth into the discriminator, which outputs the probability of the input samples being from the same distribution as the ground truth. This probability is used to update the refiner through its loss function. Instead of defining a global discriminator to classify the whole disparity map, it is defined to classify all local disparity patches separately because any local disparity patch sampled from the refined disparity map should have similar statistics to the real disparity patch. By making the discriminator output the probabilities in different receptive fields or scales (In Figure 3.3, please refer to $D1$, $D2$, ..., $D5$), the refiner will be forced to make the disparity distribution in the refined disparity map close to the real. In Equations (3.4) and (3.5) below, D_i is the probability at the i^{th} scale that the input patch to the discriminator is from the real distribution at the i^{th} scale:

$$\mathcal{L}_{JS-GAN}(R,D_i) = \mathop{\mathbb{E}}_{x\sim P_{real}} \Big[\log(D_i(x))\Big] + \mathop{\mathbb{E}}_{\tilde{x}\sim P_{refiner}} \Big[\log(1-D_i(\tilde{x}))\Big] \tag{3.4}$$

To avoid $JS-GAN$ mode collapse during training and alleviate other training difficulties, replacing $\mathcal{L}_{JS-GAN}(R,D_i)$ with the improved WGAN loss function [62] is

investigated. λ is the penalty coefficient (it was set as 0.0001 for all the experiments in this chapter) and $\hat{x}$ are the random samples (For more details, please read [62]):

$$\mathcal{L}_{WGAN}(R,D_i) = \mathop{\mathbb{E}}_{\tilde{x}\sim P_{refiner}}\Big[D_i(\tilde{x})\Big] - \mathop{\mathbb{E}}_{x\sim P_{real}}\Big[D_i(x)\Big] + \lambda \mathop{\mathbb{E}}_{\hat{x}\sim P_{\hat{x}}}\Big[(||\nabla_{\hat{x}} D_i(\hat{x})||_2 - 1)^2\Big] \quad (3.5)$$

The experiments explored the difference in performance of these two GAN loss functions. $\mathcal{L}_{GAN}(R,D_i)$ can be either $\mathcal{L}_{JS-GAN}(R,D_i)$ or $\mathcal{L}_{WGAN}(R,D_i)$ in the following context. The difference of performance with both the single scale and multiple scales will also be explored.

(4) By inputting only the refined disparity map and its corresponding ground truth into the discriminator simultaneously in each step during training, the discriminator is trained in a fully supervised manner considering whether the input disparity maps are the same. In semi-supervised mode, the refined disparity map and its corresponding ground truth are fed into the discriminator for the labeled data. For the unlabeled data, the refined disparity map of the unlabeled data and random samples from a small ground truth dataset are fed simultaneously. By doing this, the discriminator will be taught to classify the input samples based on the disparity Markov Random Field. In turn, the refiner will be trained to produce a disparity Markov Random Field in the refined disparity map that is closer to the real case.

(5) The combined loss function in the fully supervised learning approach is:

$$\mathcal{L}(R,D) = \theta_1 \mathcal{L}_{L_1}^{Ld}(R) + \theta_2 \mathcal{L}_{sm}^{Ld}(R) + \theta_3 \sum_{i=1}^{M} \mathcal{L}_{GAN}^{Ld}(R,D_i) \quad (3.6)$$

where M is the number of the scales. θ_1, θ_2, θ_3 are the weights for the different loss terms. In the fully supervised learning approach (See Equation (3.6)), only the labeled data (denoted by Ld) is fed. In the semi-supervised learning (See Equation (3.7)), in each iteration, one batch of labeled data (denoted by Ld) and one batch of unlabeled data (denoted by Ud) are fed simultaneously. As for the labeled data Ld, its L1 loss (denoted by $\mathcal{L}_{L_1}^{Ld}$), smoothness loss (denoted by $\mathcal{L}_{sm}^{Ld}$), and GAN loss (denoted by $\mathcal{L}_{GAN}^{Ld}$) are calculated. The input to the discriminator is the refined disparity map (denoted by $Fake_1$) and corresponding ground truth (denoted by $Real_1$). The GAN loss for the labeled data Ld is calculated using $Fake_1$ and $Real_1$. As for the unlabeled data Ud, only its GAN loss ($\mathcal{L}_{GAN}^{Ud}$) is calculated and the other loss terms are neglected. The unlabeled data gets its refined disparity map (denoted by $Fake_2$) from the refiner. Feed $Real_1$ and $Fake_2$ into the discriminator to get the GAN loss for the unlabeled

data. As the experiment results show, this approach allows the use of much less labeled data (expensive) in a semi-supervised method (Equation (3.7)) to achieve similar performance to the fully supervised method (Equation (3.6)) or better performance when using the same amount of labeled data with additional unlabeled data compared with the supervised method. The combined loss function in the semi-supervised method is:

$$\mathcal{L}(R,D) = \theta_1 \mathcal{L}_{L_1}^{Ld}(R) + \theta_2 \mathcal{L}_{sm}^{Ld}(R) + \frac{\theta_3}{2}\Big(\sum_{i=1}^{M} \mathcal{L}_{GAN}^{Ld}(R,D_i) + \sum_{i=1}^{M} \mathcal{L}_{GAN}^{Ud}(R,D_i)\Big) \quad (3.7)$$

3.2.3 Network Architectures

A fully convolutional neural network [90] is adopted and also the partial architectures from [70, 73, 124] are adapted here for the refiner and discriminator. The refiner and discriminator use dense blocks to increase local non-linearity. Transition layers change the size of the feature maps to reduce the time and space complexity [70]. In each dense block and transition layer, modules of the form ReLu-BatchNorm-convolution are used. Two modules in the refiner and four modules in the discriminator in each dense block are used, where the filter size is 3×3 and stride is 1. The growth rate for each dense block is dynamic (unlike [70]). In each transition layer, only one module is used, where the filter size is 4×4 and the stride is 2 (except that in the third transition layer (Tran.3) of the discriminator the stride is 1).

Figure 3.2 shows the main architecture of the refiner, where $c1$ initial disparity inputs (the experiments below use $c1 = 2$ for 2 disparity maps) and $c2$ pieces of information (the experiments below use $c2 = 2$ for the left intensity image and a gradient of intensity image) are concatenated as input into the refiner. The batch size is b and input image resolution is $32m \times 32n$ (m, n are integers). lg is the number of the feature map channels after the first convolution. To reduce the computational complexity and increase the extraction ability of local details, each dense block contains only 2 internal layers (or modules above). Additionally, the skip connections [126] from the previous layers to the latter layers preserve the local details in order not to lose information after the network bottleneck. During training, a dropout strategy has been added into the layers in the refiner after the bottleneck to avoid overfitting and the dropout part is cancelled during test to produce a determined result.

Figure 3.3 is for the discriminator. The discriminator will only be used during training and abandoned during testing. The discriminator will only influence the

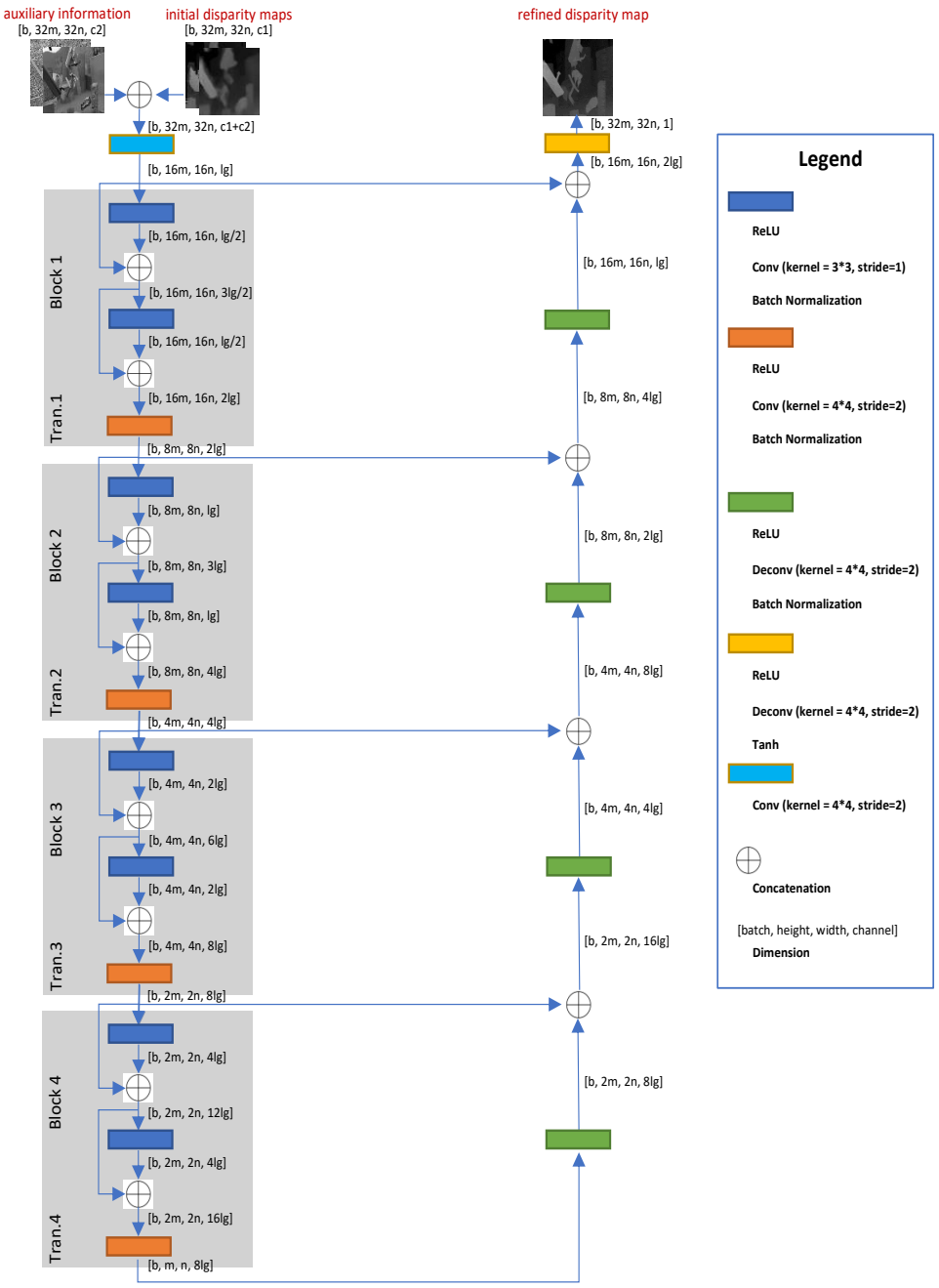


Figure 3.2: This figure shows some important hyperparameters and the refiner architecture configuration. Please refer to Table 3.2 for the specific values in each experiment.

computational costs during training. The initial raw disparity maps, information and real or refined disparity maps are concatenated and fed into the discriminator. Each dense block contains 4 internal layers (or modules above). The sigmoid function outputs the probability map ($Di, i = 1, 2, ..., 5$) that the local disparity patch is real or fake at different scales to force the Markov Random Field of the refined disparity map to get

closer to the real distribution at different receptive field sizes.

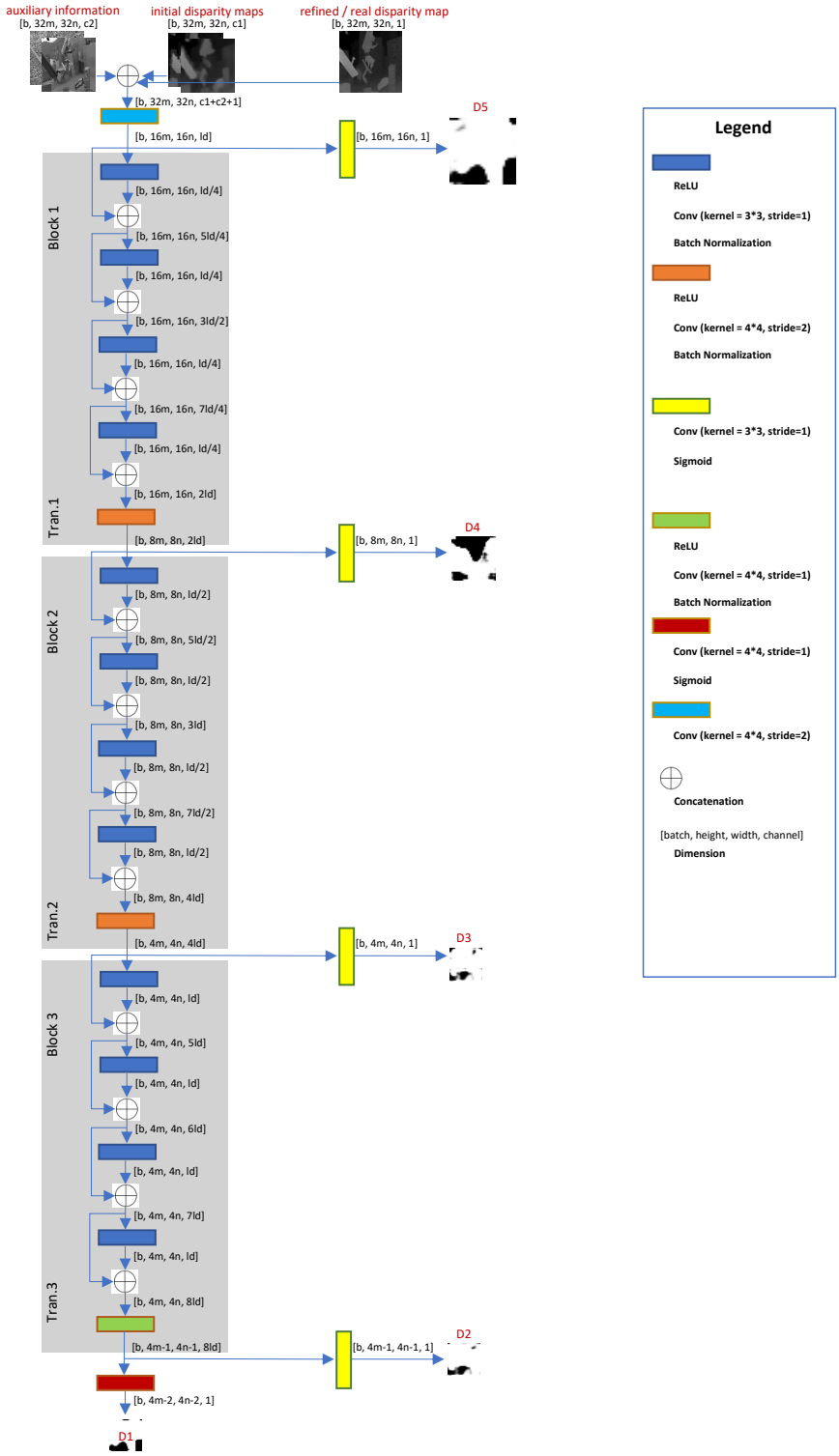


Figure 3.3: This figure shows some important hyperparameters and the discriminator architecture configuration. Please refer to Table 3.2 for the specific values in each experiment.

3.3 Experimental Evaluation

The network is implemented using TensorFlow [13] and trained & tested using an Intel Core i7-7820HK processor (quad-core, 8 MB cache, up to 4.4 GHz) and Nvidia Geforce GTX 1080Ti. First, an ablation study with initial raw disparity inputs ([68, 97]) is conducted using a synthetic garden dataset to analyze the influence of each factor in the energy function and the objective function. Secondly, three groups of experiments for three fusion tasks (monocular-stereo, stereo-ToF, stereo-stereo) show the robustness, accuracy and generality of the proposed algorithm using synthetic datasets (SYNTH3 [14], Scene Flow [97], our synthetic garden dataset (They are not available to the public currently)) and real datasets (Kitti2015 [98] dataset, Trimbot2020 Garden datasets (For more description, see Appendix A.1). A brief description of datasets in this chapter is shown in Table 3.1. In the semi-supervised method, as for each labelled training sample, it is used with its ground truth in the supervised part. It is used as well without its ground truth in the unsupervised part. All the results show the superiority of the proposed algorithm compared with the state-of-art or classical depth acquisition algorithms ([29, 59, 67, 68, 69, 97]), the state-of-art stereo-stereo fusion algorithms ([114]), the state-of-art stereo-ToF fusion algorithm [14, 94], and the state-of-art image style transfer algorithm [73].

Table 3.1: A Brief Description of Datasets Used in This Chapter.

	Supervised		**Semi-Supervised**	
Dataset	**Labeled Training Samples**	**Test Samples**	**Training Samples**	**Test Samples**
Synthetic Garden	4600	421	4600 (labeled)	421
Scene Flow	6000	1460	600 (labeled) + 5400 (unlabeled)	1460
SYNTH3	40	15	40 (labeled)	15
Kitti2015	150	50	None	None
Trimbot2020 Garden	1000	270	1000 (labeled)	270

In the following experiments, the inputs to the neural network were first scaled to $32m \times 32n$ and normalized to $[-1, 1]$. After that, the input was flipped vertically with a 50% chance to double the number of training samples. Weights of all the neurons were initialized from a Gaussian distribution (standard deviation 0.02, mean 0). All the models in all the experiments were trained with a batch size of 4 in the supervised and semi-supervised method, using Adam [84] with a momentum of 0.5. The learning rate is changed from 0.005 to 0.0001 gradually. The method in [61] is used to optimize the refiner network and discriminator network by alternating between one step on the

discriminator and one step on the refiner. The parameters θ_1, θ_2, θ_3 in Equation (3.6) or Equation (3.7) were set to make those terms contribute differently to the energy function in the training process. The L_1 distance between the estimated image and ground truth was used as the error (unit is pixel). For more details about the network settings and computational complexity, please see Table 3.2. The network is so fast that it can run the disparity fusion (e.g., up to 384 $\times$ 1280 pixels on Kitti2015 datasets) directly at 90 fps without any cropping (e.g., DSF [114] used samples with 9 $\times$ 9 pixels) or down-sampling.

Table 3.2: Computation Time and Parameter Settings.

Ablation Study with Synthetic Garden Dataset													
Para.	Test time	b	$32m$	$32n$	c_1	c_2	lg	ld	θ_1	θ_2	θ_3	α	β
Value	0.007 (s/frame)	4	480	640	2	2	12	12	395	5	1	1	650
Stereo-Monocular Fusion with Synthetic Scene Flow Dataset [14]													
Para.	Test time	b	$32m$	$32n$	c_1	c_2	lg	ld	θ_1	θ_2	θ_3	α	β
Value	0.042 (s/frame)	4	256	256	2	2	64	64	199	1	1	0.5	100
Stereo-ToF Fusion with Synthetic SYNTH3 Dataset [14]													
Para.	Test time	b	$32m$	$32n$	c_1	c_2	lg	ld	θ_1	θ_2	θ_3	α	β
Value	0.012 (s/frame)	4	544	960	2	2	16	16	395	5	1	1	1–1.3K
Stereo-stereo Fusion with Real Kitti2015 Dataset [98]													
Para.	Test time	b	$32m$	$32n$	c_1	c_2	lg	ld	θ_1	θ_2	θ_3	α	β
Value	0.011 (s/frame)	4	384	1280	2	2	16	16	1	1	1	1	1 –2K
Stereo-stereo Fusion with Real Trimbot2020 Garden Dataset													
Para.	Test time	b	$32m$	$32n$	c_1	c_2	lg	ld	θ_1	θ_2	θ_3	α	β
Value	0.008 (s/frame)	4	480	768	2	2	12	12	395	5	1	1	1–1.3K

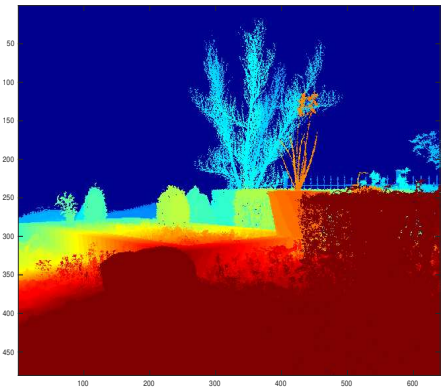
3.3.1 Ablation Study

This subsection shows the effectiveness of the loss function design in Section 3.3.1.1 and the influence of each factor in the final loss function in Section 3.3.1.2. All the experiments in this subsection are conducted on our synthetic garden dataset. The synthetic garden dataset contains 4600 training samples and 421 test samples under outdoor environments. Each sample has one pair of rectified stereo images and dense ground

truth with resolution 480 × 640 (height × width) pixels. The reason why a synthetic dataset is used is that the real dataset (e.g., Kitti2015) does not have dense ground truth, which will influence the evaluation of the network. Dispnet [97] and FPGA-stereo [68] were used to generate the two input disparity images. The authors of [68, 97] helped us get the best performance on the dataset as the input to the network. As for each model, it was trained for 100 epochs and it takes 20 h or so. The inference is fast (about 142 frames per second) for the 480 × 640 (Height × Width) resolution input. The performance demo on the synthetic garden dataset: `https://youtu.be/OqTj6h0QwUw` and Figure 3.4 shows one scene in the video.

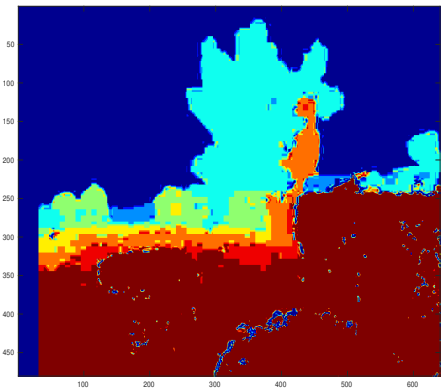
Figure 3.4: This figure shows a scene in the performance demo video on the synthetic garden dataset. The images from left to right in the first row are: left RGB image, disparity map from FPGA-stereo [68], disparity map from Dispnet [97]. The images from left to right in the second row are: the dense ground truth on the left view, disparity map from the proposed semi-supervised model, disparity map from the proposed supervised model.

One qualitative example is shown in Figure 3.5 from Section 3.3.1.1.

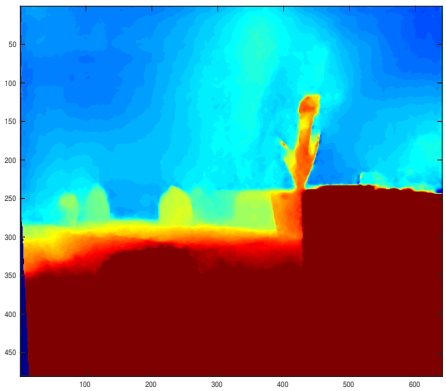


(a)

(b)

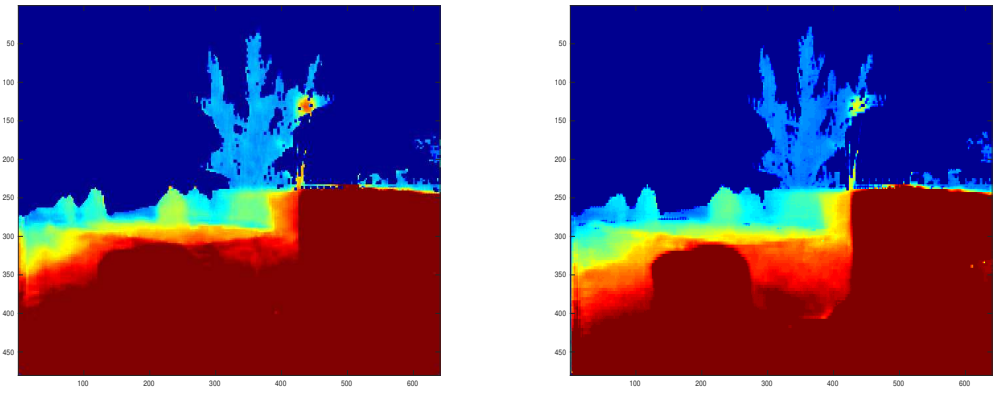


(c)



(d)

Figure 3.5: *Cont.*



(e) (f)

Figure 3.5: Two initial raw disparity inputs (**c**,**d**) were fused to get a refined disparity map (**e**,**f**) using our Supervised and Semi method on the synthetic garden dataset. (**a**) is the ground truth and (**b**) is the corresponding scene. Many, but not all, pixels from the fused result are closer to ground truth than the original inputs. (**a**) Ground Truth; (**b**) Scene; (**c**) FPGA Stereo [68]; (**d**) Dispnet [97]; (**e**) Our Supervised; (**f**) Our Semi.

3.3.1.1 Loss Function Design

The target in this part is to test the effectiveness of the objective function design from Section 3.2.2. Table 3.3 defines different combinations of the strategies that were evaluated, based on the objective functions defined in Section 3.2.2. The default network settings and some important parameters in this group of experiments, please see "Ablation Study" in Table 3.2.

Table 3.3: Model definition.

Model Name	Combination
Supervised	WGAN (Equation 3.5) + multiscale (M = 5) + supervised (Equation 3.6)
Semi	WGAN (Equation 3.5) + multiscale (M = 5) + semi-supervised (Equation 3.7)
Monoscale	WGAN (Equation 3.5) + monoscale (M = 1) + supervised (Equation 3.6)
JS-GAN	JS-GAN (Equation 3.4) + multiscale (M = 5) + supervised (Equation 3.6)

Table 3.4 shows the performance of each model. The same amount of data (4600 labeled samples) was used for the supervised and semi-supervised network training (where the semi-supervised training is augmented with the appropriate number of

refined disparity maps and random ground truth). The test data used 421 samples. The supervised and semi-supervised methods achieved similar good performance (Semi got the smallest error at 2.84 pixels). The error of the refined disparity map output by each network is much lower than the error of the input disparity maps. In the remaining experiments, only the multi-scale supervised and semi-supervised networks are used with WGAN.

Table 3.4: Mean absolute disparity error of each model on Synthetic Garden dataset (421 test samples).

	Inputs		**Experimental Outputs**			
Experiment	FPGA Stereo [68]	DispNet [97]	JS-GAN	Monoscale	Supervised	Semi
Error [px]	11.41	6.28	4.40	3.40	3.10	2.84

3.3.1.2 Influence of Each Term in Loss Function

In this part, one of the following factors (θ_1, θ_2, θ_3, α, β) in the energy function will be changed to see the influence of each cue in Equation (3.6). The 'Baseline' method in this part is also the 'Supervised' model from Section 3.3.1.1. The performance results are listed in Table 3.5. $\mathcal{L}_{L_1}(R)$ in Equation (3.2) has the largest influence (corresponding to θ_1) and then the gradient information in Equation (3.2) (corresponding to $\alpha = 0$). After that, the smoothing term in Equation (3.3) (corresponding to θ_2) and β have less influence compared with the former factors. The loss term in $\mathcal{L}_{GAN}$ (corresponding to θ_3) has the least influence.

Table 3.5: Ablation Study on Each Cue Using the Supervised Model.

	Inputs		**Experimental Outputs**					
Experiment	FPGA Stereo [68]	DispNet [97]	$\theta_1 = 0$	$\theta_2 = 0$	$\theta_3 = 0$	$\alpha = 0$	$\beta = 1$	Baseline
Error [px]	11.41	6.28	298.2	3.46	3.25	3.48	3.37	3.10

3.3.2 Robustness and Accuracy Test

Given that the proposed network does not need confidence values from the specific sensors, the network architecture can be generalized to fusion tasks using different data

sources. The following experiments will input different quality disparity maps from different sources to test the robustness and accuracy of the proposed algorithm.

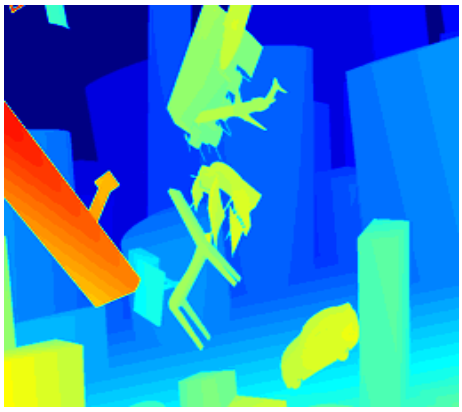
3.3.2.1 Stereo-Monocular Fusion

Monocular depth estimation algorithms are usually less accurate than stereo vision algorithms. Stereo vision algorithm PLSM [69] and monocular depth estimation algorithm Monodepth [59] were used to input the relevant initial disparity maps. Monodepth was retrained on the Scene Flow dataset (Flying A) with 50 epochs to get its left disparity maps. PLSM with semi-global matching computed the left disparity map without refinement. The default network settings and some important parameters of the networks in this part can be seen in "Stereo-Monocular Fusion" in Table 3.2. 6000 labeled samples (80%) in Scene Flow (Flying A) were used for the supervised training and 600 labeled samples (8%) + 5400 unlabeled samples for the semi-supervised training. Another 1460 samples (20%) were used for testing. DSF [114] is a recent high performance fusion algorithm that is used to compare with. Pix2pix [73] was set up to use PLSM + Monodepth as inputs and the fused disparity map as output. The code for Pix2pix is from its project website [1]. The reason to choose Pix2pix as a comparison algorithm is that disparity fusion can be seen as equivalent to an image style transfer and Pix2pix is a famous image style transfer algorithm. DSF was retrained for 10 epochs (about 5 h per epoch) and Pix2pix [73] was retrained for 100 epochs (0.15 h per epoch).

The relevant error of each algorithm is shown in Table 3.6. The supervised method (Num = 6000) and the semi-supervised method (Num = 600) achieve similar top performances while the semi-supervised method uses much less labeled training data (9 times less than the supervised method). Pix2pix behaves badly and is neglected in the following experiments. A qualitative result comparison can be seen in Figure 3.6.

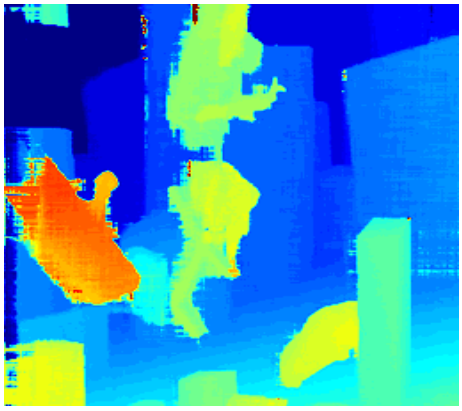
Table 3.6: Mean absolute disparity error of stereo-monocular fusion on Scene Flow (1460 test samples).

	Inputs		**Comparison**		**Our Fused**	
Training Data	**PLSM** [69]	**Monodepth** [59]	**DSF** [114]	**Pix2pix** [73]	**Supervised**	**Semi**
Num=600	2.41 px	3.30 px	2.00 px	2.91 px	1.95 px	1.60 px
Num=6000	2.41 px	3.30 px	1.87 px	2.65 px	1.55 px	NA



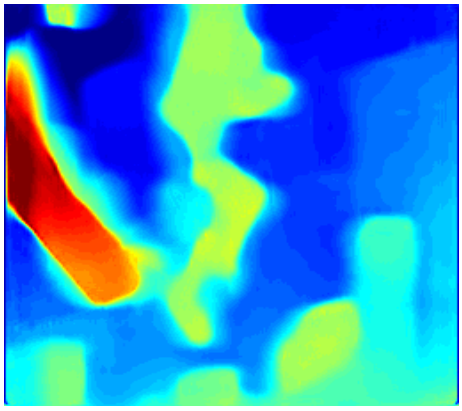
(a)

(b)



(c)

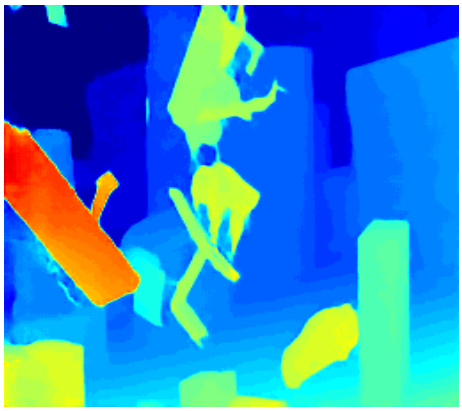
(d)



(e)

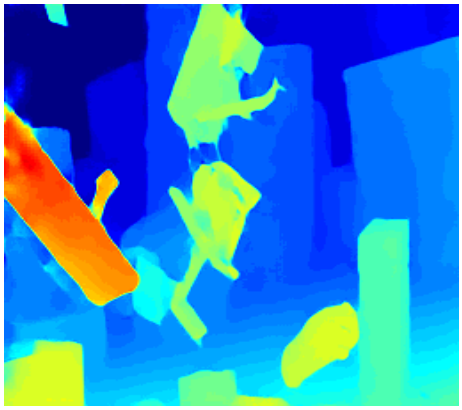
(f)

Figure 3.6: *Cont.*



(g)

(h)



(i)

(j)

Figure 3.6: A qualitative result with inputs from PLSM [69] and Monodepth [59] in stereo-monocular fusion. The lighter pixels represent bigger disparity errors in figure (**d**,**f**,**h**,**j**). (**a**) Ground Truth; (**b**) Color image; (**c**) PLSM [69]; (**d**) PLSM error; (**e**) Monodepth [59]; (**f**) Monodepth error; (**g**) Supervised 1; (**h**) Supervised 1 error; (**i**) semi 2; (**j**) Semi 2 error.

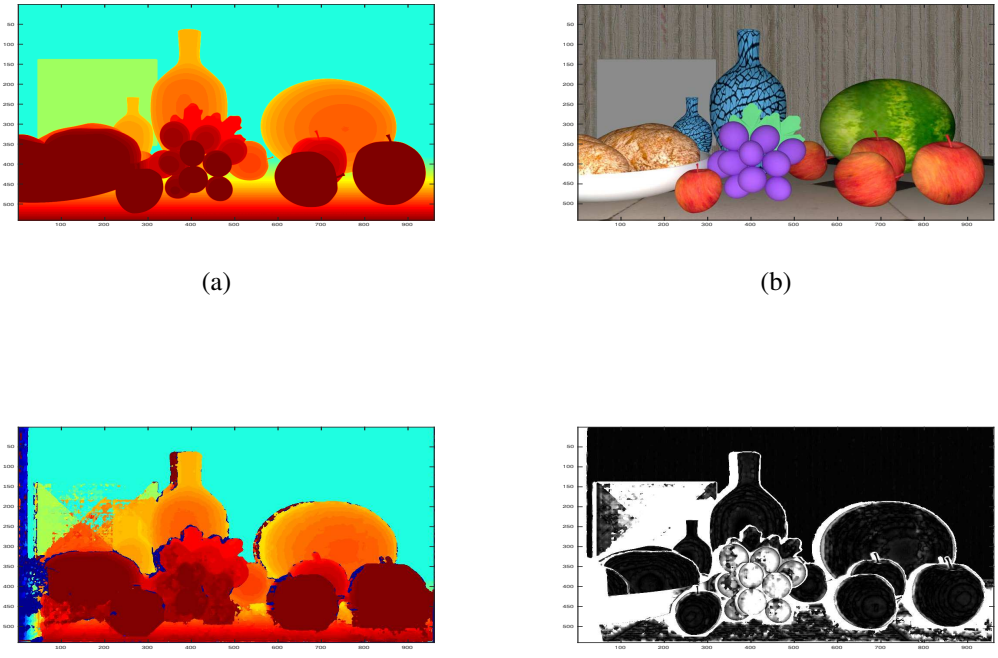
3.3.2.2 Stereo-ToF Fusion

The default network settings and some important parameters of the networks in this part, can be seen in "Stereo-ToF Fusion" in Table 3.2. The network was trained on the SYNTH3 dataset (40 training and 15 test samples with resolution 540 $\times$ 960 pixels). Semi-global matching from OpenCV was used to get the stereo disparity map, with the point-wise Birchfield-Tomasi metric, 7 $\times$ 7-pixel window size and 8-path optimization. The initial ToF depth map was projected onto the right stereo camera

image plane and up-sampled and converted to the disparity map. Limited by the very small number of training samples, the proposed networks do not reach their best performance. But, compared with the input disparity maps, the proposed methods perform slightly better (See Table 3.7). The experiment results for SGM stereo, ToF, LC [94] and DLF [14] are from the paper [14] because the proposed methods were tested on the same dataset as [14] from their website[2]. The proposed Supervised method performs less well because of the insufficient number of training samples. However, the proposed Semi method ranks first among all of the stereo-ToF fusion algorithms. One qualitative result is shown in Figure 3.7.

Table 3.7: Mean absolute disparity error of ToF-stereo fusion on SYNTH3 (15 test samples).

	Inputs		**Comparison**		**Our Fused**	
Training Data	**SGMStereo**	**ToF**	**LC** [94]	**DLF** [14]	**Supervised**	**Semi**
Num=40	3.73 px	2.19 px	2.07 px	2.06 px	2.18 px	2.02 px

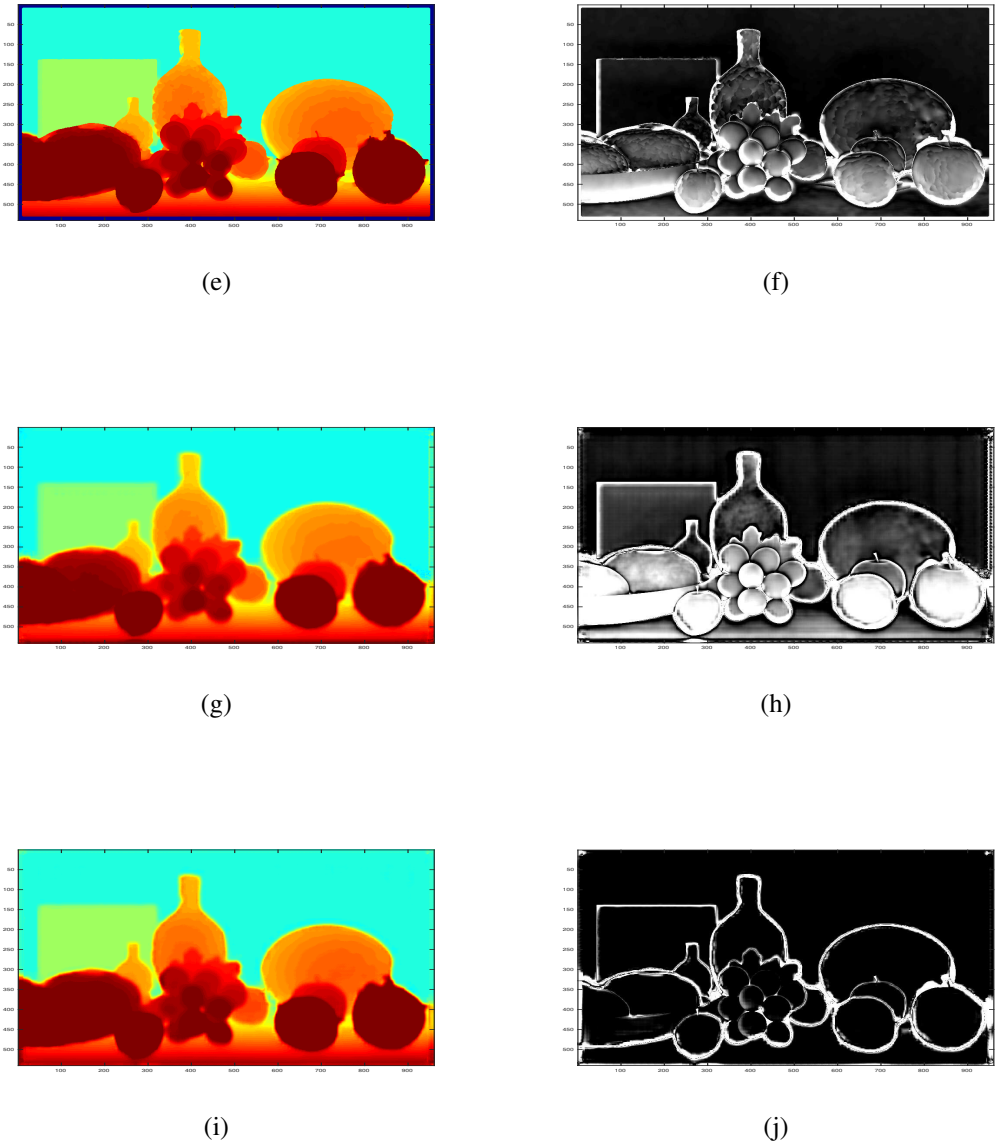


(a) (b)

(c) (d)

Figure 3.7: *Cont.*

[2]URL: `http://lttm.dei.unipd.it/paper_data/deepfusion/`



(e) (f)

(g) (h)

(i) (j)

Figure 3.7: One qualitative result for ToF-stereo fusion with many invalid pixels input. The inputs are from ToF and disparity calculation algorithm using SGM in OpenCV [6]. The lighter pixels in (**d**,**f**,**h**,**j**) represent larger disparity error. (**a**) Ground Truth; (**b**) Color image; (**c**) ToF; (**d**) ToF error; (**e**) SGM OpenCV; (**f**) SGM error; (**g**) Supervised 1; (**h**) Supervised error; (**i**) Semi 2; (**j**) Semi 2 error.

3.3.2.3 Stereo-Stereo Fusion

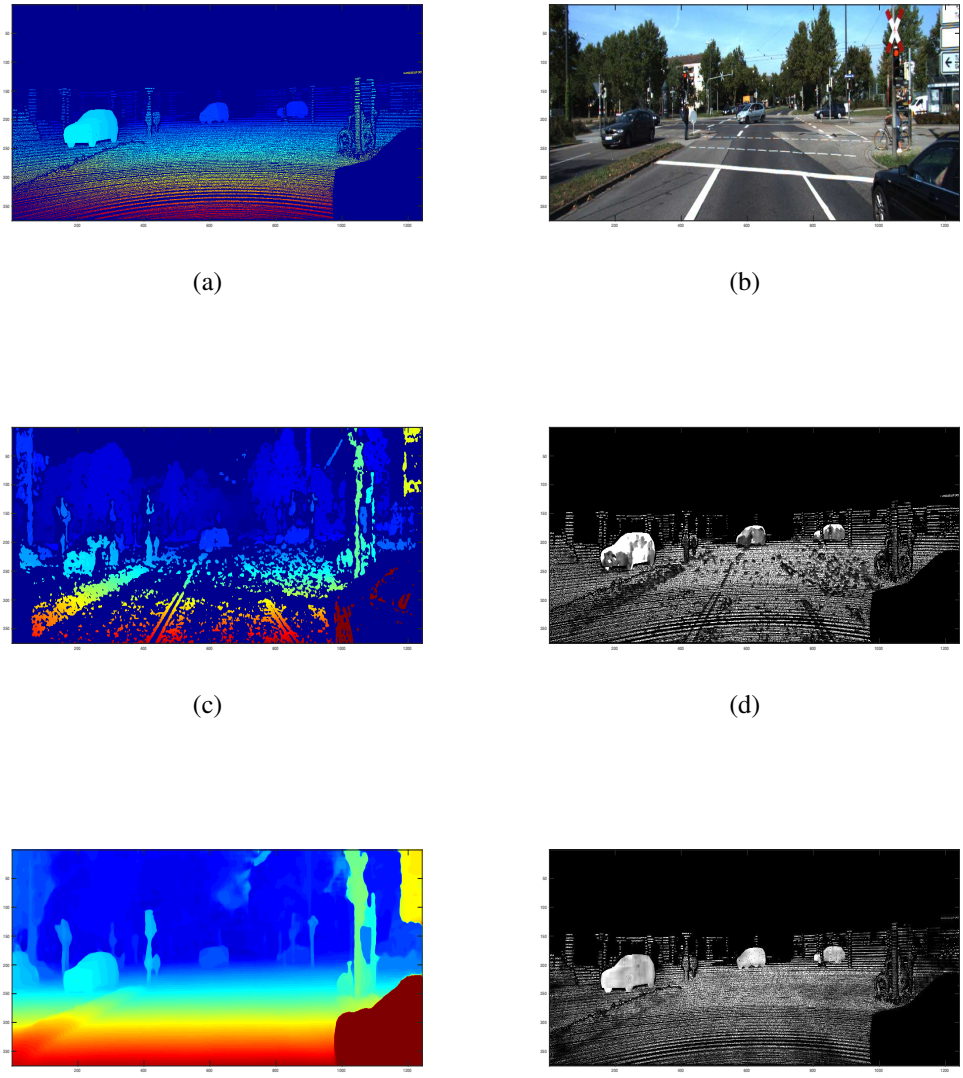
The Kitti2015 and TrimBot2020 Garden datasets have been used to test the stereo-stereo fusion. The experiment results have been shown as follows.

3.3.2.3.1 Performance on Kitti2015 Dataset The proposed network was tested on the real Kitti2015 dataset, which used a Velodyne HDL-64E Lidar scanner to get the sparse ground truth and a 1242 $\times$ 375 resolution stereo camera to get stereo image pairs. The initial training dataset contains 200 labeled samples. 50 samples from '000000_10.png' to '000049_10.png' in the Kitti2015 training dataset were used as the test dataset in this part. The other 150 samples were used as the training set for fine-tuning in this part. By flipping the training samples vertically, it doubled the number of training samples. The state-of-art stereo vision algorithm PSMNet [29] was used as one of the inputs. Their released pre-trained model[3] on the Kitti2015 dataset was used to get the disparity maps. A traditional stereo vision algorithm SGM [67] was used as the second input to the network. Their parameters were set to produce more reliable but sparse disparity maps. More specifically, the implementation ('disparity' function) from Matlab2016b was used. The relevant parameters are: 'DisparityRange' [0, 160], 'BlockSize' 5, 'ContrastThreshold' 0.99, 'UniquenessThreshold' 70, 'DistanceThreshold' 2. The settings of the neural network are shown in "Stereo-stereo Fusion with Real Kitti2015 Dataset" in Table 3.2. The proposed algorithms were compared with the state-of-art technique [114] in stereo-stereo fusion and also stereo vision inputs [29, 67]. As the ground truth of Kitti2015 is sparse, the semi-supervised method (which requires learning the disparity Markov Random Field) was not compared. The supervised method was trained on the synthetic garden dataset first and fine-tuned the pre-trained model on the Kitti2015 dataset. 150 labeled samples from '00050_10.png' to '000199_10.png' in the initial training dataset were used for the fine-tuning of the supervised method. The relevant results are shown in Table 3.8. The same conclusion can be made as with the stereo-monocular and stereo-ToF fusion: the proposed method is accurate and robust. An example result of stereo-stereo fusion is shown in Figure 3.8. The proposed method compensates for the weaknesses of the inputs and refines the initial disparity maps effectively. Compared with SGM [67] (0.78 pixels) (This is a more accurate disparity but is calculated only using more reliable pixels. On average only 40% of the ground truth pixels are used. If all the valid ground truth is used to calculate its error, it is 22.13 pixels), the fused results are much more dense and accurate. Compared with PSMNet, the proposed method preserves the details better (e.g., tree, sky), which are missing in the ground truth though. The proposed network can deal with the input (resolution: 384 $\times$ 1280) at 0.011 s/frame, which is real-time and very fast.

[3]Code and pre-trained model for PSMNet [29]: `https://github.com/JiaRenChang/PSMNet`

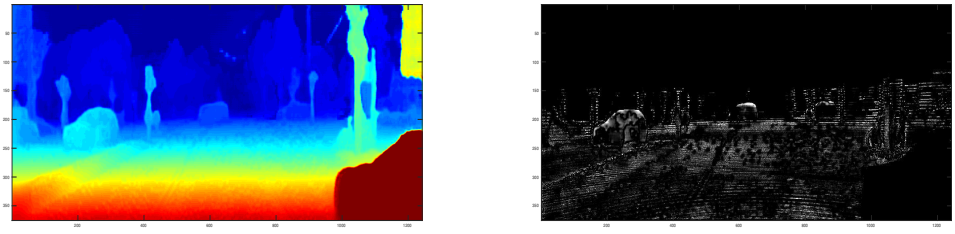
Table 3.8: Mean absolute disparity error of stereo-stereo fusion on Kitti2015 (50 test samples).

	Inputs		**Comparison**	**Our Fused**
Training Data	**SGM** [67]	**PSMNet** [29]	**DSF** [114]	**Supervised**
Num=150	0.78 px	1.22 px	1.20 px	1.17 px



(a) (b) (c) (d) (e) (f)

Figure 3.8: *Cont.*



(g) (h)

Figure 3.8: The network was trained to fuse the initial disparity maps (**c**,**e**) into a refined disparity map (**g**) for the same scene (**b**) from the Kitti2015 dataset [98] using our supervised method. (**a**) is the corresponding ground truth. (**d**,**f**,**h**) are the errors of (**c**,**e**,**g**). The lighter pixels have bigger disparity error in (**d**,**f**,**h**). (**a**) Ground Truth; (**b**) Scene; (**c**) Input Disparity 1: SGM [67]; (**d**) Input Disparity 1 Error: SGM [67]; (**e**) Input Disparity 2: PSMNet [29]; (**f**) Input Disparity 2 Error: PSMNet [29]; (**g**) Refined Disparity; (**h**) Refined Disparity Error.

3.3.2.3.2 Performance on Trimbot2020 Garden Dataset The proposed network was tested on the real Trimbot2020 Garden dataset, which used a Leica ScanStation P15 to capture a dense 3D Lidar point cloud of the whole real garden and project it to each camera view to get the dense ground truth disparity maps. A 480×752 resolution stereo camera was used to get stereo image pairs. The Trimbot2020 Garden dataset contains 1000 labeled samples for training and 270 labeled samples for testing. The network was trained on the synthetic garden dataset first and fine-tuned on the real garden dataset. Dispnet [97] and FPGA-stereo [68] were used as inputs. The authors of [68, 97] helped get the best performance on the real Trimbot2020 Garden dataset as the input to the network. The settings of the network are shown in "Stereo-stereo Fusion with Real Trimbot Garden Dataset" in Table 3.2. The relevant error of each algorithm on valid pixels is shown in Table 3.9.

Table 3.9: Mean absolute disparity error of stereo-stereo fusion on Trimbot Garden Dataset (270 test samples).

	Inputs		**Comparison**	**Our Fused**	
Training Data	**FPGA Stereo [68]**	**Dispnet [97]**	**DSF [114]**	**Supervised**	**Semi**
Num=1000	2.94 px	1.35 px	0.83 px	0.67 px	0.66 px

The supervised method and the semi-supervised method have achieved similar top performances compared with the rest. The proposed network can deal with the input (resolution: 480×768) at 125 fps, which is faster than real-time. The demo in the real outdoors garden is available from `https://youtu.be/2yyoXSwCSeM` and Figure 3.9 shows one scene in the video.

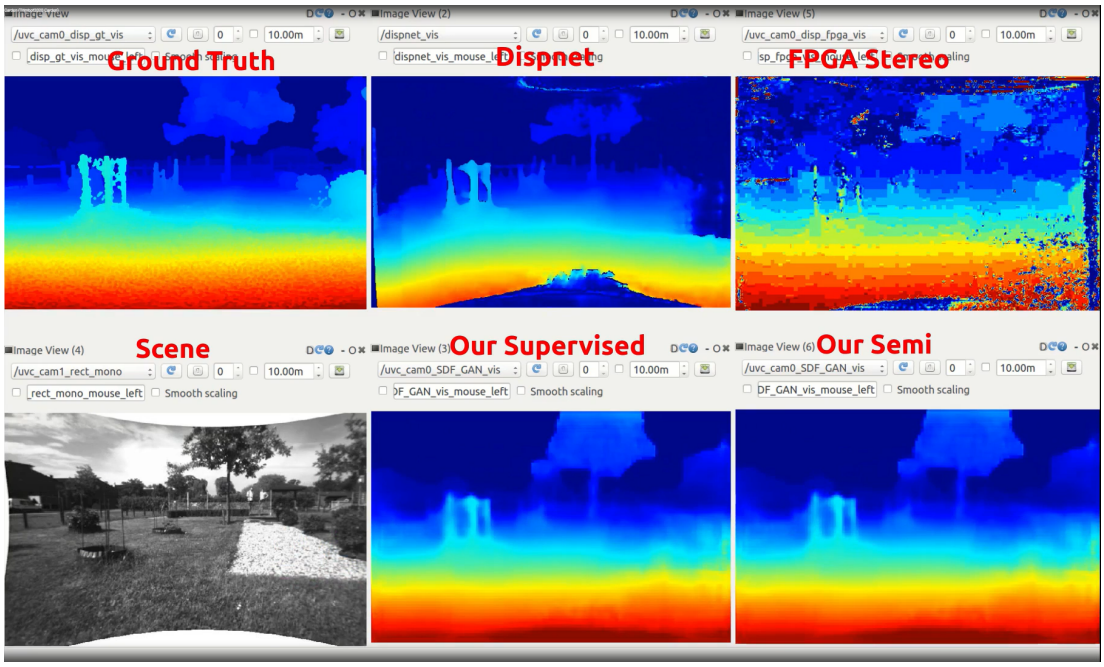
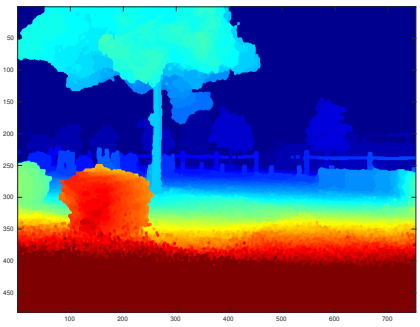


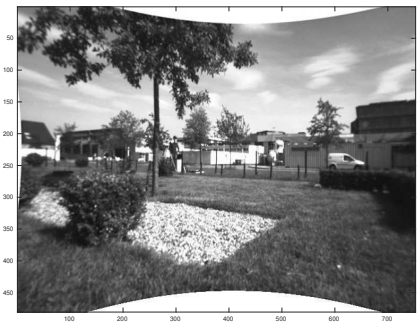
Figure 3.9: This figure shows a scene in the performance demo video on the TrimBot2020 Garden dataset. The images from left to right in the first row are: ground truth disparity map, disparity map from Dispnet [97], disparity map from FPGA-stereo [68]. The images from left to right in the second row are: intensity image for the scene, disparity map from the proposed supervised model, disparity map from the proposed semi-supervised model.

A qualitative result comparison can be seen in Figure 3.10.

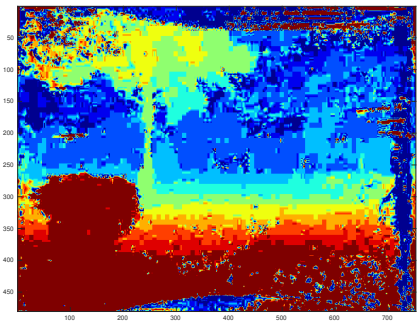
After getting the refined disparity map, the refined disparity map is projected into 3D space by using the camera intrinsic matrix. Finally, the 3D model of the whole garden is built using the relative camera pose between each two frames. The integration pipeline in ROS system from the rectified image inputs to the 3D model of the whole garden has been presented in Chapter 6.1 and the further evaluation of the supervised method on the real robot in Chapter 6.2.1.



(a)

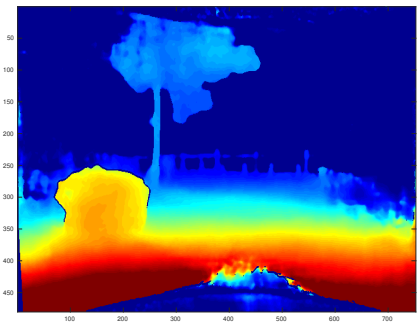


(b)

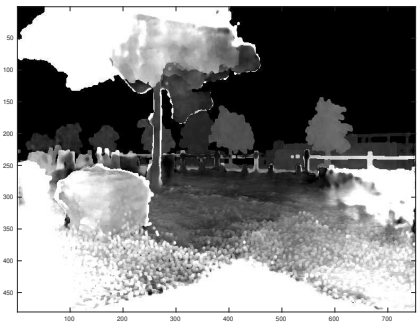


(c)

(d)



(e)



(f)

Figure 3.10: *Cont.*

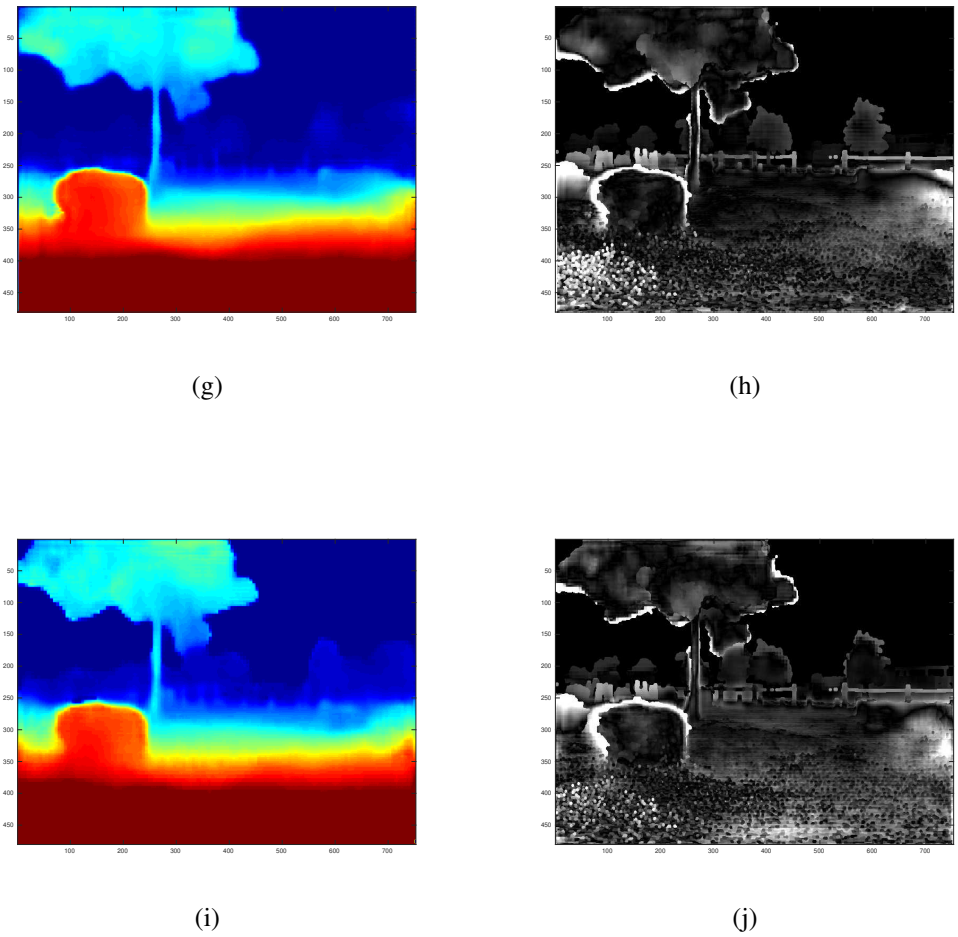


Figure 3.10: One qualitative result for stereo-stereo fusion in real Trimbot2020 Garden Dataset. The lighter pixels in (**d**,**f**,**h**,**j**) represent larger disparity error. (**a**) ground truth; (**b**) intensity image; (**c**) FPGA SGM; (**d**) FPGA SGM error; (**e**) DispNet; (**f**) DispNet error; (**g**) Supervised 1; (**h**) error 1 (**i**) Semi 2; (**j**) error 2.

3.3.3 Sensitivity Analysis

All the following experiments are conducted on the Trimbot2020 Garden dataset using the same settings with Performance on Trimbot2020 Garden Dataset in Section 3.3.2.3.2 except the control variables. The sensitivity analysis is done for the parameter $alpha$ in Equation (3.2), the number of scales M in Equation (3.6) and Equation (3.7), the number of feature maps for the refiner network and discriminator network architectures $lg = ld = L$, and also the parameter momentum in the optimization algorithm Adam.

3.3.3.1 Alpha

Table 3.10 (corresponding to Figure 3.11) shows the performance change when $alpha$ varies from 0.5 to 1.5 with an interval 0.25. Figure 3.11 shows the robustness of the proposed algorithm. When $alpha = 1$, it achieves its best performance.

Table 3.10: Sensitivity Analysis ($Alpha$).

$Alpha$	**0.5**	**0.75**	**1**	**1.25**	**1.5**
Supervised	0.75	0.69	0.67	0.86	0.72
Semi	0.71	0.69	0.66	0.74	0.85

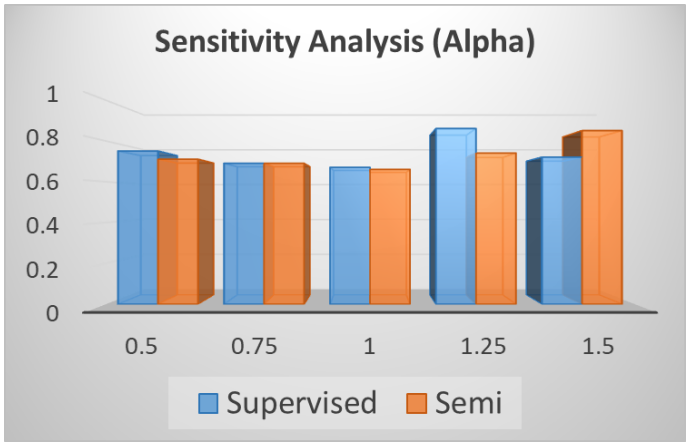


Figure 3.11: Sensitivity Analysis (Alpha).

3.3.3.2 The Number of Scales

Table 3.11 (corresponding to Figure 3.12) shows the performance change when the number of scales M varies from 1 to 5 with an interval 1. Figure 3.12 shows that with the increment of the number of scales, the error decrease gradually. Therefore $M = 5$ is chosen.

Table 3.11: Sensitivity Analysis (M).

M	**1**	**2**	**3**	**4**	**5**
Supervised	0.87	0.81	0.74	0.69	0.67
Semi	0.80	0.80	0.79	0.74	0.66

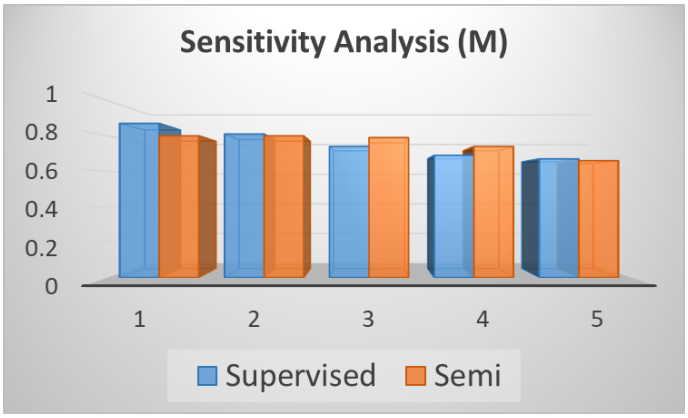


Figure 3.12: Sensitivity Analysis (M).

3.3.3.3 The Number of Feature Maps

Table 3.12 (corresponding to Figure 3.13) shows the performance change when L varies from 6 to 18 with an interval 3. Figure 3.13 shows that with the increment of the number of feature maps' channels, the overall performance does not change too much but when $L = 12$ it performs best.

Table 3.12: Sensitivity Analysis (L).

L	**6**	**9**	**12**	**15**	**18**
Supervised	0.75	0.85	0.67	0.81	0.78
Semi	0.76	0.77	0.66	0.73	0.72

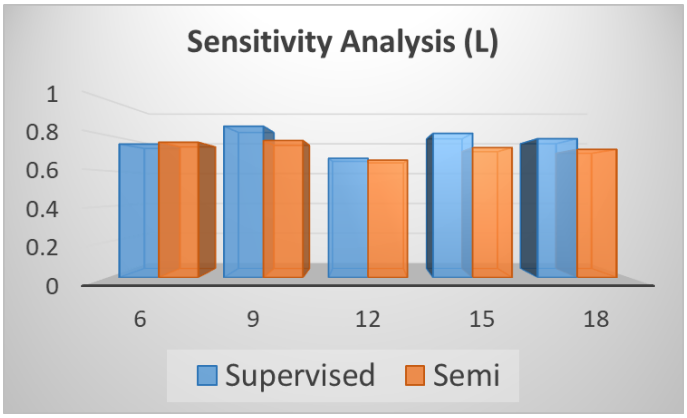


Figure 3.13: Sensitivity Analysis (L).

3.3.3.4 Momentum

As for the momentum in the Adam optimization algorithm, different momentum values are used to repeat the experiments again. The experimental results are shown in Table 3.13 and Figure 3.14. Table 3.13 (corresponding to Figure 3.14) shows the performance change when momentum varies from 0.1 to 0.9 with an interval 0.1. Figure 3.14 shows that when momentum is larger than 0.5, it achieves better performance compared with below 0.5. When it is equal to 0.5, both the supervised and semi-supervised methods achieve the best performance simultaneously.

Table 3.13: Sensitivity Analysis (Momentum).

Momentum	**0.1**	**0.3**	**0.5**	**0.7**	**0.9**
Supervised	0.81	0.83	0.67	0.76	0.67
Semi	0.87	0.90	0.66	0.66	0.70

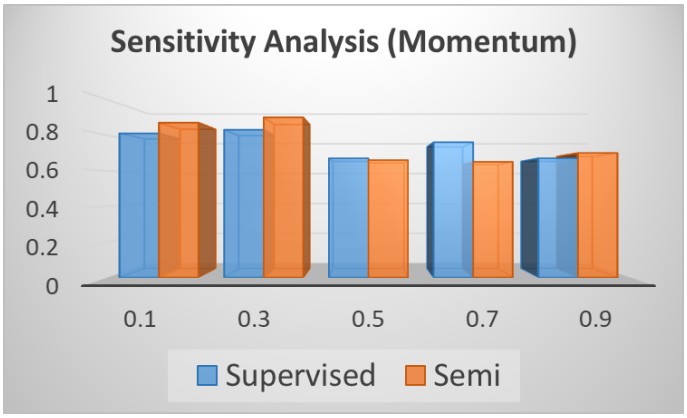


Figure 3.14: Sensitivity Analysis (Momentum).

3.3.3.5 Statistical Analysis

In this chapter, each experiment setting above has been used once. To show the robustness and accuracy of the proposed method and avoid randomness, the experiments on the real TrimBot2020 datasets were repeated five times using the same settings. The corresponding results are shown in Table 3.14. Compared with DSF algorithm, the mean error of the proposed algorithms is smaller, which shows they are more accurate statistically. Their standard deviation is slightly larger, showing they are marginally less

robust. It is less clear if there is a significant difference between the supervised and semi-supervised performances.

Table 3.14: Statistical Analysis.

	Repeated Experiment					**Statistical Result**	
Experiment	**1**	**2**	**3**	**4**	**5**	**Mean**	**Std.**
DSF	0.83	0.89	0.86	0.87	0.85	0.86	0.02
Supervised	0.73	0.77	0.67	0.70	0.72	0.72	0.04
Semi	0.67	0.71	0.66	0.76	0.71	0.70	0.04

3.4 Conclusion

This chapter first presented a supervised & a semi-supervised method to refine disparity maps by fusing the results from multiple depth calculation algorithms and other supplementary image information (such as intensity and gradient). In the previous work, researchers designed different methods for different fusion tasks requiring detailed knowledge of the performance of each sensor (eg: ToF-stereo depth fusion [14, 17, 38, 94, 104], Lidar-stereo fusion [33, 92, 109], stereo-stereo fusion [114]). Unlike the previous work above, the proposed methods can generalise to perform different fusion tasks without requiring detailed knowledge of the performance of each sensor. All the results in this chapter show the superiority of the proposed algorithms compared with the state-of-art depth acquisition algorithms ([29, 59, 67, 68, 69, 97]), the state-of-art stereo-stereo fusion algorithms ([114]), the state-of-art stereo-ToF fusion algorithm [14, 94], and the state-of-art image style transfer algorithm [73]. The proposed methods in this chapter could potentially fuse multiple algorithms (not only 2 algorithms as shown in this chapter) by concatenating more initial disparity maps in the input of the network, but this has not been explored. The objective function and network architecture are novel and effective. In addition, the proposed semi-supervised method greatly reduces the amount of ground truth training data needed, while achieving comparable performance with the proposed supervised method. The proposed semi-supervised method can achieve better performance when using the same amount of labelled data as the supervised method plus the additional unlabelled data. However, both the supervised & semi-supervised method still need the labelled data, which is

expensive. In the next chapter, unsupervised disparity fusion with adversarial neural networks will be explored.

Chapter 4

Unsupervised Depth Fusion from Multiple Sources

Existing disparity fusion methods based on deep learning achieve state-of-the-art performance, but they require ground truth disparity data to train. So far, this is the first time an unsupervised disparity fusion not using ground truth disparity data has been proposed.

In this chapter, a mathematical model for disparity fusion is proposed to guide an adversarial network to train effectively without ground truth disparity data. The initial disparity maps are input from the left view along with auxiliary information (gradient, left and right intensity image) into the refiner and the refiner is trained to output the refined disparity map registered on the left view. The refined left disparity map and left intensity image are used to reconstruct a fake right intensity image. Finally, the fake and real right intensity images (from the right stereo vision camera) are fed into the discriminator. In the model, the refiner is trained to output a refined disparity value close to the weighted sum of the disparity inputs for global initialisation. Three refinement principles are adopted to refine the results further. (1) The reconstructed intensity error between the fake and real right intensity image is minimised. (2) The similarities between the fake and real right image in different receptive fields are maximised. (3) The refined disparity map is smoothed based on the corresponding intensity image.

The adversarial network architectures are effective for the fusion task. The inference time using the proposed network is small. The network can achieve 90 fps using Nvidia Geforce GTX 1080Ti on the Kitti2015 dataset when the input resolution is 1242×375 ($Width \times Height$) without downsampling and cropping. The accuracy of this work is equal to (or better than) the state-of-the-art supervised methods.

The results presented in this chapter were achieved during September 2018. The majority of the content in this chapter is from the published paper [117].

4.1 Introduction

With the popularity of the disciplines related to 3D vision (eg: robotics, augmented and mixed reality, autonomous driving), how to get more accurate depth information using cheap devices in a 3D environment is important. Currently, there are many methods to obtain depth information, such as active illumination devices (eg: structured light cameras, Time of Flight (ToF) sensors), passive methods (monocular depth estimation [59], stereo vision [29, 67, 68, 97]) and so on. However, none of these methods is perfect in all scenes. For example: ToF is sensitive to sunshine outdoors and reflectivity of the materials; Vision-based methods are sensitive to scene content (repetitive or textureless regions); Lidar-based devices are expensive and data is sparse and lacks color information. Depth fusion from multiple sources is urgently needed, where different data sources can compensate for the weaknesses of each other.

In recent years, different kinds of depth fusion methods have emerged in different sub-tasks, such as stereo-ToF fusion [14, 38, 94], stereo-stereo fusion [114], Lidar-stereo fusion [92, 109] and general depth fusion [119]. Additionally, deep-learning based methods perform much better than the rest. However, all of these algorithms are supervised and require much ground truth depth data to train [14, 109, 114, 119]. They suffer from two big problems: (1) Ground truth depth data is hard to get and expensive. (2) It is hard to generalize well with a limited amount of labeled data. As far as I know, the proposed algorithm in this chapter is the first to develop a fully unsupervised depth fusion method, which solves the problems above and can fuse the depth inputs of different quality effectively without using ground truth depth data.

Unsupervised disparity fusion is difficult because the algorithm requires to be able to produce an extremely accurate disparity map without any ground truth disparity data. The existing unsupervised strategy based on the left and right intensity consistency cannot guarantee a highly accurate disparity map. For example, Monodepth [59] treated the left-right intensity consistency error as a global metric in their cost function to obtain the disparity value but slight intensity changes in the images will influence the global estimation strongly. However, the left-right intensity consistency is just one of our local refinement metrics, which increases the global robustness and accuracy in turn. Previous work, such as the supervised and semi-supervised model in Sdf-MAN [119] (or Chapter 3 in this thesis), achieved top disparity fusion performance but these algorithms need the ground truth disparity data to train. By combining the global disparity initialization with local disparity refinement, we show that our network can be

trained without any ground truth depth data.

In this chapter, a fully unsupervised disparity fusion framework is proposed without the requirement of ground truth depth data. The initial disparity maps from the left view along with the auxiliary information (gradient, left & right intensity image) are input into the refiner (a network similar to the generator in GAN [61] but without noise input) and the refiner is made to output the refined disparity map registered to the left view. The refined left disparity map and left intensity image are used to reconstruct the fake right intensity image. Finally, the fake and real right intensity images (from the right stereo vision camera) are fed into the discriminator (See Fig. 4.1). In the model, the refiner is trained to output a refined disparity value close to the weighted sum of the disparity inputs for global initialization (Equation 4.1). Three refinement principles are adopted to refine the depth further. (1) The reconstructed intensity error between the fake and real right intensity image is minimized (Equation 4.2). (2) The similarities between the fake and real right image in different receptive fields are maximized (Equation 4.3). (3) The refined disparity map is smoothed based on the corresponding intensity image space (Equation 4.4).

A novel and efficient network structure has been designed as well (See Figure 4.2 for refiner, Figure 4.3 for discriminator). In the refiner, from the input layer to the bottleneck, the dense blocks and transition layers [70] are used to increase local non-linearity to obtain more local detailed information. Long skip connections from previous layers to later layers are added to preserve the lost detail information after the bottleneck. The discriminator outputs the probability of the input image (real right image or reconstructed right image) being from the real distribution at different receptive field sizes (or different scales). Also, the dense blocks and transition layers have been adapted to increase the ability of the discriminator.

Section 4.2 presents the pipeline of the proposed work, the mathematical model used in the design of the objective function for the network and the architecture for the refiner and discriminator. Section 4.3 presents the experimental results.

The main contributions presented in this chapter are (Section 4.3 gives a more detailed comparison with [29, 67, 68, 97, 114, 119]):

- An efficient unsupervised strategy by combining global disparity initialization and local refinement
- An indirect method using an adversarial network to force the disparity Markov Random Field of the refined disparity map to be close to the real

- An unsupervised end-to-end uncertainty-based pipeline to fuse any disparity input

4.2 Methodology

First a pipeline using a refiner network (similar to the generator in GAN [61] without noise input) is proposed to realize disparity fusion and a mathematical model is built to design the cost function for the fully unsupervised network. Finally, the refiner and discriminator architectures are introduced.

4.2.1 Fusion Pipeline

Similar to [119], a refiner network has been used to map coarse disparity inputs to a real disparity distribution deterministically using multi-modal information (disparity, intensity, intensity gradient). Unlike [14, 114, 119], a fully unsupervised method is used to train the neural network without ground truth. Inspired by [59], the disparity output from the left view is treated as a hidden layer and used to reconstruct the intensity image of the right view. The whole process is shown in Figure 4.1.

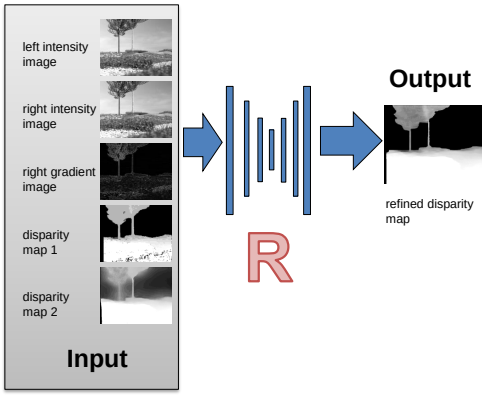
4.2.2 Objective Function

Using the reconstructed intensity error as a metric usually cannot guarantee a highly precise disparity output (eg: [59]), especially in environments with similar color and repetitive patterns (eg: garden). Additionally, the fusion accuracy may decrease compared with highly accurate disparity inputs if only the reconstructed intensity error is used as the decision metric. A more complex mathematical model is proposed as the cost function for disparity fusion.

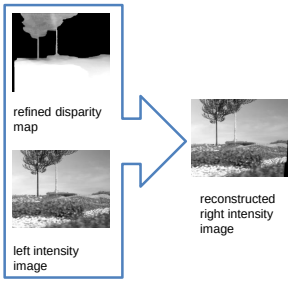
The main ideas for the mathematical model:

• The target is disparity fusion, whose accuracy should depend on the input disparity accuracy. The initial disparity maps should be used to provide the global initial value for the refined disparity map first. That is, the output of the refiner is encouraged to be similar to the input disparities (Equation 4.1).

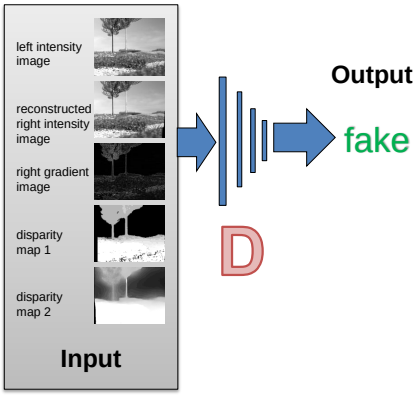
• The initialization based on Equation 4.1 can provide a coarse disparity map. The refinement will be realized by three local decision strategies discussed below together. The fake right intensity image is reconstructed from the left intensity image and disparity map. The accuracy of the refined disparity map can be assessed indirectly



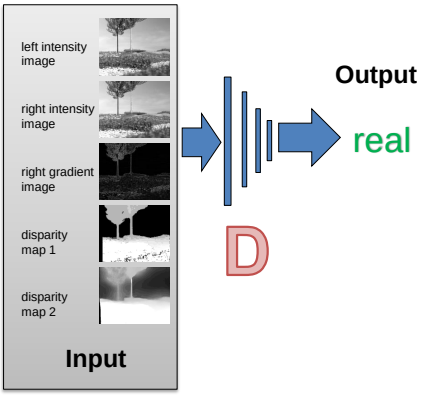
(a) Refiner



(b) Reconstruct right intensity image



(c) Negative examples



(d) Positive examples

Figure 4.1: (a): The inputs to the refiner (R) are the initial disparity maps ('disparity map 1', 'disparity map 2') in the left view and auxiliary information (left intensity image, right intensity image, right gradient image). The gradient of the left view is calculated from the left intensity image directly. The refiner produces a refined disparity map in the left view. (b): The refined left disparity map and left intensity image are used to reconstruct the right intensity image. (c)(d): The input to discriminator (D) is the combination of the left intensity image, right gradient image, the initial disparity inputs and the reconstructed/real right image. The discriminator tries to discriminate whether the input is fake (reconstructed right image) in (c) or true (real right image) in (d). The images in each block come from the training process on a synthetic garden dataset.

by comparing the reconstructed right image and real right image. The L_1 intensity error is designed based on the gradient in Equation 4.2 and the distance between the Markov Random Field of the refined disparity map and real disparity distribution is described in Equation 4.3 indirectly. A disparity smoothness term is also designed to reduce the

outliers and strong noise in Equation 4.4 using the gradient.

More specifically, the cost function has been designed as the following:

(1) Different from classical stereo vision algorithms without initial disparity inputs, disparity fusion is aimed at refinement of initial disparity inputs. A constraint is added that the output should be close to the weighted sum of the initial disparity inputs at each pixel. The output accuracy will increase with the accuracy enhancement of the initial disparity inputs.

$$\mathcal{L}_c(G) = \mathop{\mathbb{E}}_{u\in\tilde{x},u_s\in\bar{x}_s,\tilde{x}\sim P_G,s=1..Z}[w_{u_s}||\tilde{x}_u - \bar{x}_{u_s}||_1] \tag{4.1}$$

In Equation 4.1, $\bar{x}_{u_s}$ is the disparity value of a pixel u_s of the s^{th} initial disparity input (In Fig. 4.1, it is 'disparity map 1' or 'disparity map 2') corresponding to u in the refined disparity output $\tilde{x}$ (In Fig. 4.1, it is 'refined disparity map'). P_G represents the distribution of the samples $\tilde{x}$ from the refiner. w_{u_s} is the confidence of the pixel u_s from the s^{th} initial disparity input. If no prior knowledge is available, $w_{u_s} = 1/Z$ for all pixels. Z is the number of initial disparity inputs.

(2) To encourage disparity estimates at edges to be more accurate, gradient information is integrated as a weight into the L_1 distance to make the disparity edges clearer.

$$\mathcal{L}_{L_1}(G) = \mathop{\mathbb{E}}_{I_r\sim P_R,\tilde{I}_r\sim P'_G}[\exp(\alpha|\nabla(I_r)|)||I_r - \tilde{I}_r||_1] \tag{4.2}$$

In Equation 4.2, I_r is the real right intensity image from the right camera (In Fig. 4.1, it is 'right intensity image'.) and $\tilde{I}_r$ is the reconstructed right intensity image from the refiner (In Fig. 4.1, it is 'reconstructed right intensity image'). $\nabla(I_r)$ is the gradient of the gray image in the right view (In Fig. 4.1, it is 'right gradient image' calculated from Sobel operator). $\alpha \geq 0$ weights the gradient value. $||\bullet||_1$ is L_1 distance. P'_G represents the distribution of the samples $\tilde{I}_r$ reconstructed from the left intensity image and corresponding refined disparity map. P_R represents the distribution of the samples I_r from the right camera in the stereo vision setting. The goal is to encourage disparity estimates at edges (larger gradients) to be more accurate with less reconstructed intensity error.

(3) The right intensity image is reconstructed from the left intensity image using the refined disparity output. Unlike [119], the reconstructed right intensity image and real right intensity image are input into the discriminator in this chapter, which also gives indirect feedback about whether the refined disparity distribution is close to the ground

truth. By making the discriminator output the probabilities at different receptive fields or scales (In Fig. 4.3, please refer to D1, D2,..,D5), the refiner will be forced to make the disparity distribution in the refined disparity map close to the real.

To alleviate training difficulties and avoid GAN mode collapse, the Improved WGAN loss function [62] is adopted. In Equation 4.3, D_i is the probability at the i^{th} scale that the input image patch to the discriminator is from the real distribution at the i^{th} scale. λ is the penalty coefficient. $\hat{I}_r$ is the random sample and $P_{\hat{I}_r}$ is its corresponding distribution. (For more details, please read [62]).

$$\begin{aligned}\mathcal{L}_{wgan}(G,D_i) = &\mathop{\mathbb{E}}_{\tilde{I}_r\sim P'_G}[D_i(\tilde{I}_r)] - \mathop{\mathbb{E}}_{I_r\sim P_R}[D_i(I_r)] \\ &+\lambda \mathop{\mathbb{E}}_{\hat{I}_r\sim P_{\hat{I}_r}}[(||\nabla_{\hat{I}_r} D_i(\hat{I}_r)||_2 - 1)^2]\end{aligned} \tag{4.3}$$

(4) After the strategies above, the neural network may produce outliers (black holes) in the texture-less areas. In order to suppress the outliers and noise in the refined disparity map, a gradient-based smoothness term is used to propagate more accurate disparity values to the areas with similar color by the assumption that the disparity in the neighborhood should be similar if the intensity is similar. However, this term tends to produce blurry edge in the refined disparity map.

$$\mathcal{L}_{sm}(G) = \mathop{\mathbb{E}}_{u\in\tilde{x},v\in N(u),\tilde{x}\sim P_G}[\exp(\gamma - \beta|\nabla(I_l)_{uv}|)||\tilde{x}_u - \tilde{x}_v||_1] \tag{4.4}$$

In Equation 4.4, $\tilde{x}_u$ is the disparity of a pixel u in the refined disparity map $\tilde{x}$ from the refiner. $\tilde{x}_v$ is the disparity value of a pixel v in the neighborhood $N(u)$ of pixel u. $\nabla(I_l)_{uv}$ is the gradient in the left intensity image (the refined disparity map is produced on the left view) from pixel u to pixel v. It is calculated from the left intensity image considering the diagonal, left and right directions. $\beta \geq 0$ and $\gamma \geq 0$ are responsible for how close the disparities are if the intensities in the neighborhood are similar.

(5) Finally, our object function for the fully unsupervised approach is:

$$\begin{aligned}G^* = \arg\min_G \max_{D_i} [&\theta_1 \mathcal{L}_{L_1}(G) + \theta_2 \mathcal{L}_{sm}(G) + \theta_3 \mathcal{L}_c(G) \\ &- \theta_4 \sum_{i=1}^{M} \mathcal{L}_{wgan}(G,D_i)]\end{aligned} \tag{4.5}$$

In Equation 4.5, M is the number of the scales (In Fig. 4.3, $M = 5$, please refer to D1, D2, .., D5). θ_1, θ_2, θ_3, θ_4 are the weights for the different loss terms.

4.2.3 Network architectures

A fully convolutional neural network [90] is adopted and also the partial architectures from [70, 119, 124] are adapted here for the refiner and discriminator. The refiner and discriminator use dense blocks [70] to increase local non-linearity. Transition layers [70] change the size of the feature maps to reduce the time and space complexity. In each dense block and transition layer, modules of the form ReLu-BatchNorm-convolution are used. Two modules in the refiner and four modules in the discriminator in each dense block are used, where the filter size is 3×3 and stride is 1. In each transition layer, only one module is used, where the filter size is 4×4 and the stride is 2 (except that in the third transition layer (Tran.3) of the discriminator the stride is 1).

Figure 4.2 shows the main architecture of the refiner. c1 initial disparity inputs (the experiments below use $c1 = 2$ for 2 disparity maps) and c2 pieces of information (the experiments below use $c2 = 3$ for the left intensity image, the right intensity image and a gradient of the right intensity image) are concatenated as input into the refiner. The batch size is b and resolution is $32m \times 32n$. lg is the number of the feature map channels after the first convolution. The refined disparity map is treated as a hidden layer in the network and used to reconstruct the intensity image in the right view. To reduce the computational complexity and increase the extraction ability of local details, each dense block contains only 2 internal layers. Additionally, the skip connections [126] from the previous layers to the latter layers preserve the local details in order not to lose information after the network bottleneck. During training, a dropout strategy has been added into the layers in the refiner after the bottleneck to avoid overfitting and the dropout part is cancelled during testing to produce a deterministic result.

Figure 4.3 is for the discriminator. The discriminator will only be used during training and abandoned during testing. The architecture of the discriminator will only influence the computational complexity during training. The initial disparity maps, information and real or reconstructed right images are concatenated and fed into the discriminator. Each dense block contains 4 internal layers. The sigmoid function (function tf.sigmoid in Tensorflow platform [9]) outputs the probability map ($D_i, i = 1,2..5$) that the input is real or fake at different scales to force the Markov Random Field of the refined disparity map to get closer to the real distribution at different receptive field sizes.

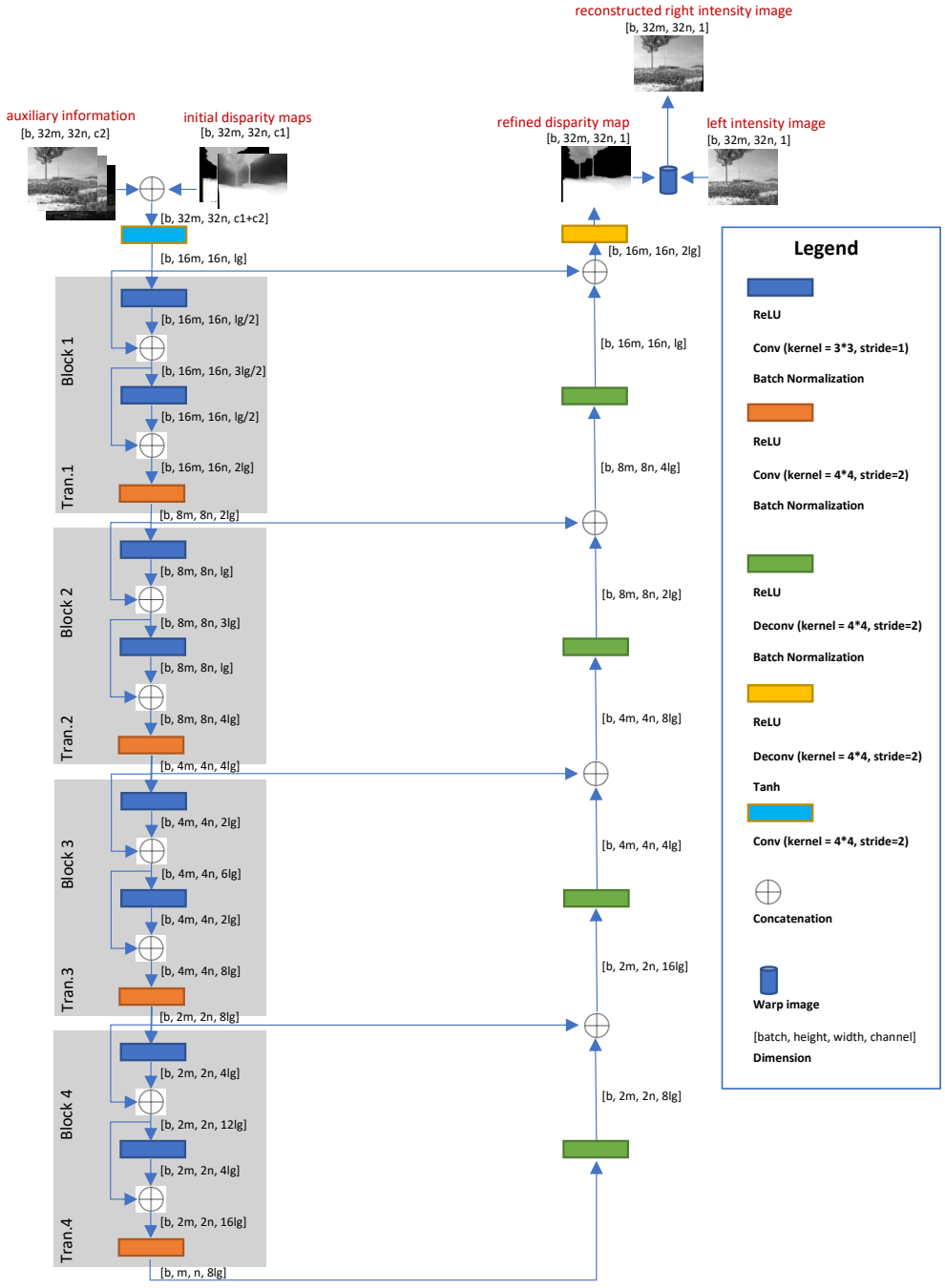


Figure 4.2: This figure shows some important hyperparameters and the refiner architecture configuration. Please refer to Table 4.1 for the specific values in each experiment.

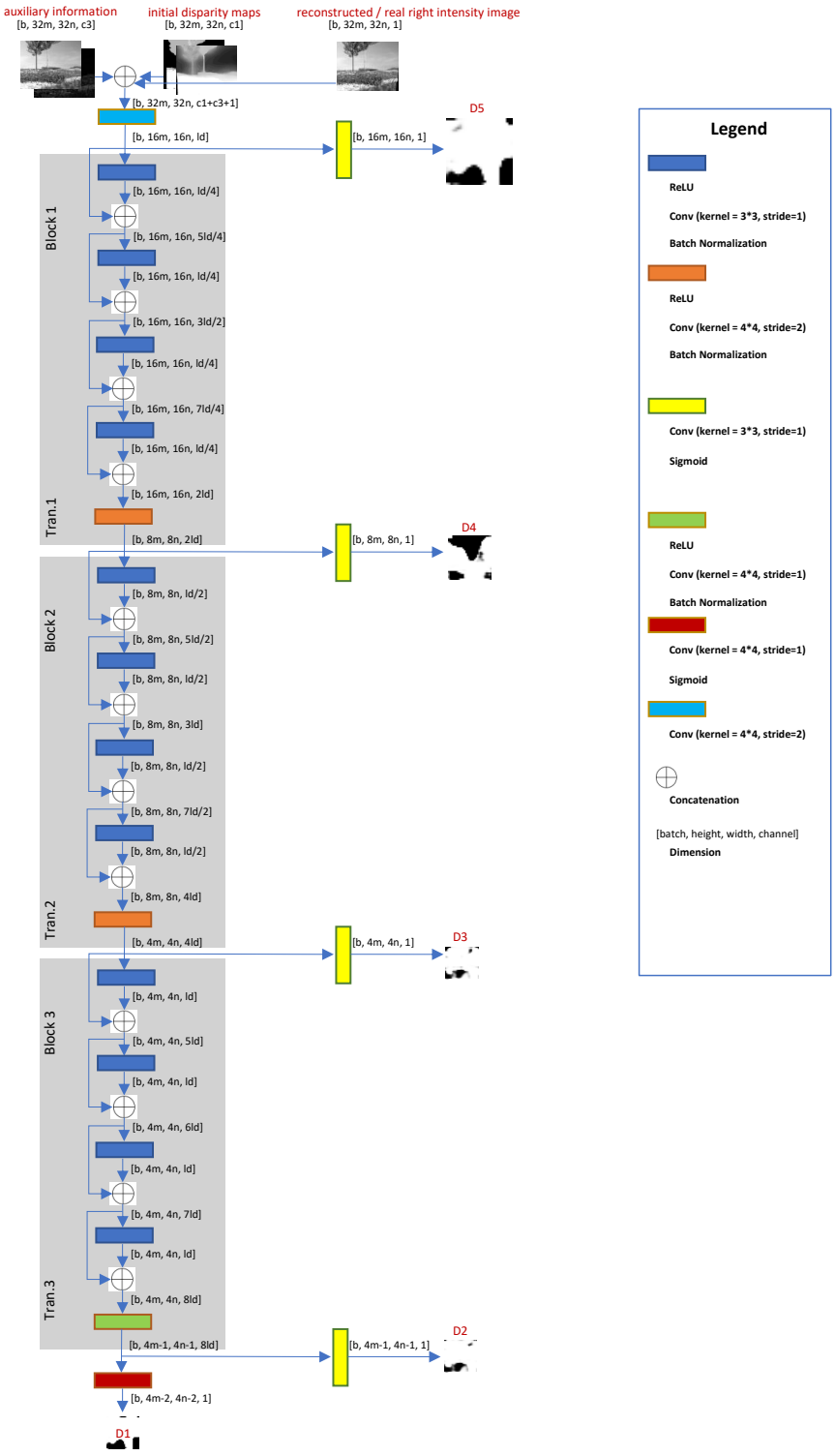


Figure 4.3: This figure shows some important hyperparameters and the discriminator architecture configuration. Please refer to Table 4.1 for the specific values in each experiment.

4.3 Experiments

The network is implemented using TensorFlow [13] and trained and tested using an Intel Core i7-7820HK processor (quad-core, 8MB cache, up to 4.4GHZ) and Nvidia Geforce GTX 1080Ti. First, an ablation study with initial disparity inputs [68, 97] is conducted using a synthetic garden dataset to analyze the influence of each factor in the energy function. Secondly, a real test on the Kitti2015 dataset [98] is done with two initial inputs [29, 67]. All the results show the proposed algorithm superiority compared with the state-of-art or classical stereo vision algorithms [29, 67, 68, 97], and the state-of-the-art stereo-stereo fusion algorithms [114, 119].

In the following experiments, the inputs to the neural network were first padded to $32M \times 32N$ (M, N are integers) using 0 and normalized to [-1, 1]. After that, the input was flipped vertically with a 50% chance to double the number of training samples. Weights of all the neurons were initialized from a Gaussian distribution (standard deviation 0.02, mean 0). Each model is trained for 100 epochs on a synthetic garden dataset[1] and 500 epochs on Kitti2015, with a batch size 4 using Adam [84] with a momentum of 0.5. The learning rate is changed from 0.005 to 0.0001 gradually. The method in [61] is used to optimize the refiner and discriminator by alternating between one step on the discriminator and one step on the refiner. The parameters θ_1, θ_2, θ_3, θ_4 in Equation 4.5 were set to make those four terms contribute differently to the energy function in the training process. If the difference of two initial disparity values on the same pixel is small (<0.3 pixels), a large value (0.99) is assigned to their confidence weight in Equation 4.1. If not, they were set to a medium value (0.5). Besides the confidence estimation above, some other special empirical confidence estimation for some disparity inputs were adopted in the following experiments (For more details, see the corresponding experiments). Additionally, any post-processing was not did on the occlusion area and the other areas. The L_1 distance between the estimated value and ground truth is used as the error. The unit is pixel. For more details about the network settings and computational complexity, please see Table 4.1. To highlight the real test, the network can do the disparity fusion (up to 384×1248 pixels) directly at 90 HZ without any cropping or downsampling.

Table 4.1: Computation Time and Initial Parameter Setting

	Ablation Study with Synthetic Garden Dataset														
Para.	Train time	Test time	b	32m	32n	c_1, c_2, c_3	lg	ld	θ_1	θ_2	θ_3	θ_4	α	β	γ
Value	0.22 (h/epoch)	0.010 (s/frame)	4	384	480	2, 3, 2	32	32	10	0.1	0.001	1	3	650	5
	Real Test with Kitti2015 Dataset														
Para.	Train time	Test time	b	32m	32n	c_1, c_2, c_3	lg	ld	θ_1	θ_2	θ_3	θ_4	α	β	γ
Value	0.03 (h/epoch)	0.011 (s/frame)	4	384	1248	2, 3, 2	10	10	1	20 to 1	0.0001 to 1K	1	3	1 to 3000	5

Table 4.2: Ablation Study on Each Cue (Unit: Pixel)

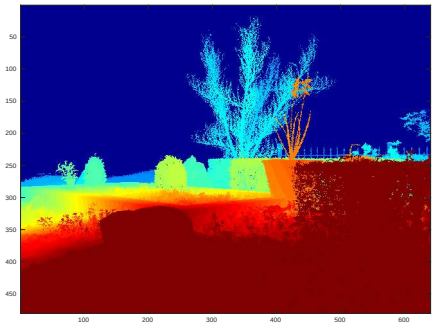
Experiment	Input1 [97]	Input2 [68]	$\theta_1 = 0$	$\theta_2 = 0$	$\theta_3 = 0$	$\theta_4 = 0$	$\alpha = 0$	$\beta = 65$	$\gamma = 1$	Baseline
Error (pixel)	4.67	8.52	3.53	3.94	**228.31**	3.70	3.55	5.86	3.67	**3.45**
Tuning experience	overall accuracy	overall accuracy	contrast influence	smooth influence	**initialized disparity**	disparity MRF	accuracy at edge	smooth effect	smooth effect	

4.3.1 Ablation Study

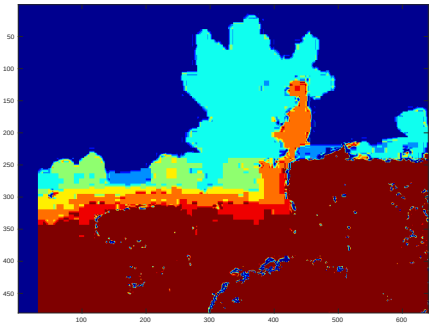
The synthetic garden dataset contains 4600 training samples and 421 test samples under outdoor environments. Each sample has one pair of rectified stereo images and dense ground truth with resolution 480×640 ($height \times width$) pixels. The reason using a synthetic dataset is that the real dataset (eg: Kitti2015 [98]) does not have dense ground truth, which will influence the evaluation of the network. Dispnet [97] and FPGA-stereo [68] were used as inputs. The authors of [68, 97] helped us get the best performance on the dataset as the input to the network. Besides the confidence estimation strategy above, another special rule to Dispnet was added because the disparity map from Dispnet is very inaccurate in remote areas. That is, the pixels whose disparity is less than 4 (pixels) have a small confidence (0.1) and, otherwise, have a big confidence (0.9) because it is more accurate for close scenes. The default settings for the baseline network is shown in Table 4.1. In the ablation study, one of the following factors (θ_1, θ_2, θ_3, θ_4, α, β, γ) in our energy function is changed to see the influence of each cue. The performance results are listed in Table 4.2. One example is Figure 4.4. As said in the introduction, the disparity constraint term θ_3 (Equation 4.1) encourages the network to produce disparities close to the initial disparity inputs. It is the most important factor (228.31 pixels errors) because the other factors do not provide an accurate global initialization but mainly refine the disparity value in the local area using

[1] It is not available to the public now.

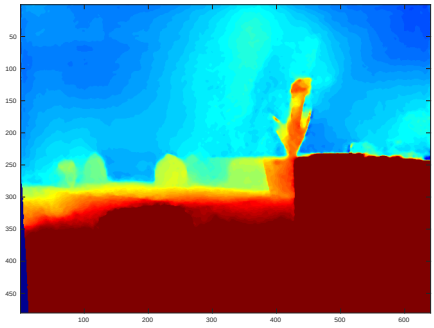
(a) Scene



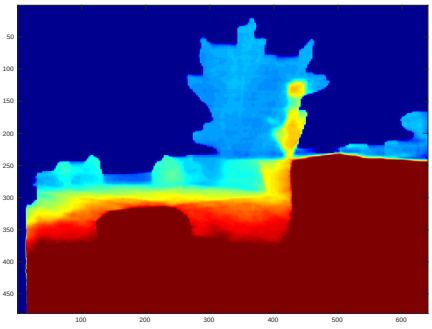
(b) Ground Truth



(c) FPGA Stereo [68]



(d) Dispnet [97]



(e) Fused Disparity

Figure 4.4: Two initial disparity inputs (c)(d) were fused to get a refined disparity map (e) using our baseline method on the synthetic garden dataset. (b) is the ground truth and (a) is the corresponding scene. Many, but not all, pixels from the fused result are closer to the ground truth than the original inputs.

pixel-pixel intensity (θ_1,α), smoothness (θ_2, γ, β) and Markov Random Field (θ_4) . When tuning the network, the experience also followed the theory in the methodology

section (Table 4.2).

To explore the influence of the input accuracy and the number of initial inputs, one or two initial disparity maps with different noise levels were input into the networks. The disparity inputs were obtained by adding different levels of Gaussian noise $N(0,\sigma^2)$ to the normalized ground truth (ground truth divided by 480). The confidence for each pixel in the experiments is proportional to the noise value under the assumption that accurate confidence estimation in the future can be obtained. The fusion accuracy (Table 4.3) increases with the number and accuracy of the disparity inputs. When there is only one input with $\sigma = 0.002$, the proposed method still can refine the input disparity (0.77 <1 pixel) because there are accurate confidence estimates. The proposed method sometimes fails to refine only one input disparity map when there are no confidence estimates (all confidences are 100%) and the mean error (<1 pixel) is too small. Otherwise, it works robustly and effectively.

Table 4.3: Average error (pixel) on Synthetic Garden

	Fuse one input		**Fuse two inputs**		
Noise std.	**input1**	**fused**	**input1**	**input2**	**fused**
$\sigma = 0.002$	0.77	0.69	0.77	0.77	0.64
$\sigma = 0.004$	1.53	1.13	1.53	1.53	0.71
$\sigma = 0.008$	3.06	1.58	3.06	3.06	0.80
$\sigma = 0.016$	6.12	3.05	6.12	6.12	1.33

4.3.2 Real Data

The stereo-stereo and stereo-Lidar depth fusion have been done as follows.

4.3.2.1 Stereo-stereo Fusion

The network was tested on the real Kitti2015 dataset [98], which used a Velodyne HDL-64E Lidar scanner to get the sparse ground truth and a 1242×375 resolution stereo camera to get stereo image pairs. The training dataset contains 400 unlabeled and labeled samples in all. There are another 400 samples in the test dataset. 50 samples from '000000_10.png' to '000049_10.png' in the Kitti2015 training dataset were used as our test dataset. 50 samples from '000050_50.png' to '000099_10.png' in the Kitti2015 training dataset were used as our validation dataset. The rest 700 samples were used

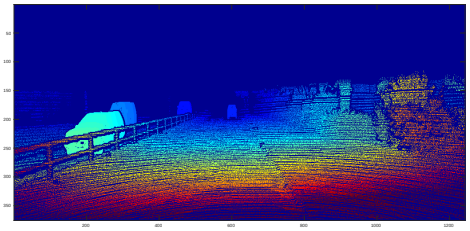
as our training set. By flipping the training samples vertically, it doubled the number of training samples. The state-of-art stereo vision algorithm PSMNet [29] was used as one of our inputs. Their released pre-trained model[2] on the Kitti2015 dataset was used to get the disparity maps. A traditional stereo vision algorithm SGM [67] was used as the second input to the network. Given that the sparsity of SGM is not so important, its parameters were set to produce more reliable disparity maps. Big confidence values (0.8) were assigned to its valid pixels and 0 to its invalid pixel confidences. More specifically, the implementation ('disparity' function [3]) from Matlab2016b was used. The relevant parameters are: 'DisparityRange' [0, 160], 'BlockSize' 5, 'ContrastThreshold' 0.99, 'UniquenessThreshold' 70, 'DistanceThreshold' 2. The settings of the neural network are shown in Table 4.1. The proposed algorithm was compared with the state-of-the-art technique [114, 119] in stereo-stereo fusion and also stereo vision inputs [29, 67]. The proposed method was compared with the supervised model in Sdf-MAN [119]. The supervised model in Sdf-MAN was trained on our synthetic garden dataset first and fine-tuned on the Kitti2015. 150 labeled samples from '00050_10.png' to '000199_10.png' in the initial training dataset were used for Sdf-MAN fine-tuning. Compared with SGM [67] (0.78 pixels) (This is a more accurate disparity but is calculated only using more reliable pixels. On average only 40% of the ground truth pixels are used. If all the valid ground truth are used to calculate its error, it is 22.13 pixels), the fused results are much more dense and accurate. The performance of the proposed algorithm (0.83 pixels) (See Table 4.4) is better than Sdf-MAN (1.17 pixels). The reason is because Sdf-MAN can not generalize in the real environment well. However, the proposed algorithm is not affected by such problems because the unsupervised method can use the unlabeled data directly.

Table 4.4: Average error (pixel) on Kitti2015

Source Error		**Fused Algorithm Error**		
SGM [67]	PSMNet [29]	DSF [114]	Sdf-MAN [119]	Ours
0.78	1.22	1.20	1.17	0.83

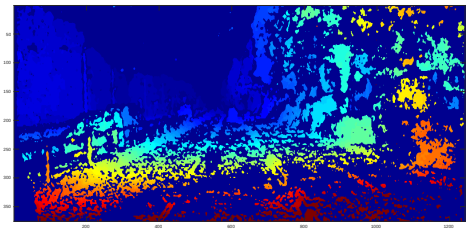
For qualitative results, see Figure 4.5. The proposed method compensates for the weaknesses of the inputs and refines the initial disparity maps effectively. Compared with SGM [67], the fused results are much more dense and accurate. Compared with PSMNet, the proposed method preserves the details better (eg: mountain). The proposed

[2]PSMNet [29]: `https://github.com/JiaRenChang/PSMNet`

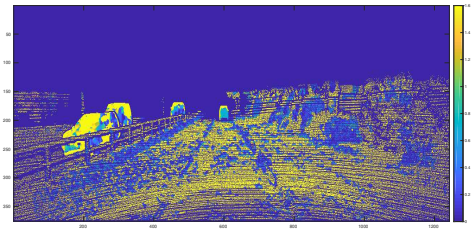


(a) Ground Truth

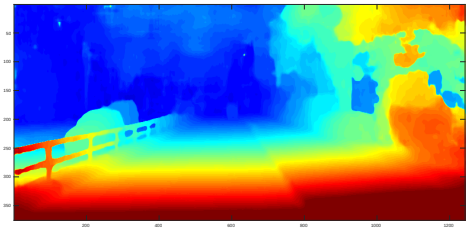
(b) Scene



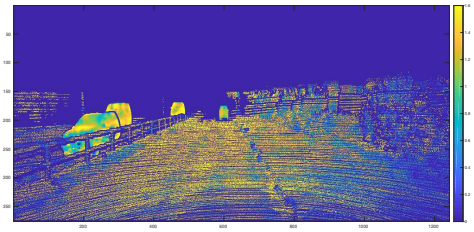
(c) Input Disparity 1: SGM [67]



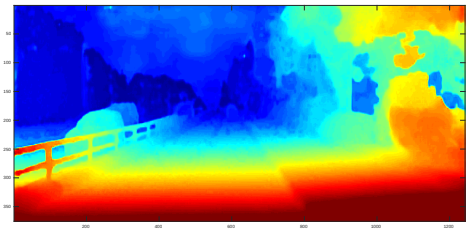
(d) Input Disparity 1 Error: SGM [67]



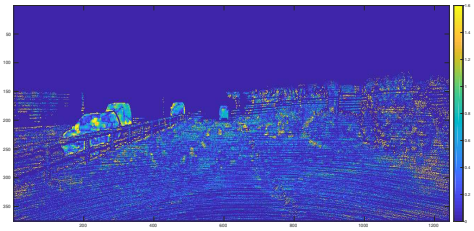
(e) Input Disparity 2: PSMNet [29]



(f) Input Disparity 2 Error: PSMNet [29]



(g) Refined Disparity



(h) Refined Disparity Error

Figure 4.5: The unsupervised adversarial network was trained to fuse the initial disparity maps (c), (e) into a refined disparity map (g) for the same scene (b) from the Kitti2015 dataset [98]. (a) is the corresponding ground truth. (d), (f), (h) are the errors of (c), (e), (g). The colorbars (from blue to white) corresponds to 0 - 1.6 pixels and the lighter pixel have bigger error in (d), (f), (h).

method fully fails in the sky. The reason is that the pixels (disparity = 0) are treated as invalid (confidence = 0) in SGM. However, the disparity values of the pixels in the sky area from PSMNet are all larger than 0 (confidence >0). The PSMNet misleads the network to adopt their disparity value as the initialization. The wrong confidence measurement can bring big error to the refined disparity map. The problem can be solved by adding more cues, such as semantic meaning, to make the confidence measurement more accurate.

The efficiency and effectiveness of the proposed network structures on the Kitti2015 dataset have been explored. The same settings from Table 4.1 except the channel (l_g, l_d) of the features in each layer were used. The results are shown in Table 4.5. When $l_g = l_d = 10$, our network achieves the best accuracy (0.83 pixels) compared with the rest. Additionally, the four groups of experiments with different l_g, l_d achieve similar performance, which shows the robustness of the proposed method.

Table 4.5: Average error (pixel) on Kitti2015

Structure	Error (pixel)	Time (s/frame)
$l_g = l_d = 5$	1.00	**0.009**
$l_g = l_d = 10$	**0.83**	0.011
$l_g = l_d = 15$	0.92	0.014
$l_g = l_d = 20$	0.87	0.017

4.3.2.2 Stereo-Lidar Fusion

Our network can generalize to any sub-fusion task based on left-right consistency. A stereo-lidar fusion demo on Kitti2015 has also been done. The valid points in the initial ground truth were removed by half randomly to get the Lidar input to our network. The confidence of each valid Lidar data point was set as extremely large (100%). PSMNet was used as the stereo input again and set the confidence of each pixel as 50%. One hundred labeled images in the initial Kitti2015 training dataset were used to train (from '000100_10.png' to '000199_10.png') and the other 100 to test (from '000000_10.png' to '000099_10.png'). The error of PSMNet [29] is 1.22 pixels and our error is 0.86 pixels after fusion.

4.4 Conclusion and Discussion

This chapter proposed an unsupervised method to fuse the disparity estimates of multiple state-of-art disparity/depth algorithms. The experiments have shown the effectiveness of the energy function design based on multiple cues and the efficiency of the network structure. The proposed network can be generalised to other fusion tasks based on left-right image consistency (only stereo-stereo and stereo-Lidar fusion were used in this chapter). The work in this chapter reduces the cost of acquiring labelled data necessary for use in a supervised method (eg: [14, 109, 114, 119]). All the experimental results in this chapter show the proposed algorithm has better performance compared with the state-of-art stereo vision algorithms [29, 67, 68, 97], and the state-of-the-art stereo-stereo fusion algorithms [114, 119].

Additionally, when the initial disparity maps were from SGM [67] and Dispnet [97], which was trained on Scene Flow dataset [97] only, the proposed unsupervised depth fusion method was still able to refine them effectively on Synthetic Garden dataset, whose ground truth was not seen by SGM, Dispnet and the proposed method in advance. Future work will explore the influence from the training datasets used by the initial depth estimation methods and the proposed unsupervised depth fusion method.

Given the low computation cost of the algorithm, the combination of the proposed method and existing depth-acquisition algorithms is a good solution to obtaining high accuracy depth maps. The supervised & semi-supervised model could be used in Chapter 3, and the unsupervised disparity fusion could be used in this chapter to refine the disparity maps from different sources. The disparity map could be converted into the 3D point cloud. In the next chapter, how to align the point clouds from different views to obtain the relative 6D pose will be introduced.

Chapter 5

Rigid Point Cloud Registration

The point clouds generated using the methods from the previous two chapters can not avoid having noise. Accurately registering noisy point clouds is a challenging task. Existing rigid registration methods fail to use the physical 3D uncertainty distribution of each point from a real sensor in the dynamic alignment process. It is mainly because the uncertainty model for a point is static and invariant and it is hard to describe the change of these physical uncertainty models in different views. Additionally, the existing Gaussian mixture alignment architecture cannot efficiently implement these dynamic changes.

This chapter proposes a simple but more effective architecture combining error estimation from sample covariances and dynamic global probability alignment using the convolution of uncertainty-based Gaussian Mixture Models (GMM). Firstly, an effective way is proposed to describe the change of each 3D uncertainty model, which represents the structure of the point cloud better. Unlike the invariant GMM (representing the fixed point cloud) in the traditional Gaussian mixture alignment, here two uncertainty-based GMM that change and interact with each other in each iteration are used. In order to have a wider basin of convergence than other local algorithms, a more robust energy function is designed by convolving the two GMM over the whole 3D space efficiently.

Tens of thousands of trials have been conducted on hundreds of models from multiple datasets to demonstrate the superior performance of the proposed method compared with the current state-of-the-art methods [28, 76, 86, 103, 156, 162, 169]. All the materials including our code are available from `https://github.com/Canpu999/DUGMA`. The results presented in this chapter were achieved during November 2017. The majority of the content in this chapter is from our paper [118].

5.1 Introduction

With recent improvements in depth sensing devices and algorithms, 3D point clouds are more accessible to researchers. However, using an expensive high-precision 3D scanner to get accurate and large-scale 3D point clouds is still not popular. Accurate alignment of noisy and partial point clouds with many outliers from cheap low-resolution sensors is still a core technique in various fields, such as computer vision, robotics, virtual and augmented reality.

Finding the accurate transformation between two noisy rigid point clouds is generally hard: true point-to-point correspondences seldom exist, which limits the accuracy of the methods based on ICP [23, 34, 54, 116, 134, 156, 169]. As for the methods based on local descriptors [86, 127, 139, 162], coarse points tend to cause inaccurate local descriptors to mismatch with each other. Also, the variable density (distant areas have a lower point cloud density) will make them unstable. Aligning probabilistic models can effectively mitigate the problems above. Many researchers have been exploring different kinds of probabilistic models [27, 76, 103, 138, 142, 167] to represent the real surface structure.

However, as far as we can tell, no one has incorporated the physical 3D uncertainty distribution information for each point from a real sensor into the probabilistic model, which allows describing the real surface structure more accurately. The challenges mainly lie in two parts: the first is how to get the real uncertainty distribution information from the real sensors. In the recent years, an increasing number of researchers have been investigating how to estimate the uncertainty of the acquired data for different sensors, such as the Kinect sensor [105], the time of flight sensor [39], the structure from motion sensor [51] and the stereo vision sensor [95]. These suggest using physical noise models for each point to represent their individual occurrence probability in 3D space. The second challenge is how to use physical uncertainty information for each point from each specific view in the registration process. Specifically, if we use a 3D Gaussian probability model with a covariance to represent the uncertainty of each point in the 3D space, the covariance should change with each different coordinate system in each iteration. The registration process is dynamic. Moreover, the use of the covariance estimated from different viewpoints leads to position estimates that are compatible with each physical covariance. After that, we build a bridge to make two point clouds interact with each other by sharing information, which is obviously different from traditional Gaussian mixture alignment [28, 76]. The GMM of the fixed point cloud

in their methods is invariant and cannot share their state with the GMM of the moving point cloud, reducing the registration accuracy and also making the usage of physical uncertainty models unavailable.

In this chapter, we propose a simple architecture combining error estimation from sample covariances and dual dynamic global probability alignment using the convolution of uncertainty-based GMM from point clouds in the whole 3D space. Firstly, we propose an effective way to describe the change of each 3D uncertainty model in the dynamic registration process, which represents the structure of the point cloud much better. Unlike the invariant GMM (representing the fixed point cloud) in traditional Gaussian mixture alignment, we use a dynamic uncertainty-based GMM for each point set, which interact in each iteration. To be less susceptible to local minimum, we define a more robust energy function by convolving the two dynamic GMM over the whole 3D space rather than use time-consuming optimization methods, such as branch and bound [28, 138, 156]. The proposed dual dynamic uncertainty-based GMM's alignment can be optimized efficiently by the EM algorithm [24, 40], which experimentally shows a wider basin of convergence than other local algorithms. A new empirical approximation will be proposed to reduce the amount of calculation drastically.

The rest of this chapter is organized as the following. In Section 5.2, the dynamic uncertainty-based Gaussian mixture alignment theory is presented. In Section 5.3, tens of thousands of trials have been conducted on multiple datasets through simulation, which is more systematic testing than that done for the compared algorithms. Also, we show real application tests with most recent and advanced algorithms.

The main contributions presented in this chapter are (Section 5.3 gives a more detailed comparison with [28, 76, 86, 103, 156, 162, 169]):

- Incorporation of the invariant 3D uncertainty distribution information (represented by a 3D Gaussian distribution with a physical covariance) into the dynamic registration;
- A bridge to make the two point clouds interact with each other by creating a novel point proximity weight term;
- A more robust energy function and an efficient approximation to the optimization step that greatly reduces computational complexity.

5.2 Methodology

First we introduce the change of 3D uncertainty distribution, then build a bidirectional dynamic bridge between the two point clouds, and finally introduce the framework of our math model. Table 5.1 lists some of the symbols and their notations.

Table 5.1: Symbols & Notations

Symbol	Notation
$\mathbf{X}, \mathbf{Y}$	Two point clouds
D	Dimensionality of the point clouds
N, M	Number of points in X, Y point cloud
x_n	One point in X point cloud
y_m	One point in Y point cloud
Σ_{x_n}, Σ_{y_m}	Covariance for point x_n and y_m
$\mathbf{I}$	Identity matrix
$\mathbf{0}$	Column vector of all zeros

5.2.1 Change of 3D Uncertainty Distribution

We will use a Gaussian function with a covariance to represent the distribution of one point in 3D space. The specific covariance for each point represents the physical 3D uncertainty distribution for that point from a real sensor.

(1) If a point with covariance Σ_{orig} has been rotated by $\mathbf{R}$, Σ will be $\Sigma = \mathbf{R}\Sigma_{orig}\mathbf{R}'$.

(2) A scaling factor for the covariance of a point is proportional to the average minimum distance σ between two point clouds to ensure that the probability of all the points in the other point cloud will not become too small when the two point clouds are far away from each other. See Algorithm 2.

5.2.2 Dynamic Gaussian Mixture Alignment

The Gaussian function $g_{x_n}(\tau)$ of the point x_n predicts the probability that x_n appears at the position τ in its own coordinate system. Based on Gaussian weights around each point, we will define a probability-like function that not only depends on the distribution of the point (represented by isotropic or anisotropic covariance) but also whether a possible corresponding point cp_{x_n} in the other point cloud is nearby. We

model the presence of a corresponding point by a weight function $w_{x_n}(\tau, cp_{x_n})$ that has significant value only when a potential corresponding point cp_{x_n} from point cloud $\mathbf{Y}$ is near the position τ. A similar definition holds for $g_{y_m}(\tau)$, $w_{y_m}(\tau, cp_{y_m})$. Either point cloud can receive and send current state information from or to the other point cloud bi-directionally to evaluate the current registration quality.

In the analysis below, we assume the $\mathbf{Y}$ point cloud has been already transformed from the initial point cloud $\mathbf{Y_0}$ by rotation $\mathbf{R}$ and translation $\mathbf{t}$ (which become the domain for the optimization of the evaluation function). The product $g_{x_n}(\tau)g_{y_m}(\tau)$ represents x_n, y_m appearing at the same position τ in the same coordinate system. It encodes the underlying prior knowledge, ie. x_n, y_m are possible corresponding points from two point clouds. In other words, any two points from the fixed and moving point cloud can be a corresponding pair in our system and the likelihood depends on the probability of correspondence that x_n, y_n appear at the same position τ, which is different from soft assignment [60] in essence.

5.2.3 The Description of Our Model

Based on the previous discussion, we design the uncertainty-based GMM as follows:

$$G_{\mathbf{X}}^{\mathbf{I},\mathbf{0}}(\tau) = \sum_{n=1}^{N} w_{x_n}(\tau, cp_{x_n}) g_{x_n}(\tau), \tag{5.1}$$

$$G_{\mathbf{Y}}^{\mathbf{R},\mathbf{t}}(\tau) = \sum_{m=1}^{M} w_{y_m}(\tau, cp_{y_m}) g_{y_m}(\tau), \tag{5.2}$$

where the gaussian kernels are given by

$$g_{x_n}(\tau) = \frac{1}{\sqrt{(2\pi)^D |\Sigma_{x_n}|}} e^{-\frac{1}{2}(\tau - x_n)^T \Sigma_{x_n}^{-1} (\tau - x_n)} \tag{5.3}$$

$$g_{y_m}(\tau) = \frac{1}{\sqrt{(2\pi)^D |\Sigma_{y_m}|}} e^{-\frac{1}{2}(\tau - y_m)^T \Sigma_{y_m}^{-1} (\tau - y_m)} \tag{5.4}$$

$$w_{x_n}(\tau, cp_{x_n}) = e^{-\frac{1}{2}(cp_{x_n} - \tau)^T \Sigma_{x_n}^{-1} (cp_{x_n} - \tau)} \tag{5.5}$$

$$w_{y_m}(\tau, cp_{y_m}) = e^{-\frac{1}{2}(cp_{y_m} - \tau)^T \Sigma_{y_m}^{-1} (cp_{y_m} - \tau)} \tag{5.6}$$

$G_{\mathbf{X}}^{\mathbf{I},\mathbf{0}}(\tau)$ denotes the GMM from the fixed point cloud $\mathbf{X}$ and $G_{\mathbf{Y}}^{\mathbf{R},\mathbf{t}}(\tau)$ represents the GMM from the moving point cloud $\mathbf{Y}$ after rotation $\mathbf{R}$ and translation $\mathbf{t}$. Thus $y_m = \mathbf{R}y_{m0} + \mathbf{t}$, $\Sigma_{y_m} = \mathbf{R}\Sigma_{y_{m0}}\mathbf{R}'$, $|\Sigma_{y_m}| = |\mathbf{R}\Sigma_{y_{m0}}\mathbf{R}'| = |\Sigma_{y_{m0}}|$ due to $|\mathbf{R}| = 1$. $\Sigma_{y_m}^{-1} = (\mathbf{R}\Sigma_{y_{m0}}\mathbf{R}')^{-1} = \mathbf{R}\Sigma_{y_{m0}}^{-1}\mathbf{R}'$ due to $\mathbf{R}\mathbf{R}' = \mathbf{I}$. At all times, $x_n = x_{n0}$ and $\Sigma_{x_n} = \Sigma_{x_{n0}}$. Each point has its own covariance.

Integrating the product of the two dynamic GMM (representing the overlapping effect of the two point clouds) over the whole 3D space, as we shall show in our experiments, makes the energy function more robust, accurate and have a wider convergence basin compared with [28, 76, 103].

We now formulate the optimization over rotation $\mathbf{R}$ and translation $\mathbf{t}$ as an EM-like process. First, an energy function is defined as the following

$$E = \int P(\tau) \log \left[G_{\mathbf{X}}^{\mathbf{I},\mathbf{0}}(\tau) G_{\mathbf{Y}}^{\mathbf{R},\mathbf{t}}(\tau) \right] \mathrm{d}\tau, \tag{5.7}$$

where τ integrates over all the domain of the point clouds; $P(\tau)$ is the probability that there is a point at the position τ. We design it as the sum of the probability that all the possible corresponding pairs appear at the position τ in

$$P(\tau) = \sum_{i=1}^{N} \sum_{j=1}^{M} P(\tau, x_i, y_j) = \sum_{i=1}^{N} \sum_{j=1}^{M} g_{x_i}(\tau) g_{y_j}(\tau). \tag{5.8}$$

Equation (5.7) can be rewritten as

$$E = \int P(\tau) \log \Big[\sum_{i=1}^{N} \sum_{j=1}^{M} F^{\mathbf{R},\mathbf{t}}(\tau, x_i, y_j) \Big] \mathrm{d}\tau, \tag{5.9}$$

with a combined term

$$F^{\mathbf{R},\mathbf{t}}(\tau, x_i, y_j) = w_{x_i}(\tau, cp_{x_i}) g_{x_i}(\tau) w_{y_j}(\tau, cp_{y_j}) g_{y_j}(\tau). \tag{5.10}$$

By the definitions above, the weight term $w_{x_i}(\tau, cp_{x_i})$ is nearly zero when point x_i is far from any point in $\{y_j\}$. This allows us to avoid having to compute correspondences by using all y_j in place of cp_{x_i} (and similarly for cp_{y_j}) and simplify $F^{\mathbf{R},\mathbf{t}}$ with

$$\tilde{F}(\tau, x_i, y_j) = w_{x_i}(\tau, y_j) g_{x_i}(\tau) w_{y_j}(\tau, x_i) g_{y_j}(\tau). \tag{5.11}$$

We maximize Equation (5.9) to get the estimated rotation and translation by minimizing its negative. We adopt the EM algorithm [24, 40] to solve for $\mathbf{R}$, $\mathbf{t}$. The main idea is: guess the values of the parameters firstly in the last iteration (denoted by 'old') and calculate the posteriori probability $P^{old}(x_i, y_j|\tau)$ using Bayes' theorem, which corresponds to the expectation stage. After that, minimize the expectation of the negative log-likelihood function $\mathcal{L}$ to find the parameters in the current iteration (denoted by 'new'), which corresponds to the maximization stage. We get

$$\mathcal{L} = -\int \sum_{i=1}^{N} \sum_{j=1}^{M} B(\tau, x_i, y_j) \log(\tilde{F}^{new}(\tau, x_i, y_j)) \, \mathrm{d}\tau, \tag{5.12}$$

$$B(\tau, x_i, y_j) = P(\tau) P^{old}(x_i, y_j \mid \tau). \tag{5.13}$$

Neglecting the constant term and using $P(\tau) \approx P^{old}(\tau)$, we simplify the target function to

$$\mathcal{L} = \sum_{i=1}^{N} \sum_{j=1}^{M} \int P^{old}(\tau, x_i, y_j)\, Mah^{new}(\tau, x_i, y_j)\, \mathrm{d}\tau, \tag{5.14}$$

where a term similar to Mahalanobis distance is obtained:

$$\begin{aligned} Mah^{new}(\tau, x_i, y_j) = &\frac{1}{2}(\tau - x_i)^T (\Sigma_{x_i}^{-1} + \Sigma_{y_j}^{-1})(\tau - x_i) \\ &+ \frac{1}{2}(\tau - y_j)^T (\Sigma_{x_i}^{-1} + \Sigma_{y_j}^{-1})(\tau - y_j). \end{aligned} \tag{5.15}$$

As we will justify below, there is no real benefit to integrate the whole 3D space, because the values of the Gaussian functions are only significant near the data points themselves. In fact, because most values are quite low, we approximate the integral by a sum at only the data points to speed up the algorithm drastically (unlike [28]). We need evaluate only each term $P^{old}(\tau, x_i, y_j) Mah^{new}(\tau, x_i, y_j)$ at the positions of x_i and y_j, which will reduce the time complexity greatly to only $O(MN)$. Applying this simplification, the approximated energy function becomes

$$\tilde{\mathcal{L}} = \sum_{i=1}^{N} \sum_{j=1}^{M} \sum_{\tau \in \{x_i, y_j\}} P^{old}(\tau, x_i, y_j)\, Mah^{new}(\tau, x_i, y_j). \tag{5.16}$$

Expanding the last sum and uniting like terms we get

$$\tilde{\mathcal{L}} = \sum_{i=1}^{N} \sum_{j=1}^{M} C_{i,j}^{old} \cdot (y_j - x_i)^T (\Sigma_{x_i}^{-1} + \Sigma_{y_j}^{-1})(y_j - x_i), \tag{5.17}$$

where

$$\begin{aligned} C_{i,j}^{old} = &(2\pi)^{-D} |\Sigma_{x_i}|^{-\frac{1}{2}} |\Sigma_{y_j}|^{-\frac{1}{2}} \\ &\left(e^{-\frac{1}{2}(y_j - x_i)^T \Sigma_{x_i}^{-1} (y_j - x_i)} + e^{-\frac{1}{2}(x_i - y_j)^T \Sigma_{y_j}^{-1} (x_i - y_j)}\right). \end{aligned} \tag{5.18}$$

The x_i, y_j, Σ_{x_i} and Σ_{y_j} in $C_{i,j}^{old}$ are from the previous iteration. We minimize Equation (5.17) over the rotation $\mathbf{R}$ and translation $\mathbf{t}$ domain, using interior point optimization [154] as summarized in Algorithm 2.

Algorithm 2 DUGMA Point Cloud Registration (See Section 5.2 for details).

Input: Two point clouds $\mathbf{X}, \mathbf{Y}$ and their covariances Σ_x, Σ_y, initial transformation $\mathbf{R} = \mathbf{I}$, $\mathbf{t} = \mathbf{0}$.

Output: Estimated rotation $\mathbf{R}$ and translation $\mathbf{t}$.

EM-like optimization, repeat until convergence:
procedure E-STEP ▷ *Update* $\mathbf{Y}$, σ, Σ_y, $C_{i,j}^{old}$
  $\mathbf{Y} \leftarrow \mathbf{R}\mathbf{Y} + \mathbf{t}$
  $\sigma \leftarrow \frac{1}{M} \sum_{j=1}^{M} d_{min}(y_j, \mathbf{X})$ ▷ *Minimum distance*
  $\Sigma_y \leftarrow \sigma\, \mathbf{R}\Sigma_y\mathbf{R}'$
  $C_{i,j}^{old} \leftarrow$ compute Eq. (5.18)
procedure M-STEP ▷ *Solve for* $\mathbf{R}$,$\mathbf{t}$
  Use interior point algorithm to solve Eq. (5.17):
  $(\mathbf{R}, \mathbf{t}) \leftarrow \arg\min_{\mathbf{R},\mathbf{t}} \tilde{\mathcal{L}}$

5.3 Experiments

We implemented our algorithm using Matlab and Cuda C++. We ran all the algorithms on a laptop with Intel Core i7-7820HK processor (quad-core, 8MB cache, up to 4.4GHZ) and NVidia Geforce GTX 1080 with 8GB GDDR5X. To test the accuracy and robustness of our algorithm, our proposed method is compared with relevant recent algorithms from the top journals and conferences: CPD [103], GMMREG [76], BDICP [169], GOICP [156], GOGMA [28], 3DMATCH [162][1], FDCP [86][2]. All the code is directly from the authors. We did not compare ours with [47] because we could not get our re-implemented algorithm to work well based on their partial released code. The Stanford 3D Scanning Repository [87] and our new 3D dataset have been used to do performance comparison of the algorithms. After that, 30 real scenes with ground truth from multiple Kinect sensors in our new dataset[3] have been used to show the approach works on par or better in a real application compared with the rest.

[1] We only compared ours with 3DMATCH in the Kinect data application with their pre-trained weights.

[2] FDCP has compared their results with [138] so we neglected [138].

[3] https://github.com/Canpu999/DUGMA

5.3.1 Simulation

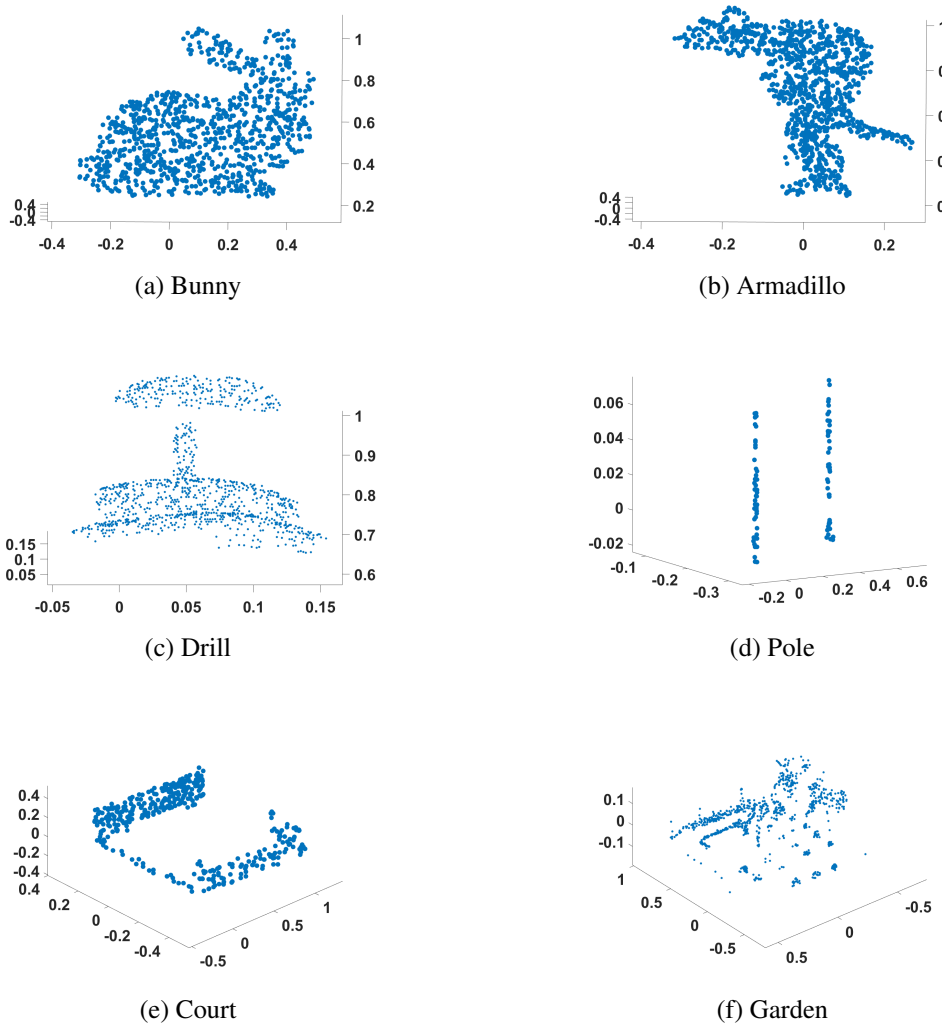
To synthesize the two point clouds to register, we randomly choose a model from the datasets above for two point clouds firstly. A different random large segment of each point cloud is removed completely to simulate occlusion. After that, the two occluded models are sampled differently, which simulates the resolution of different sensors in real scenarios. Also, different anisotropic Gaussian noise with random standard deviations and zero mean has been added to each point to simulate the complex noise in real environments resulting from known and unknown factors. The variances of all the noise on each axis have been stored in the covariances accurately. Next, outliers have been added into both point clouds to simulate outliers acquired by the sensors. Finally, an initial rigid transformation is applied to the moving point cloud.

The experiments are divided into four groups given the four influence factors or variables: noise, outliers, occlusion, and initial rotation. In each group of experiments, one controlled variable will be changed and the values of the other variables will be picked randomly from a default range. The experiment is conducted 3 times at each controlled value for each of 100 shapes with a random perturbation each time, see Algorithm 3. The maximum iteration value for all is 100. For FDCP, we set `gridStep` $= 1.5$ and `Rho` $= 0.1$ to make it robust to different densities. For GOGMA we set the scale parameter for SVM (0.5,0.5) to limit GOGMA running time to around 100 seconds per registration. For GOICP, Mean Squared Error (MSE) convergence threshold `MSEThresh`=0.2. The rest of the parameters share default values from their original implementations. We use $||\mathbf{t_{gt}} - \mathbf{t_{est}}||_F$, $||\mathbf{I} - \mathbf{R_{gt}}\mathbf{R_{est}^{-1}}||_F$ [72] to estimate the quality of the registration, where $\mathbf{R_{gt}}, \mathbf{t_{gt}}$ are the ground truth and $\mathbf{R_{est}}, \mathbf{t_{est}}$ are estimated results respectively and $||\bullet||_F$ is the Frobenius norm.

Algorithm 3 Controlled and random variables process. For each method, 14700 trials have been done.

```
for controlled_variable := start by step to end do
    for shape := 1 to 100 do
        for instance := 1 to 3 do
            Produce data with controlled and random variable
            Do registration (different algorithms)
            Calculate the registration error
```

From the Stanford 3D Scanning Repository [87] (50) and our new dataset (50) we got 100 models from various views of different objects and scenes. Each was



(a) Bunny (b) Armadillo

(c) Drill (d) Pole

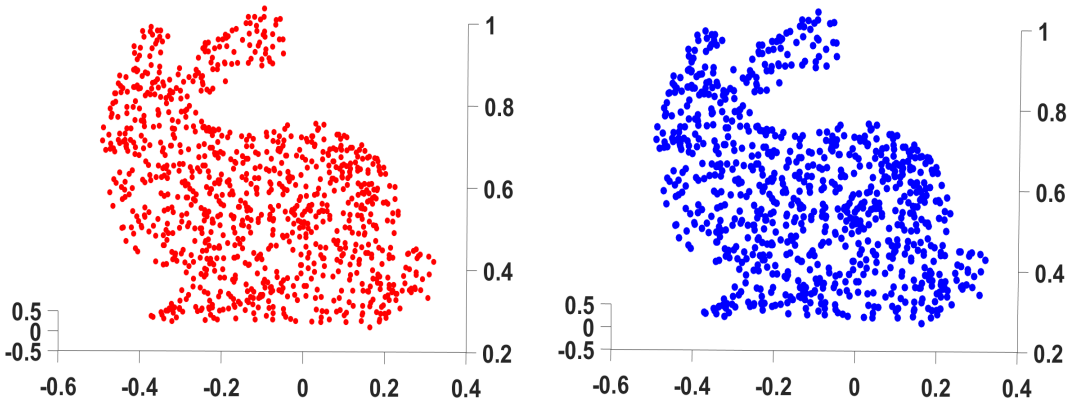
(e) Court (f) Garden

Figure 5.1: six example 3D models, (a)(b)(c) from Stanford 3D Scanning Repository, (d)(e)(f) from our new dataset

downsampled to about 1000 points with different densities. Figure 5.1 shows 6 example models from different scenes and objects.

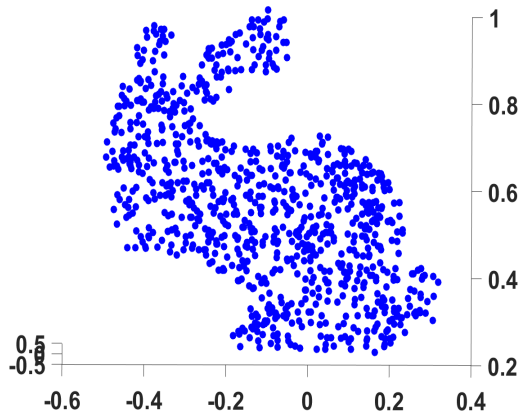
We apply different effects to simulate the real factors in the real environment. Figure 5.2 shows examples of the effects. In our experiment, the sampling rate is set to 90% and 85% for the fixed and moving point cloud, respectively. Table 5.2 gives specific information about the parameters.

Figure 5.3 shows one successful registration in a real garden. After registration, we could see the hedges and trees overlap well although there is a big patch of occlusion in both two point clouds, many outliers and noise.

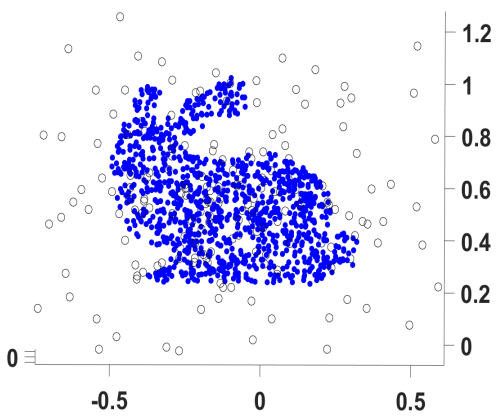


(a) Model

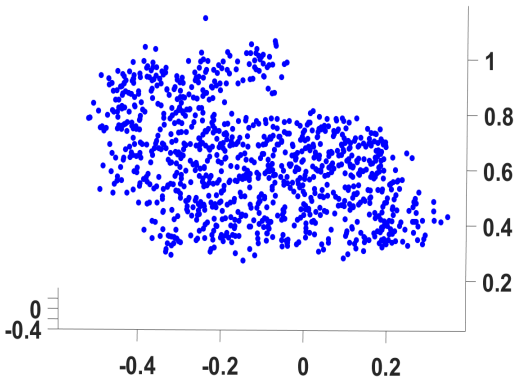
(b) Sampling rate 85%



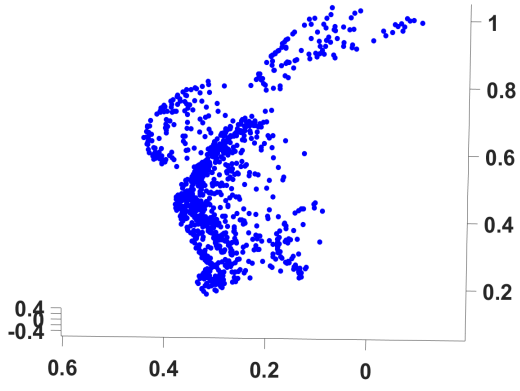
(c) Occlusion 10%



(d) Outliers 200



(e) Noise standard deviation=0.1



(f) Initial rotation angle in x,y,z=(0°,30°,30°)

Figure 5.2: Different influences from various factors.

Table 5.2: Range for random and controlled factors

Factor	*random range*	*controlled range*
Initial rotation	[-20°, 20°]	[-60°, 60°]; step=8°
Outliers	[0, 500] that is, [0, ≈33%]	[0, 2000] that is, [0, ≈67%]; step=200
Noise standard deviation	[0, 0.2] × radius of point cloud	[0, 0.3] × radius of point cloud; step=0.03
Occlusion	[0, 15%]	[0, 30%]; step=0.03

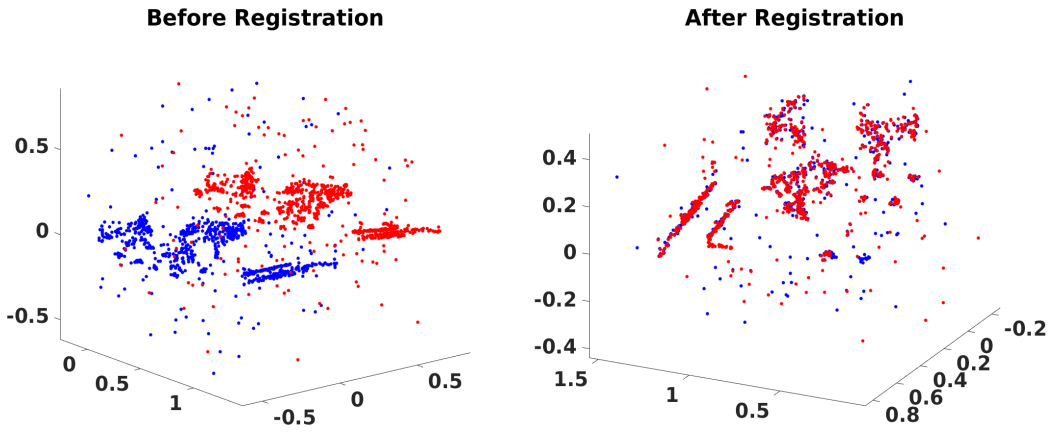


Figure 5.3: A successful registration in a real garden.

When the initial rotation angle value is the controlled variable, it ranges from [-60°, 60°], with an 8° step. In the experiments, the specific rotation angle around each axis is chosen as 0 or the initial rotation angle value randomly. Figure 5.4 shows that beyond -40°or 40°, the proposed algorithm breaks down because the iteration count exceeds the maximum. But within [-40°, 40°], our algorithm is much more stable and accurate compared with the rest. In Figures 5.4 through 5.7, the 'Time' axis refers to the average running time per registration. If rotation and translation error is below 0.2 and 0.1 respectively in a trial, the trial is a success (third plot).

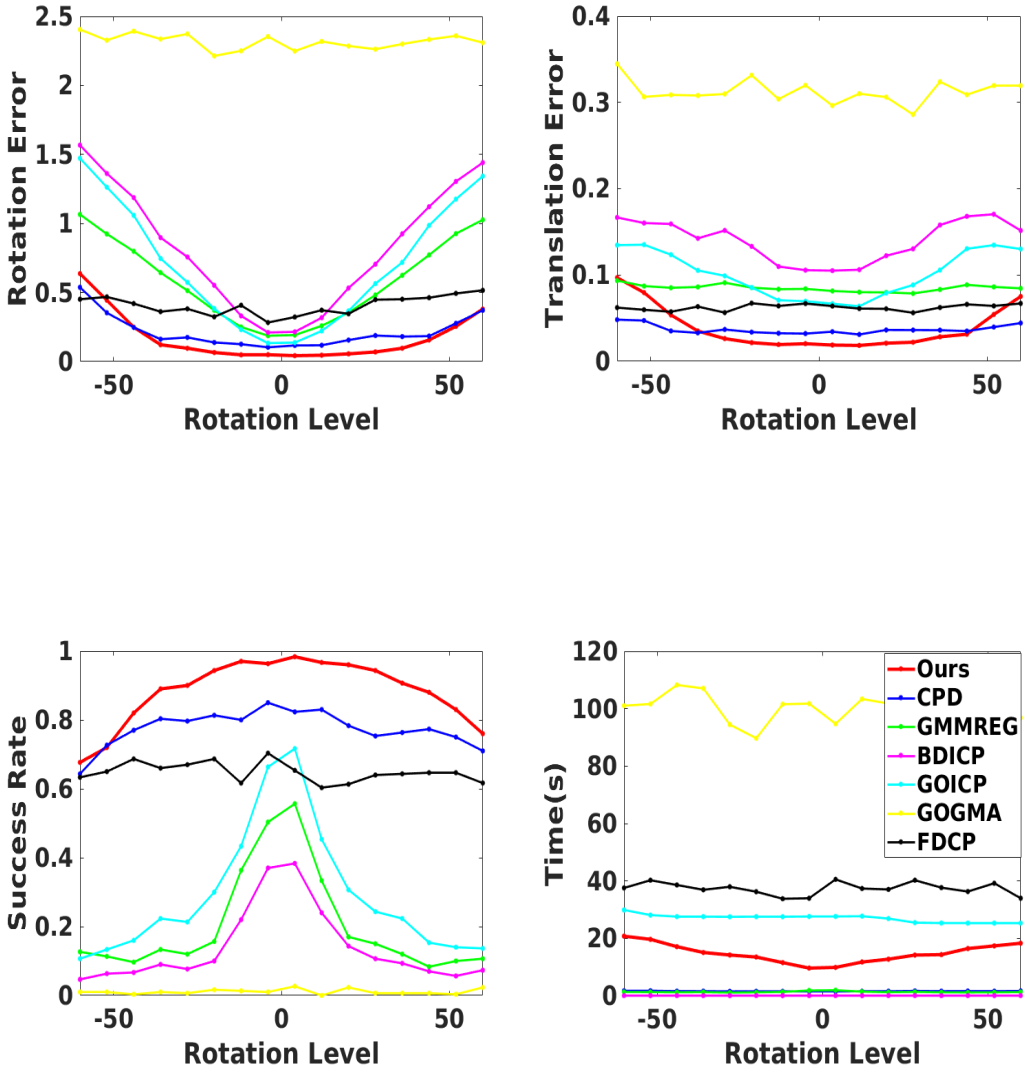


Figure 5.4: Rotation Experiment. It shows the performance change with the initial rotation between two point clouds. Top left shows the estimated rotation error after convergence ('Ours' is lower). Top right shows the estimated translation error after convergence ('Ours' is lower). Bottom left shows the percentage of experiments converging correctly ('Ours' is higher). Bottom right shows the average running time for one registration ('Ours' is in the middle).

When occlusion rate is the controlled variable, Figure 5.5 shows within 25%, the proposed algorithm performs well.

Judging that GOGMA needs much more time (about 1000 sec for a trial) to achieve a much better performance and behaved poorly in the experiments above, we will

neglect GOGMA in the remaining noise and outlier experiments but later compare the proposed algorithm with it in the small dataset registration experiment.

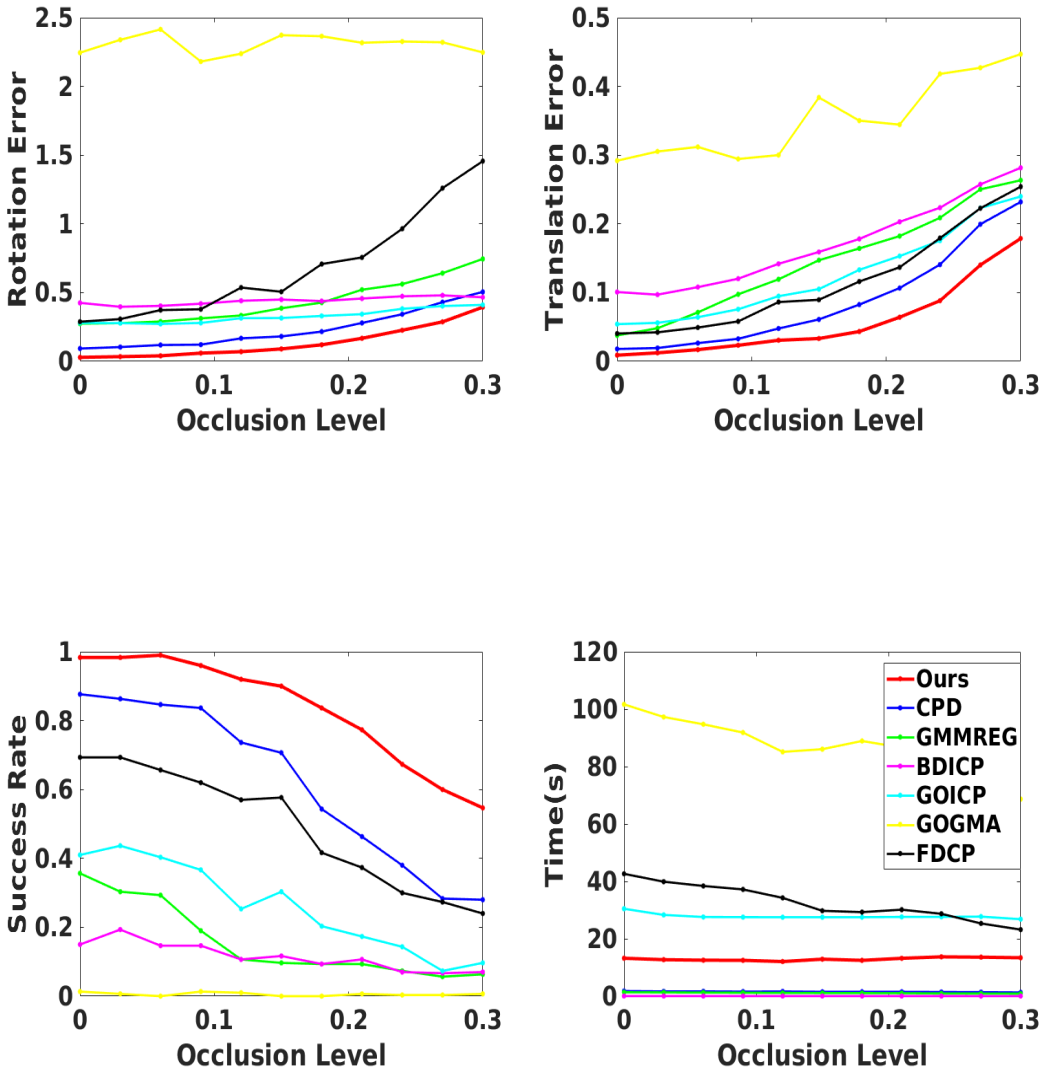


Figure 5.5: Occlusion Experiment. It shows the performance change with the percentage of occlusion in the point clouds. Top left shows the estimated rotation error after convergence ('Ours' is lower). Top right shows the estimated translation error after convergence ('Ours' is lower). Bottom left shows the percentage of experiments converging correctly ('Ours' is higher). Bottom right shows the average running time for one registration ('Ours' is in the middle).

When outliers are the controlled variable, we use covariances generated in the same manner as the true data points. Figure 5.6 shows the proposed algorithm has superior performance again.

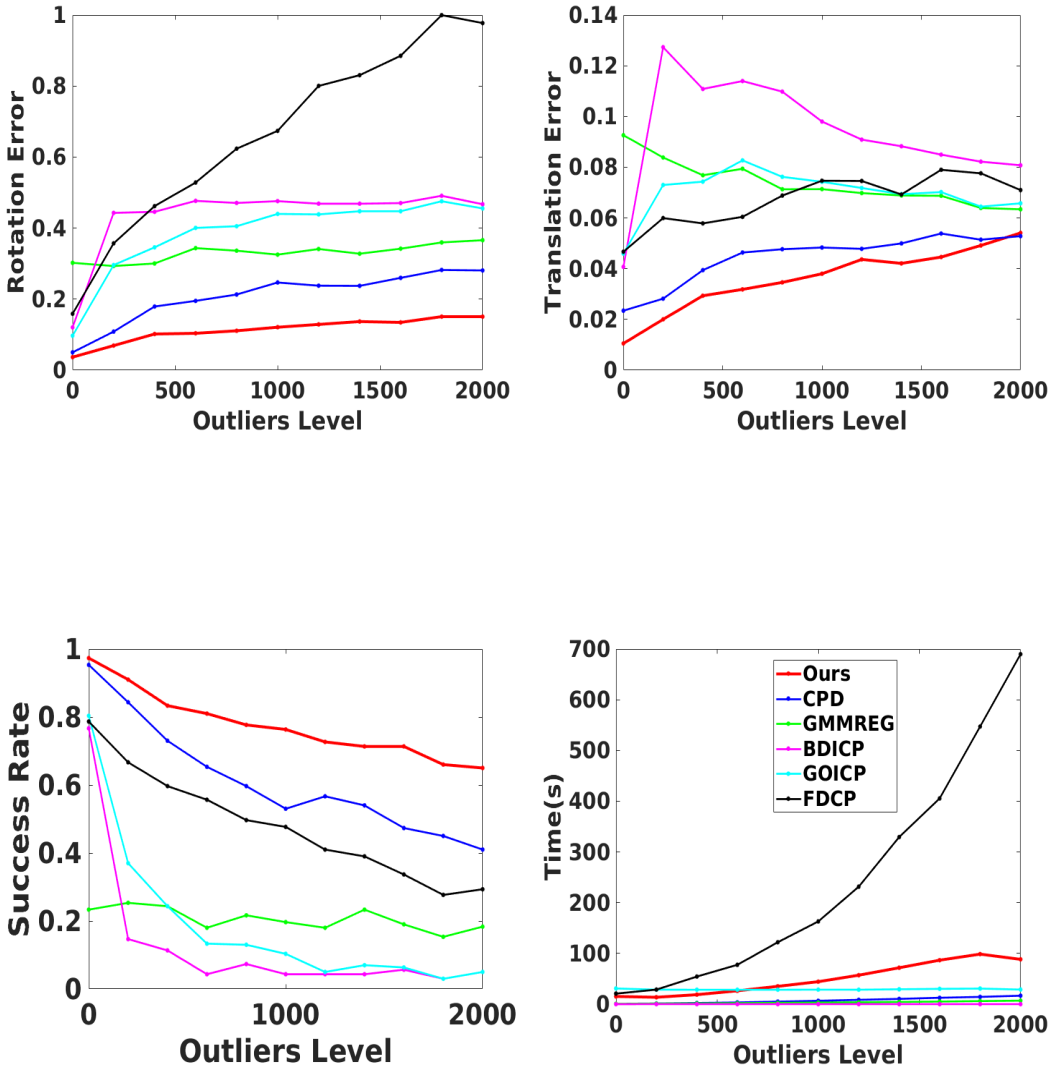


Figure 5.6: Outlier Experiment. It shows the performance change with the number of the outliers in the point clouds. Top left shows the estimated rotation error after convergence ('Ours' is lower). Top right shows the estimated translation error after convergence ('Ours' is lower). Bottom left shows the percentage of experiments converging correctly ('Ours' is higher). Bottom right shows the average running time for one registration ('Ours' is in the middle).

When the noise level is the controlled variable, Figure 5.7 shows robust and accurate performance compared with the rest.

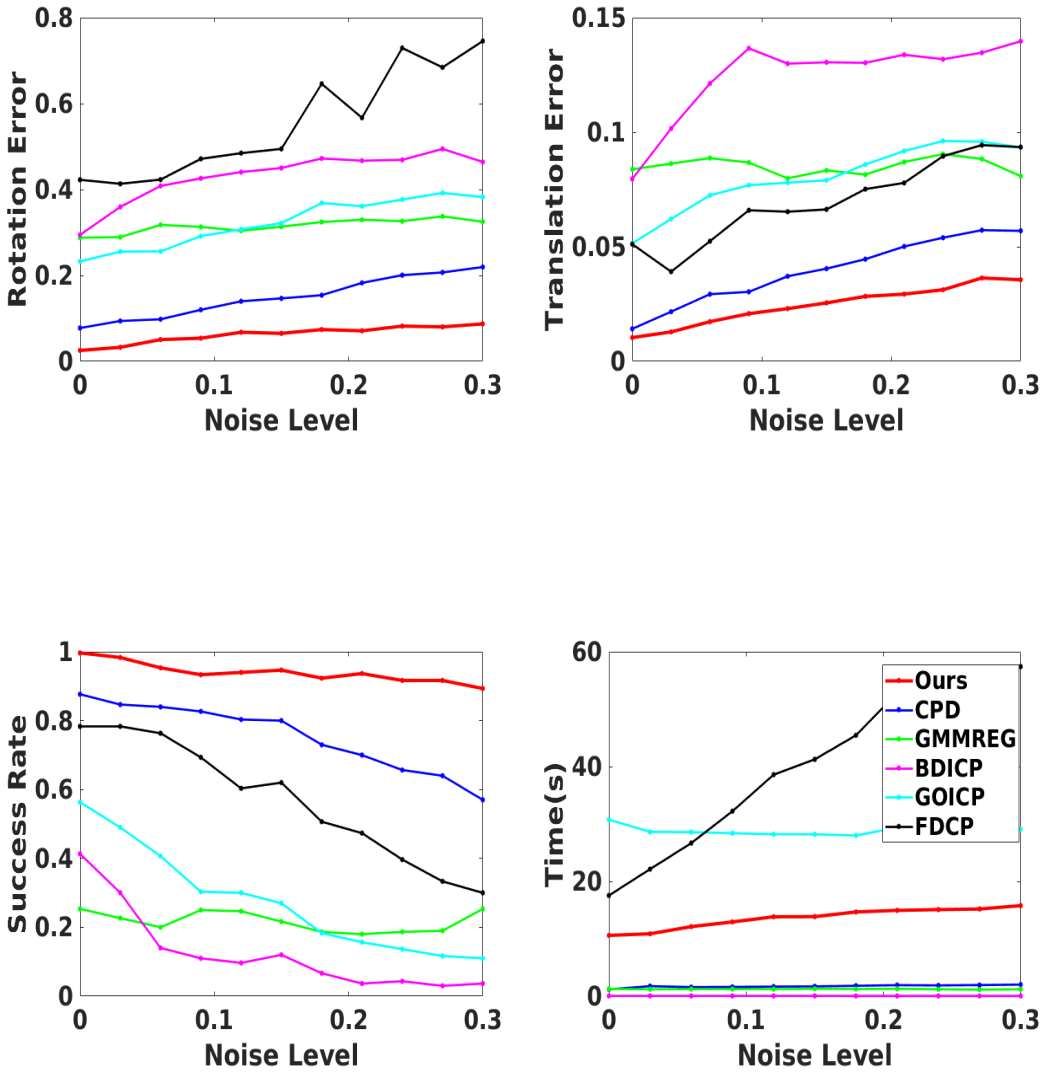


Figure 5.7: Noise Experiment. It shows the performance change with the noise extent in the point clouds. Top left shows the estimated rotation error after convergence ('Ours' is lower). Top right shows the estimated translation error after convergence ('Ours' is lower). Bottom left shows the percentage of experiments converging correctly ('Ours' is higher). Bottom right shows the average running time for one registration ('Ours' is in the middle).

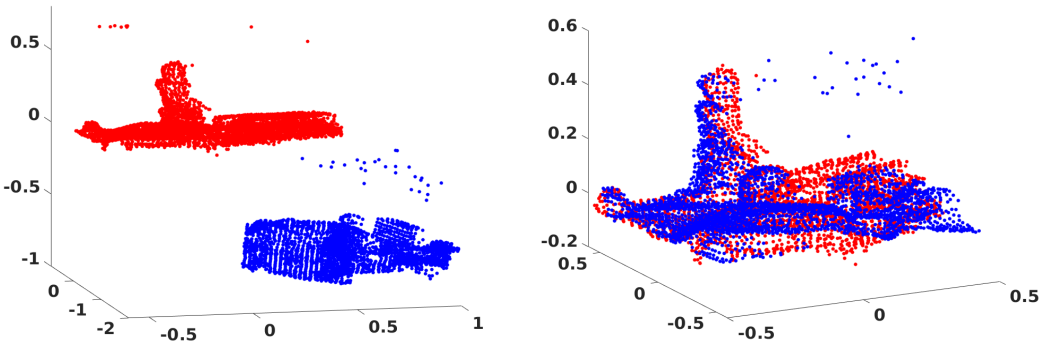
5.3.2 Real Data from Multiple Kinect Sensors

The simulation experiments above show our algorithm works well with very accurate covariance estimates. In the real case, it is hard to get very accurate covariance estimates. In this real application, we estimate an inaccurate covariance for each 3D point from a Kinect sensor to test our algorithm. We design the uncertainty of each valid 3D point acquired by the Kinect sensor based on the depth value d and the angle α between the camera and the normal of the surface [105].

$$U(\alpha, d) = exp[w_1(1-\cos\alpha) + w_2 d] \tag{5.19}$$

We use $w_1 = 1.6658$ and $w_2 = 0.2776$ by letting $U(\pi/3, 0) = U(0, 3) = 2.2$. The number 2.2 is set manually and the algorithm works well if that number is in the range [1,10] (known by our experiments). We simply multiplied the uncertainty and the identity matrix to estimate a coarse covariance for each point. Future work will explore more accurate real covariances to represent the 3D uncertainty distribution.

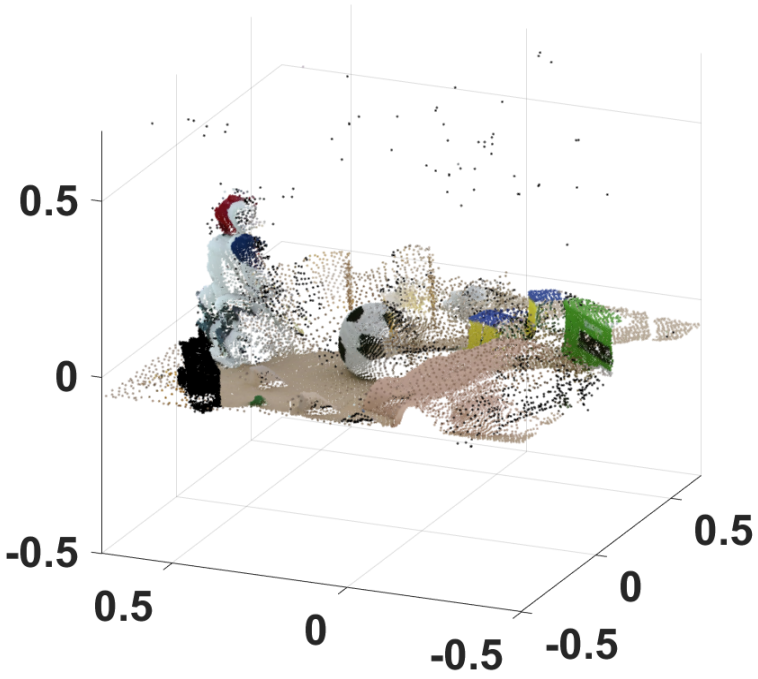
We tested our algorithm using two point clouds from two Kinect sensors. The ground truth of the rotation and translation between the two Kinect sensors is known by calibration. Figure 5.8 (a), (b) show the scene before and after registration using our algorithm. Figure 5.8 (c) adds the colour texture information into the two registered point clouds.



(a) Before registration

(b) After registration

Figure 5.8: *Cont.*



(c) After registration (texture mapped)

Figure 5.8: The figure shows a critical example of aligning two noisy and partial 3D scans with many outliers and different densities from two real low-resolution scanners using our algorithm.

In the experiment, the two point clouds have been downsampled to $\sim$4K points or so using the grid average method. The downsampling was small from $\sim$20K to $\sim$4K (rather than 640×480 to $\sim$4K): the Kinect scan was cropped to the fixed scenes. We applied the same initial rotation to all the algorithms and reduced the scale parameter for SVM (0.08, 0.08) in GOGMA and `MSEThresh` $= 0.001$ in GOICP to make them get their best performance. Here we present results from 30 scenes in our new dataset. We calculated the error and running time based on only successful registrations (rotation and translation error is below 0.2 and 0.1). After all the algorithms have converged, our successful rate (most important) ranks first (96%). The estimated mean rotation of our algorithm (0.04) is lowest, see Table 5.3. Otherwise, our algorithm is about 7 times faster than GOGMA whose success rate (93%) ranks second.

Table 5.3: Experiment Results for the Real Application

Method	Error Mean **(R)**	Error Std **(R)**	Error Mean **(t)**	Error Std **(t)**	Suc. rate	Time s/trial
CPD [103]	0.05	0.03	**0.03**	**0.01**	50%	42.0
Gmmreg [76]	-	-	-	-	0	-
BDICP [169]	-	-	-	-	0	-
GOICP [156]	-	-	-	-	0	-
GOGMA [28]	**0.04**	0.03	**0.03**	0.02	93%	1125
3dmatch [162]	0.08	0.04	0.04	0.02	23%	**6.6**
FDCP [86]	0.06	**0.02**	0.04	**0.01**	40%	8.2
Ours	**0.04**	0.03	0.04	0.02	**96%**	163

5.3.3 Additional Experiments

In addition, the method has also been tested without any uncertainty information, and each covariance matrix has been replaced with an identity matrix. The results in Section 5.3.3.1 show that the proposed method with uncertainty information is better than that without uncertainty, and both are superior to the other comparison algorithms. After that, the performance of the proposed algorithm with extremely strong factors was tested in Section 5.3.3.2.

5.3.3.1 Our method without uncertainty information for each point

All the settings were the same as in Section 5.3.1 in this chapter. One more comparison experiment was added in each group: the proposed method without uncertainty information for each point. All the covariances were replaced in the proposed method with an identity matrix, which is called 'std' method in all the figures (Figure 5.9, Figure 5.10, Figure 5.11, Figure 5.12) below. It is evident that the proposed method with uncertainty information is better than the proposed method without uncertainty information, except for the 'Outlier Experiment'. The reason is that in the proposed method with uncertainty information, all the outliers are treated as the normal points. The uncertainty information for all the outliers are wrong, and the outliers will mislead the proposed method with uncertainty information more. From all the experiments below, the proposed method (with and without uncertainty information) is superior to

the other comparison algorithms.

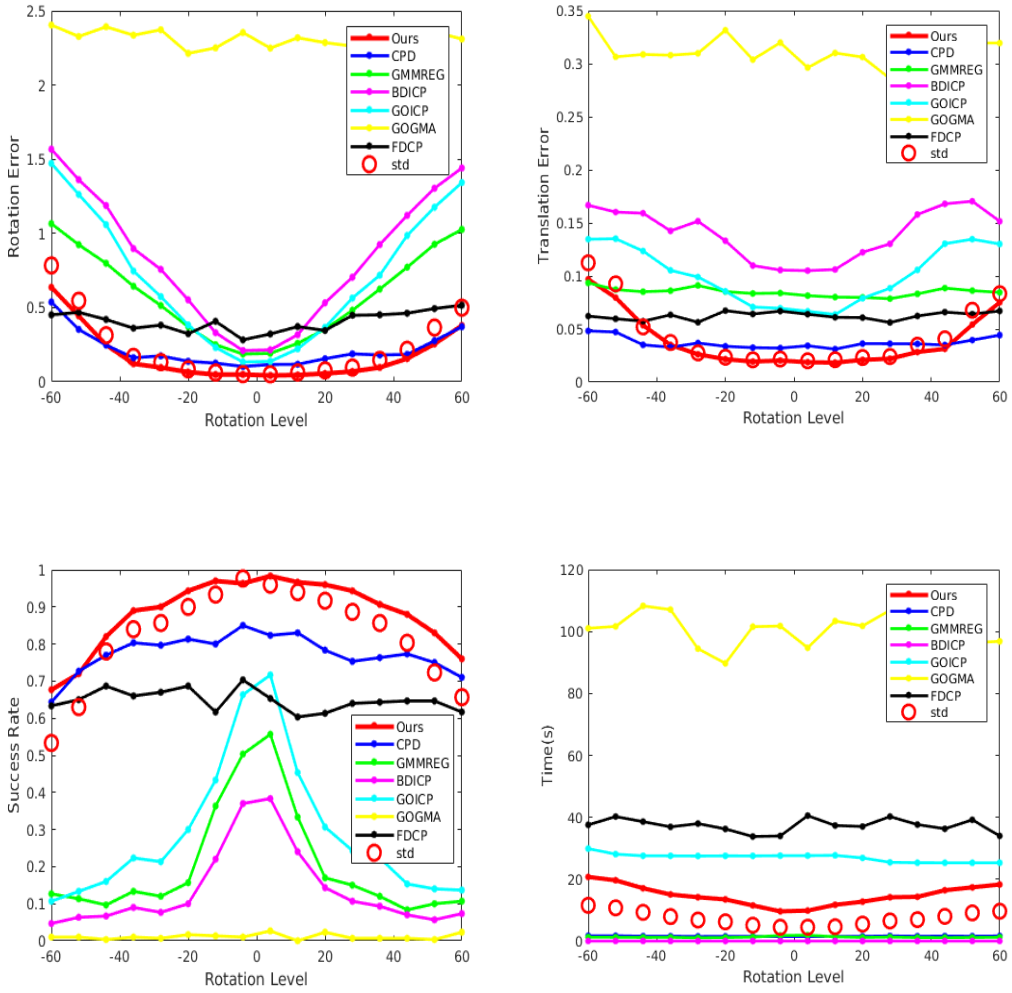


Figure 5.9: Additional Rotation Experiment. It shows the performance change versus the initial rotation between two point clouds. This figure shows the same results as Figure 5.4, except that an additional experiment with the identity matrix covariance has been added (dots). Its performance is slightly worse but faster.

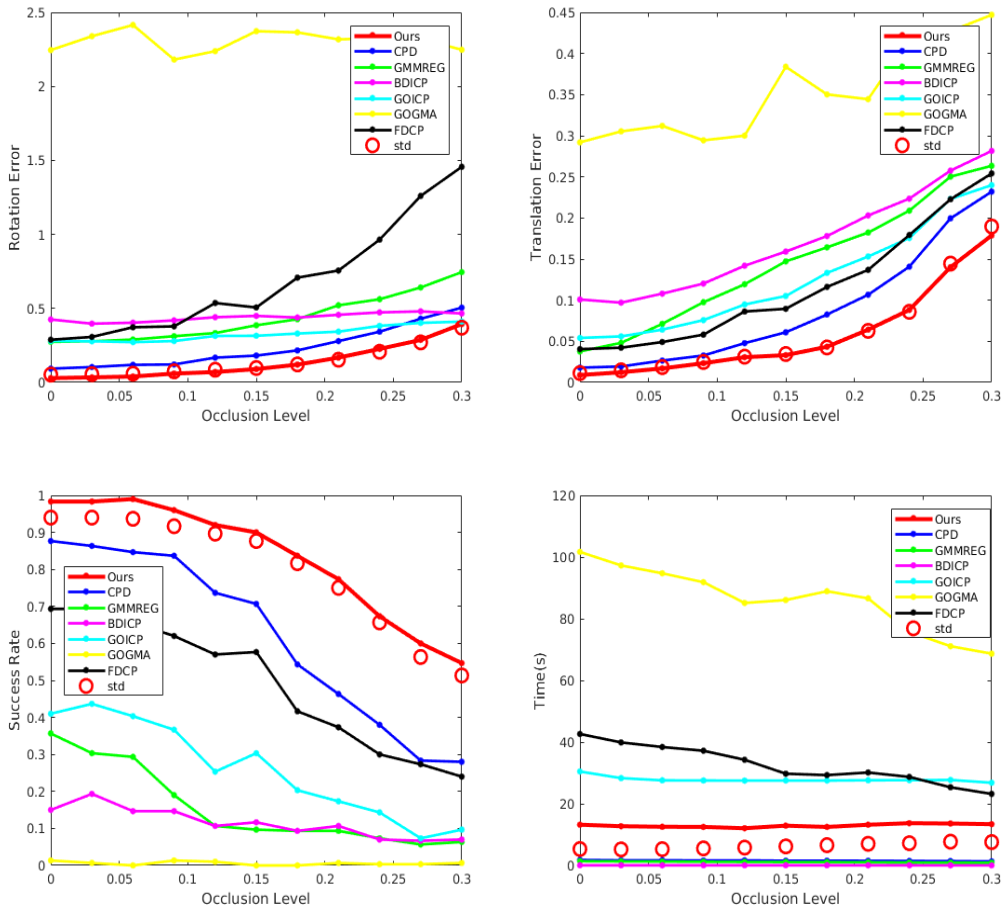


Figure 5.10: Additional Occlusion Experiment. It shows the performance change versus the percentage of occlusion in the point clouds. This figure shows the same results as Figure 5.5, except that an additional experiment with the identity matrix covariance has been added (dots). Its performance is slightly worse but faster.

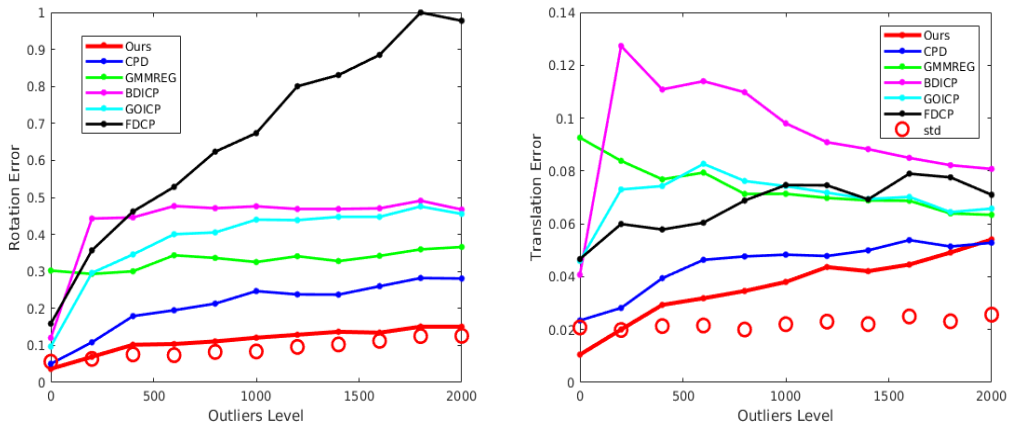


Figure 5.11: *Cont.*

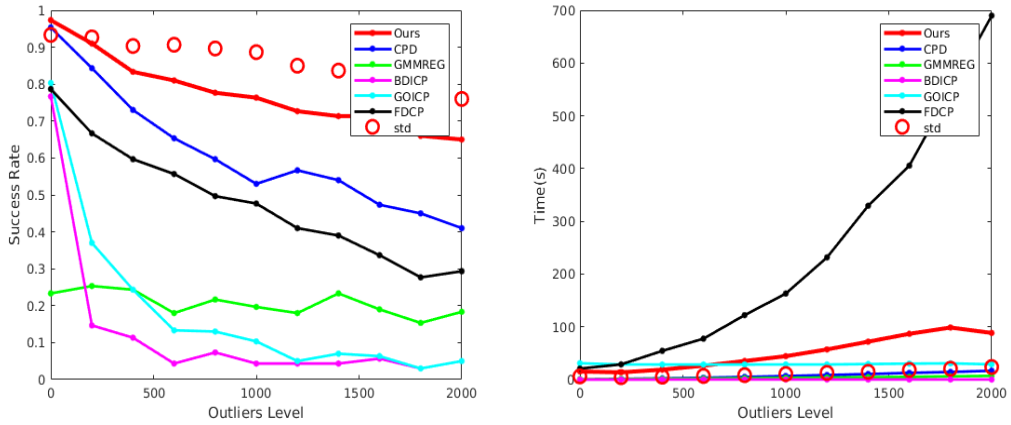


Figure 5.11: Additional Outlier Experiment. It shows the performance change versus the number of the outliers in the point clouds. This figure shows the same results as Figure 5.6, except that an additional experiment with the identity matrix covariance has been added (dots). Its performance is slightly worse but faster.

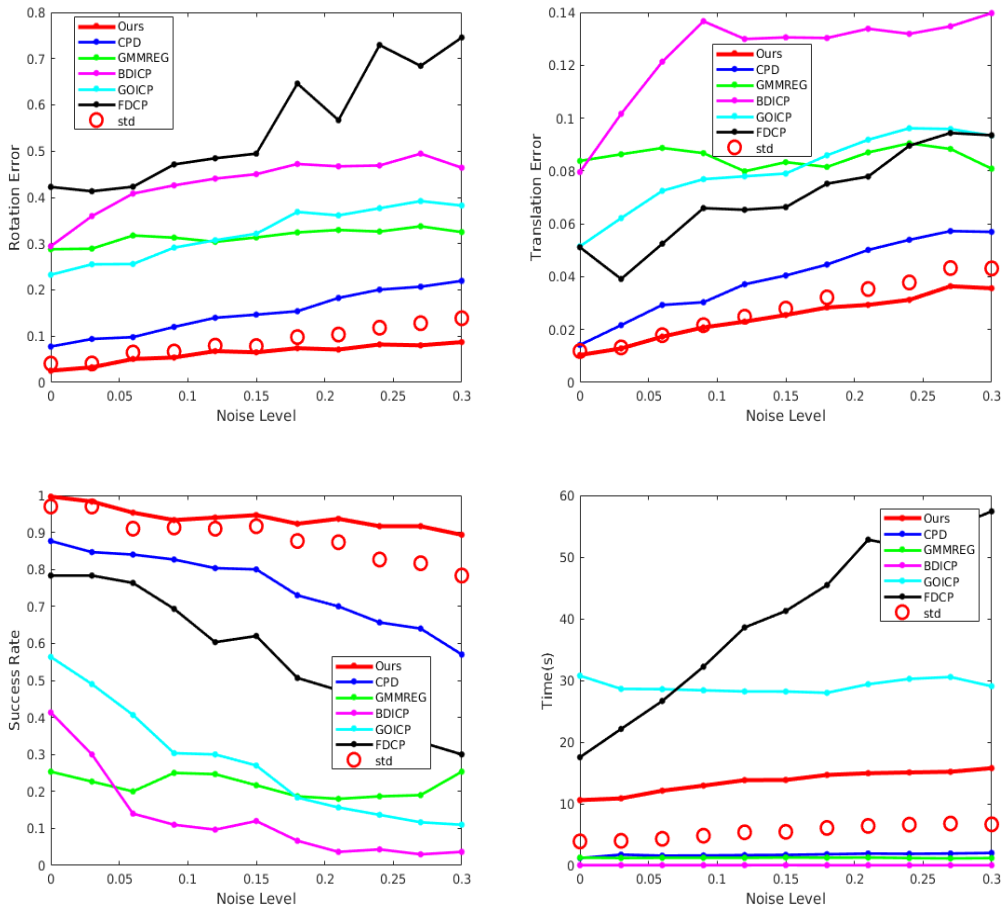
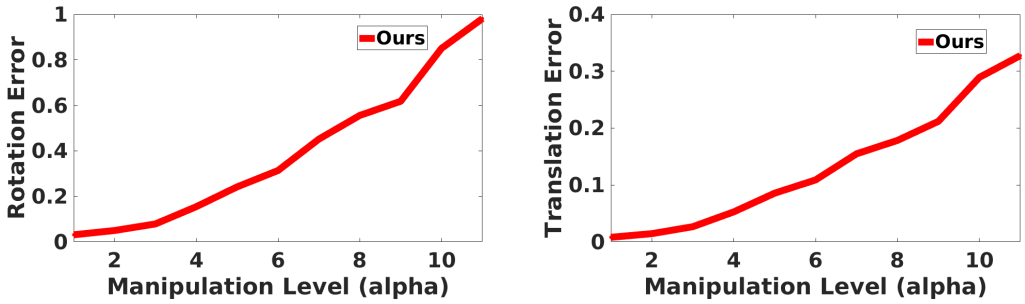


Figure 5.12: Additional Noise Experiment. It shows the performance change versus the noise extent in the point clouds. This figure shows the same results as Figure 5.7, except that an additional experiment with the identity matrix covariance has been added (dots). Its performance is slightly worse but faster.

5.3.3.2 Strong Manipulation

To show the performance with a large range of strong factors, the following experiment was conducted. Manipulate both point clouds using: $alpha$ * [rotation angle around each $axis = \pm 8^\circ$, 20% random outliers, noise $std = 5\%$ radius of point cloud, $occlusion = 5\%$ random part of point cloud]. $alpha = [1, 11]$. For each $alpha$, run 300 trials (3 trials/shape*100 shapes) and the maximum iteration of 100. The experiment was conducted on the dataset in Section 5.3.1 in this chapter, as shown in Figure 5.13.



(a) Rotation Error (b) Translation Error

Figure 5.13: Strong Manipulation

5.4 Conclusion

The proposed point cloud registration algorithm in this chapter is simpler and more effective than the previous algorithms. The 3D uncertainty distribution was incorporated into the dynamic Gaussian mixture alignment process. The obvious difference between the proposed algorithm and the previous ones is that it needs covariances at each point as input, which requires error models of how to estimate the real covariance for each kind of sensor. A more robust energy function for point cloud registration was proposed here with an efficient approximation to the optimization step that greatly reduces the computational complexity. To test the accuracy and robustness of the proposed algorithm, the proposed method was compared with relevant recent algorithms from the top journals and conferences: CPD [103], GMMREG [76], BDICP [169], GOICP [156], GOGMA [28], 3DMATCH [162], FDCP [86]. All the experiments show that the proposed method is very robust, accurate, and works well. The proposed method could be used in rigid 2D point cloud registration as well, see Appendix B. Additionally, the proposed method was adapted to the real robot. Further evaluation of the proposed method in this chapter on the real robot is presented in Section 6.2.2. The previous 3 chapters (Chapter 3, Chapter 4 and this chapter) are relatively independent. In the next chapter, the proposed methods in these chapters will be integrated into the ROS system on a real robot (called Trimbot) for solving practical problems in the real world.

Chapter 6

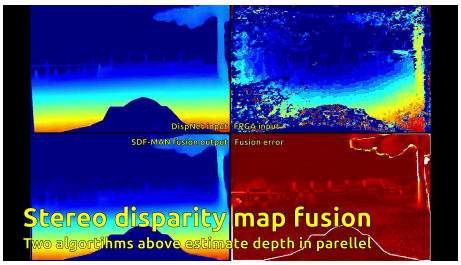
Application on the Real Robot

In this chapter, the previously proposed algorithms with different functions are integrated into the navigation system of a real robot (called the Trimbot) to solve the practical problems in a real garden. The fusion pipeline in the ROS system includes the following: rectified stereo intensity image input from a stereo vision camera, two stereo vision algorithms to obtain initial disparity maps for the same scene, the proposed supervised disparity fusion algorithm to fuse the initial disparity maps, conversion of the refined disparity map into the 3D point cloud in the current view, registering consecutive point clouds to obtain the 6D pose, and fusing multiple point clouds from different views into the global system. In addition, the proposed supervised disparity fusion algorithm and rigid point cloud registration are further evaluated in a real garden for the TrimBot2020 project. A real robot outdoor navigation demo can be seen at in the following: `https://youtu.be/4OfdHUsR-ZI`. The corresponding sequence of screenshots from the video is shown in Figure 6.1. The results presented in this chapter were achieved during January, 2019. The majority of the content in this chapter is from our confidential report to TrimBot2020. More specifically, Radim Tylecek wrote the part in Section 6.2.1 based on my experiment results. As for the rest in this chapter, I finished it by my own.

(a) t = 0:00:01

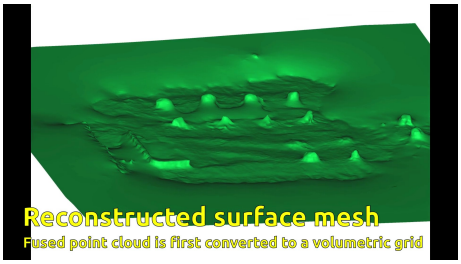
(b) t = 0:00:07

(c) t = 0:00:18

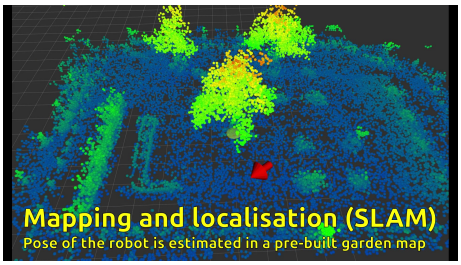


(d) t = 0:00:38

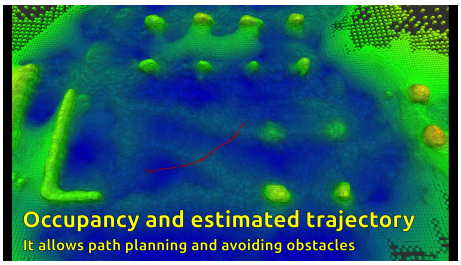
(e) t = 0:00:46



(f) t = 0:01:01



(g) t = 0:01:17



(h) t = 0:01:38

Figure 6.1: A sequence of screenshots from the robot navigation demo video. The format of t in this figure is t = hour:min:sec.

6.1 Fusion Pipeline in ROS System

Table 6.1: ROS Topic Information

Topic	ROS Topic Name	ROS Topic Type	Comment
imageL	$/uvc\_cam1\_rect\_mono$	$sensor\_msgs/Image$	rectified left intensity image
imageR	$/uvc\_cam0\_rect\_mono$	$sensor\_msgs/Image$	rectified right intensity image
disp1	$/uvc\_cam1\_disp\_fpga$	$sensor\_msgs/Image$	disparity map from stereo vision camera [68]
disp2	$/dispnet$	$stereo\_msgs/DisparityImage$	disparity map from Dispnet [97]
disp	$/local\_fusion\_disparity$	$stereo\_msgs/DisparityImage$	disparity map from the supervised method in Sdfman [119]
camInfoL	$/uvc\_cam1\_rect\_mono/camera\_info$	$sensor\_msgs/CameraInfo$	left camera information
point cloud	$/local\_fusion\_pcl2$	$sensor\_msgs/PointCloud2$	point cloud from disparity map
camera pose	$/local\_fusion\_rel\_pose\_dugma$	$geometry\_msgs/PoseStamped$	relative camera pose

The SkyAware S360 stereo vision camera publishes[1] the rectified left intensity image (*imageL*), rectified right intensity image (*imageR*), disparity map in the left view ($disp1$) and the left camera information ($camInfoL$). The rectified left & right intensity images are subscribed[2] to by the node[3] Dispnet. The node Dispnet calculates and publishes the disparity map in the left view ($disp2$). The node Sdf-MAN subscribes to $disp1$, $disp2$, and *imageL* to obtain a refined disparity map in the left view ($disp$). Subsequently, $disp$ and $camInfoL$ are subscribed to by the node Projection to obtain the corresponding point cloud (*point cloud*). The node DUGMA subscribes to *point cloud* at time $t = m-1$ and $t = m$ (m is positive integer) to obtain the relative camera pose (*camera pose*). Figure 6.2 shows the pipeline in the ROS system at time $t = m-1$ and $t = m$, Table 6.1 lists the ROS topic information, and Table 6.2 lists the node information.

We test the ROS fusion pipeline above in a real garden in Wageningen. Figure 6.3 shows the intermediate results. In Figure 6.3, (a) shows the left rectified intensity image *imageL* and (b) shows the right rectified intensity image *imageR*. (c) is the initial disparity map $disp1$ from the SkyAware S360 stereo vision camera. (d) is the disparity map $disp2$ from Dispnet. (e) is the refined disparity map $disp$ from Sdf-man. (f) (g) show the appearance of *point cloud* from front and left views. We can see that the error

[1]Definition of 'publish': broadcast a message in the ROS network continually.
[2]Definition of subscribe: receive specific messages over the ROS system continually.
[3]Definition of 'node': the ROS term for an executable that is connected to the ROS network.

Table 6.2: Node Information

Node	Comment
Dispnet	To calculate the disparity map using Dispnet [97]
Sdf-MAN	To refine the disparity maps using Sdf-MAN [119]
Projection	To convert the disparity map into a point cloud
DUGMA	To calculate relative camera pose between each two point clouds [118]

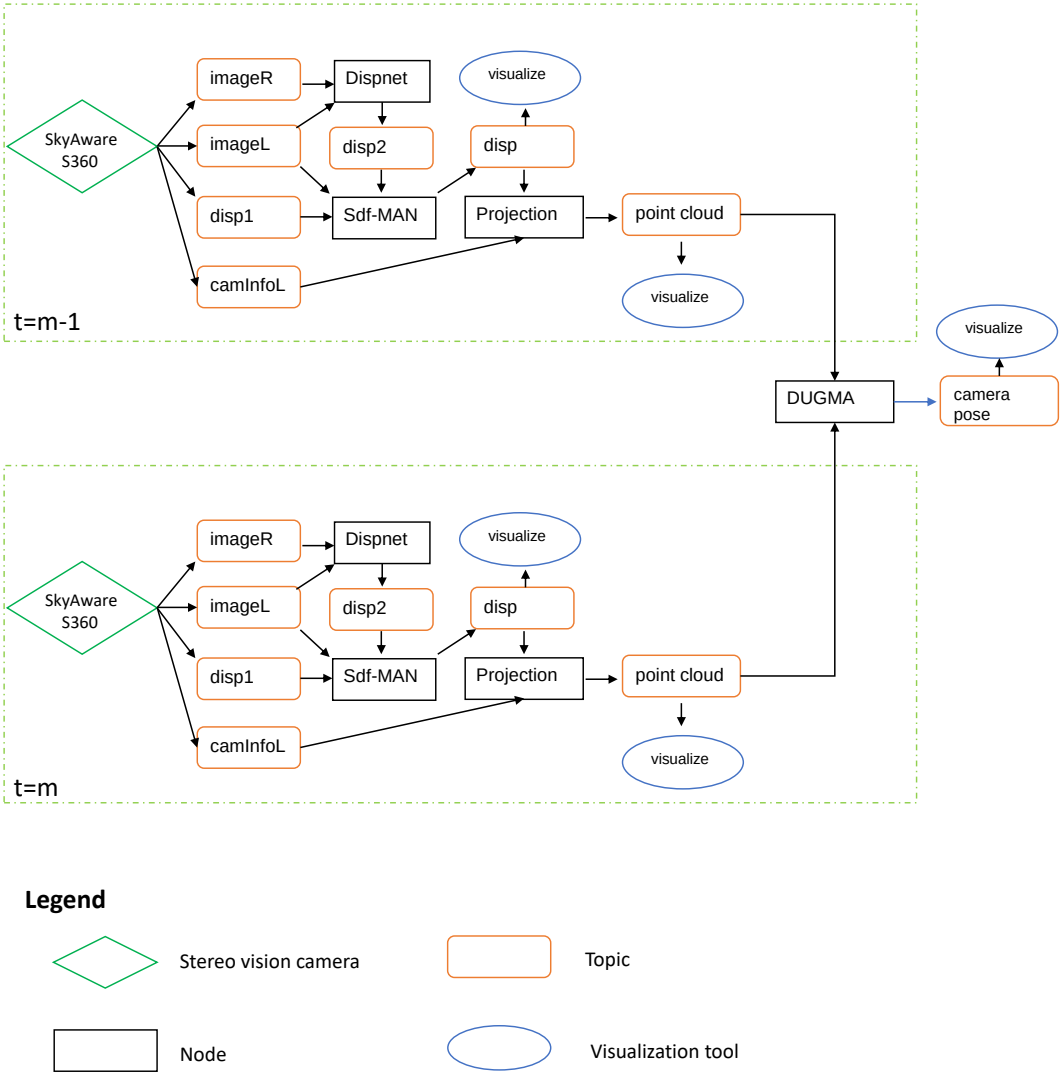
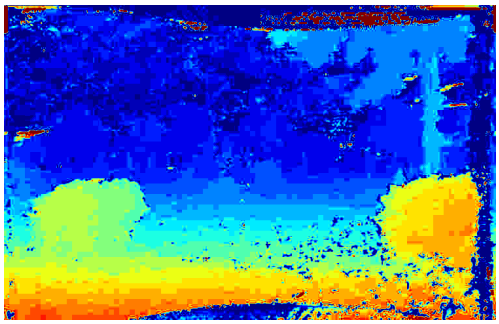


Figure 6.2: The Pipeline in ROS System at $t = m - 1$ and $t = m$.

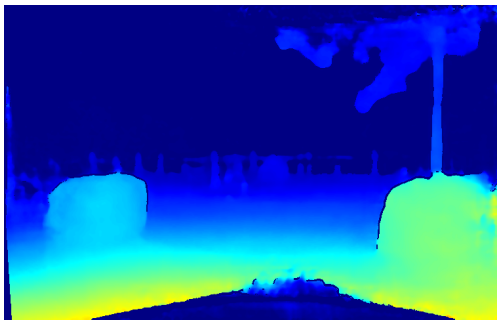
increases with the distance.

(a) *imageL*

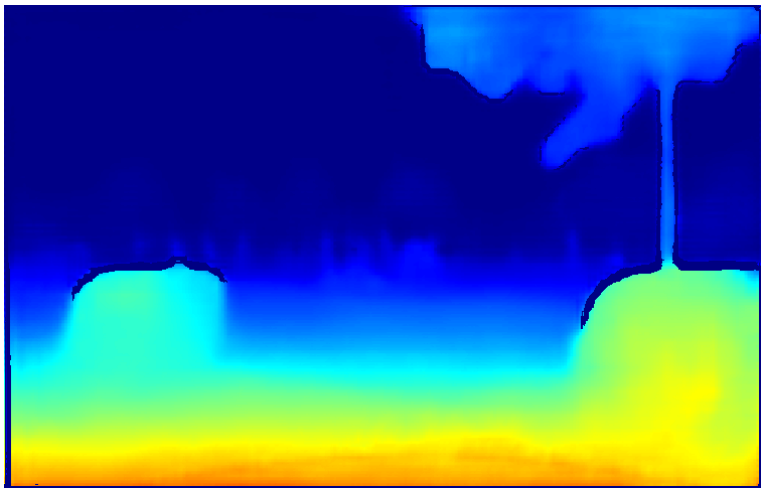
(b) *imageR*



(c) *disp*1

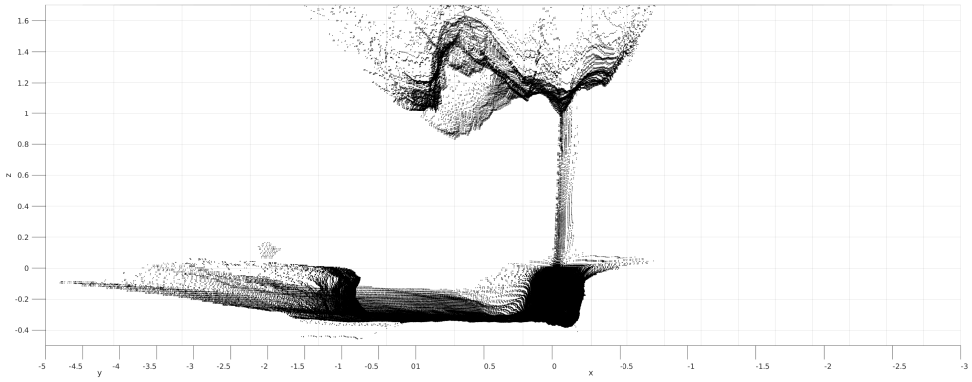


(d) *disp*2

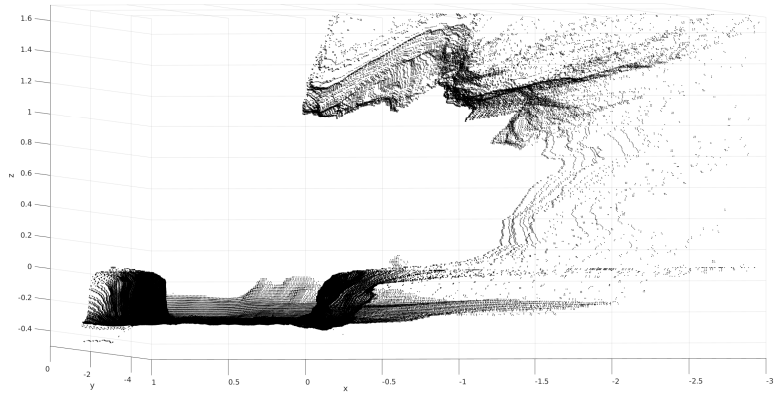


(e) *disp*

Figure 6.3: *Cont.*



(f) Appearance of *point cloud* from front view



(g) Appearance of *point cloud* from left view

Figure 6.3: Intermediate Results in the Pipeline in ROS System

A set of 270 point clouds from Sdf-MAN was fused to evaluate the global fusion. That is, we fused the *point cloud* from frame $t = 1$ to $t = 270$ into the global coordinate system using the ground truth camera pose from the Topcon Total Station position tracking system rather than DUGMA[4]. From each view, only a subset of points were used: Points further than 10 m from the camera were discarded due to low accuracy. Similarly, points in the top 1/3 of the image (moving tree branches or sky) were ignored to focus on reconstruction of the ground surface and bushes, which are of interest for

[4]It is because DUGMA does not work all the time, which fails to reconstruct the whole garden. In the near future, we will explore fusing multiple camera poses from visual SLAM, IMU, and DUGMA to obtain a more accurate and robust camera pose.

navigation and trimming in this project. All point clouds were merged into one using the ground truth pose estimates from the Topcon Total Station position tracking system, as shown in Figure 6.4. While hedges and topiary bushes can be identified clearly, tree trunks (and other thin structures) are not well represented. This is because most of their corresponding points are noisy; hence, they were removed by the statistical outlier filter.

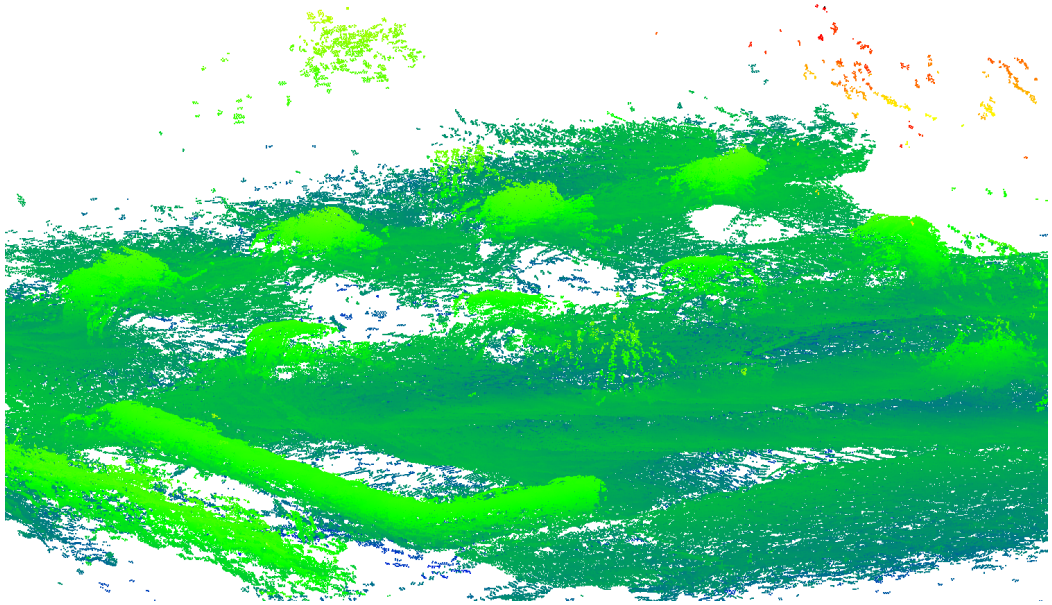


Figure 6.4: The figure shows the merged point cloud after merging 270 fused point clouds from Sdf-MAN using ground truth poses.

6.2 Evaluation

In this section, the performance of two steps in the fusion pipeline will be evaluated (disparity fusion result '*disp*', pose estimation result '*camera pose*' in Figure 6.2) on the Trimbot2020 Garden Dataset A.1.

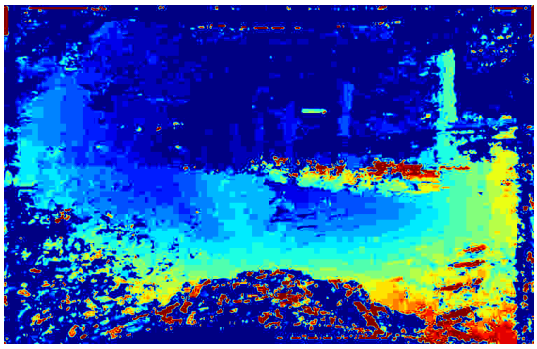
6.2.1 Disparity Fusion Evaluation

The supervised Sdf-MAN fusion of the two inputs ('*disp*1', '*disp*2' in Figure 6.2) was fine-tuned on 1000 training images from the Trimbot2020 Garden Dataset A.1. The GT (ground truth) disparity was computed from the static off-line scan of the garden using a laser scanner, which did not correspond exactly to the actual images of moving

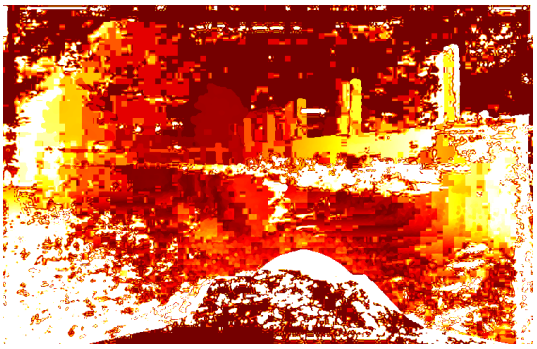
dynamic objects, which were taken subsequently during an actual drive of the platform through the garden. Also, the limited accuracy of the GT camera poses used for the point cloud projection resulted in invalid GT disparities near object boundaries.

Two test examples comparing the inputs and output can be seen in Figure 6.5 and Figure 6.6 on the test set. Overall, the Sdf-MAN algorithm ('$disp$') provides smoother results and has learned small corrections of Dispnet ('$disp2$'), which are visible in the far range (background). In Figure 6.6 it is also able to reasonably extrapolate disparities outside the undistorted visible area near borders (tree crown and close ground). The extent of how the coarse FPGA stereo input '$disp1$' (a) actually contributes to the fusion was not systematically investigated, but in this example it seems to help to rectify the uneven (bent) ground surface estimation of Dispnet (c), resulting in flat ground in the fused output (e).

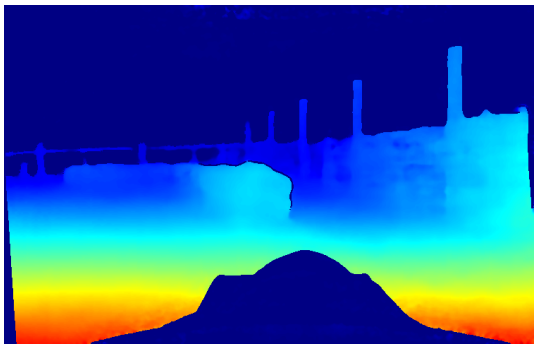
The previously mentioned inaccuracy of the GT object boundaries not only impacted on the evaluation (errors indicated there can be falsely reported), but also on the fine-tuning of the networks. In the case of Sdf-MAN, we observed excessive smoothing of discontinuities, even after pixels with steep disparity gradients were removed from the output maps during post-processing. Smoothing near boundaries reduces the overall disparity error cost for which the network is optimised, but can negatively impact the subsequent reconstruction steps by adding noise to the free space behind objects ('curtain' artifacts). Dispnet alone appears to handle disparity discontinuities better, with only one pixel wide gaps at the object boundary or sharper discontinuity steps.



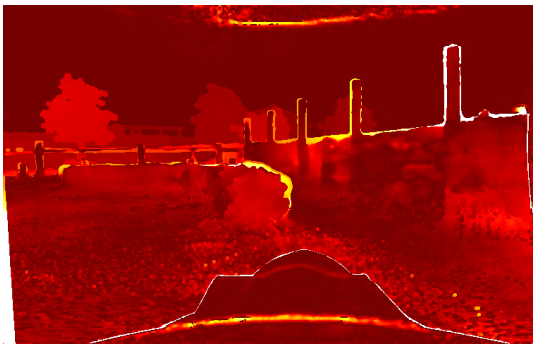
(a) FPGA stereo '$disp1$'



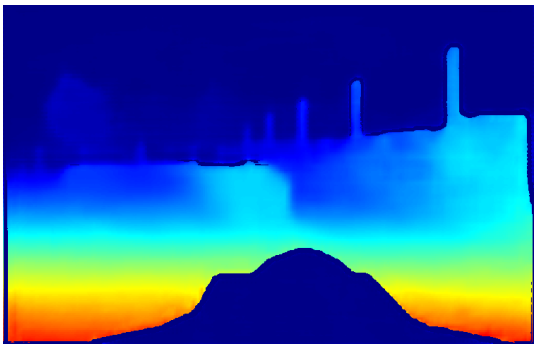
(b) FPGA stereo '$disp1$' error



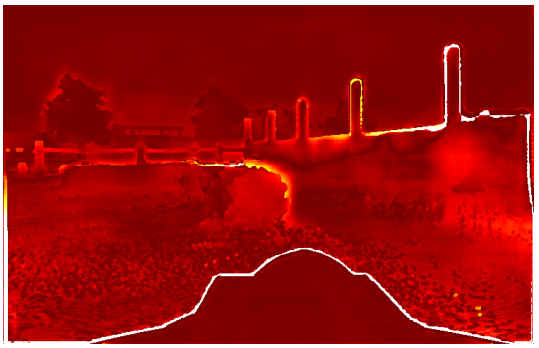
(c) Dispnet '$disp2$'



(d) Dispnet '$disp2$' error

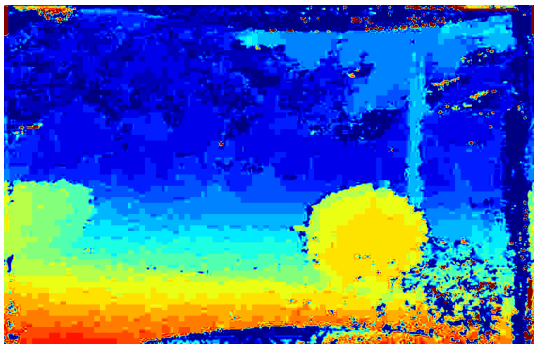


(e) Sdf-MAN '$disp$'

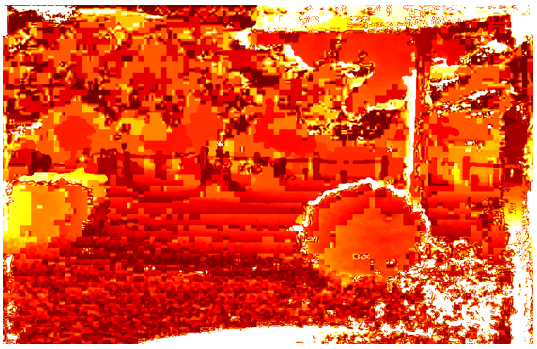


(f) Sdf-MAN '$disp$' error

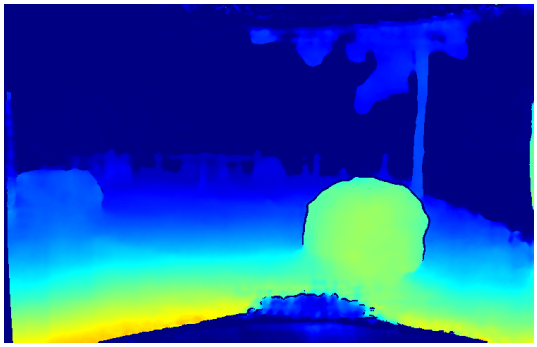
Figure 6.5: Disparity maps and errors of two inputs (FPGA stereo '$disp1$', Dispnet '$disp2$') and the fused result (Sdf-MAN '$disp$') and its error from the front-view stereo pair. *Left:* color-coded disparity (cold = far, warm = near). *Right:* color-coded accuracy (dark = small error, bright = large error, white = 5+ pixels error). The GT provided by the static laser point cloud can be inaccurate near dynamic object boundaries. The large dark region at the bottom centre is the vehicle, which is excluded from analysis.



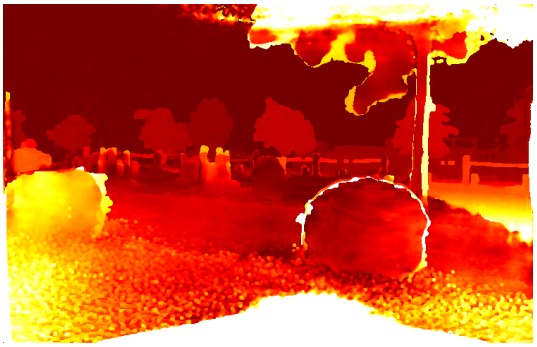
(a) FPGA stereo '$disp1$'



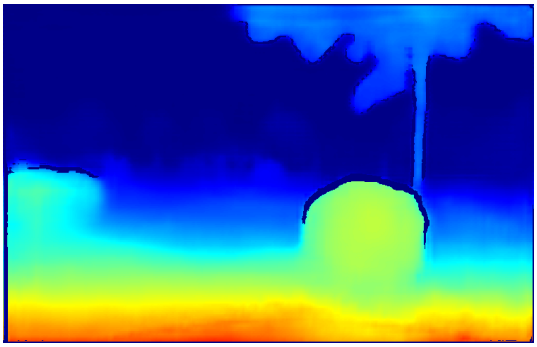
(b) FPGA stereo '$disp1$' error



(c) Dispnet '$disp2$'



(d) Dispnet '$disp2$' error



(e) Sdf-MAN '$disp$'

(f) Sdf-MAN '$disp$' error

Figure 6.6: Disparity maps and errors of two inputs (FPGA stereo '$disp1$', Dispnet '$disp2$') and the fused result (Sdf-MAN '$disp$') and its error from the side-view stereo pair. *Left:* color-coded disparity (cold = far, warm = near). *Right:* color-coded accuracy (dark = small error, bright = large error, white = 5+ pixels error). The GT provided by the static laser point cloud can be inaccurate near dynamic object boundaries.

Figure 6.7 shows the absolute disparity error of the static pixel subset and the distribution of error magnitudes (badX[5]) after post-processing the initial Sdf-man output[6]. The overall statistics suggest that the mean absolute error of the two inputs, FPGA stereo '*disp*1' with 2.94 px and Dispnet '*disp*2' 1.36 px, was successfully reduced by Sdf-MAN '*disp*' to 0.80 px. The same proportion of the performance gain was seen in the total badX scores (percentage of pixels with absolute error $\geq X$ px) for all $X = 1, \ldots, 4$.

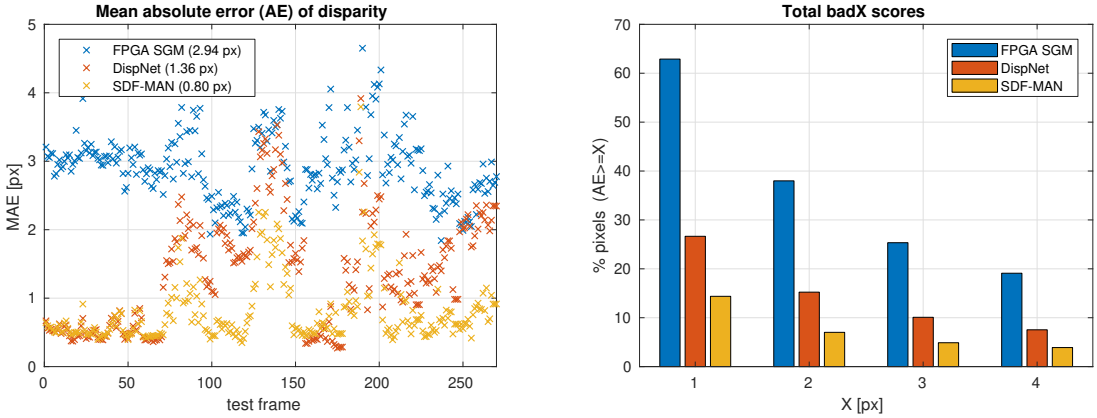


Figure 6.7: Comparison of mean absolute disparity errors (AE) of two inputs (FPGA stereo, DispNet) and the fused result (Sdf-MAN) on the real garden test set. Numbers in brackets are overall (mean) values over all frames.

The distribution of badX per-frame results presented in the four plots in Figure 6.8 suggest that the fusion helps particularly with difficult instances, while the easier ones were already fully solved by Dispnet (to the GT accuracy limit). Detailed analysis of the mean absolute error of individual image frames revealed that the performance was improved by fusion in 60% of all cases. Moreover, 30% of all cases revealed a comparably similar performance (± 0.2 px), and in 10% of all cases the fusion degraded the Dispnet result. This performance decrease can be attributed to instances where the two inputs were contradictory rather than complementary; hence, violating the assumption for successful fusion.

[5]It refers to the percentage of pixels whose absolute error was bigger than X pixels in one frame.
[6]In the post-processing step, we removed the disparities at the edge.

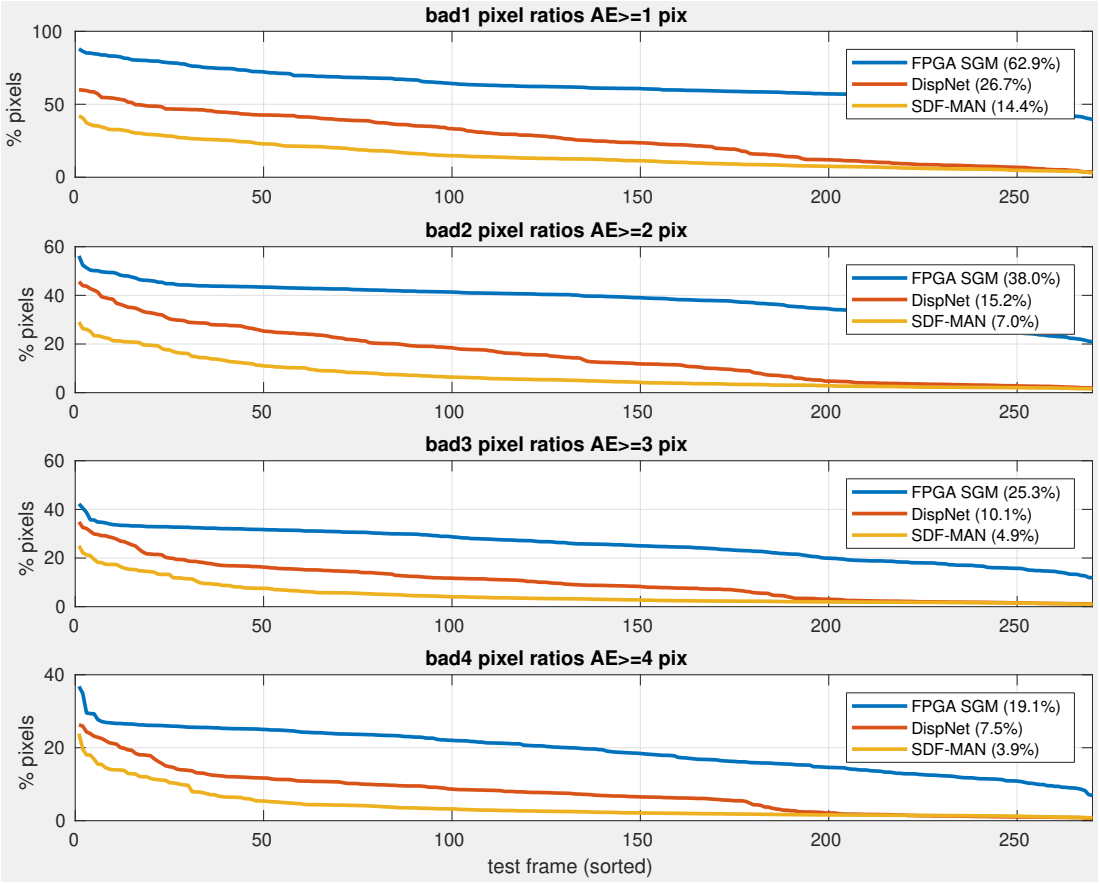


Figure 6.8: BadX pixel ratio plots show the distribution of per-frame results (sorted independently for each method) with difficult instances to the left and easy ones to the right (lower error ratio is better). The four plots show results for increasing levels of absolute error (1, 2, 3, 4).

6.2.2 Camera Pose Evaluation

The goal of relative pose estimation with DUGMA is to provide refined poses for alignment of point clouds. The performance was evaluated on the Trimbot2020 Garden Dataset A.1, where 250 registration samples were chosen to evaluate DUGMA. The average distance between two consecutive poses is approximately 0.5 m and rotation up to 45°. The poses in the dataset correspond to every 10th originally captured image frame.

Registration was computed independently for all pairs of consecutive camera poses, based on the point clouds generated by back projection of fused Sdf-MAN disparities. The older point cloud $t-1$ in the pair was considered fixed and the 6DoF relative pose

of the other (moving) point cloud t was estimated by DUGMA. Both point clouds were subsampled to 500 points: the horizontal ground segment was removed by using height threshold, and nearly 500 points were sampled from vertical objects. Objects are salient features of the point cloud and allow the establishment of the right correspondences between the two point clouds. The initial estimate was set to identity and the algorithm was run for 50 iterations, with an average run-time of 5 s on a GPU.

A successful registration is defined as follows: mean absolute rotation angle error around each axis is below 10°, and maximum absolute translation error around each axis is below 0.3 m. To know the uncertainty of each registration, the following uncertainty measurement is defined.

$$Confidence = 100 - \sigma/\rho * 100 \tag{6.1}$$

In Equation 6.1, σ is the mean minimum distance between the closest points from two point clouds after registration. ρ is the parameter defined by the user according to the density of the point cloud. After using the confidence measurement, the uncertainty system regards the registration whose confidence is above 60% as correct. 117 samples ($Confidence > 60\%$) are considered correct. Among those 117 samples, 83 samples are real successful registrations (83/117=70.94%). The mean rotation error around each axis is 4.35°, and the mean translation error around each axis is 0.14 meters. Additionally, there are 50 samples which have low confidence but successful registrations. There are 133 samples (83+50=133) which have been registered successfully indeed. There are 117 samples (250-133=117) which have been registered unsuccessfully indeed. Figure 6.9 shows the rotation and translation error across 250 registration samples after sorting according to the rotation and translation error. Red points represent the registration results whose estimated confidence is not below 60%. Cyan points represent the registration results whose estimated confidence is below 60%. We could see the majority of the points with high estimated confidence enjoy low rotation and translation error.

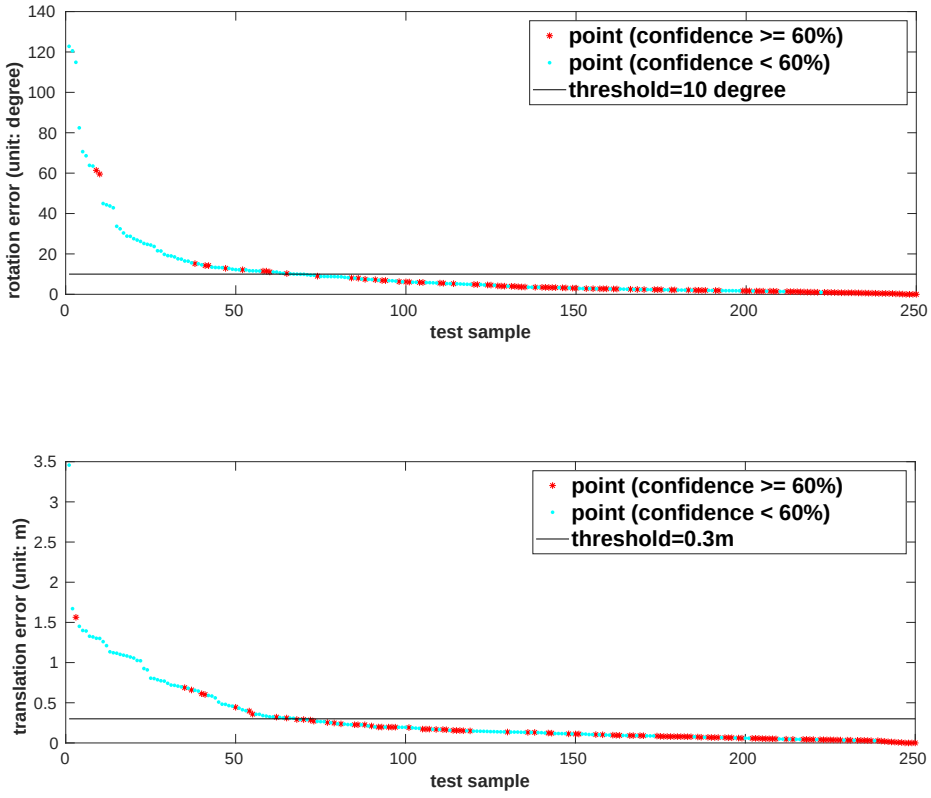
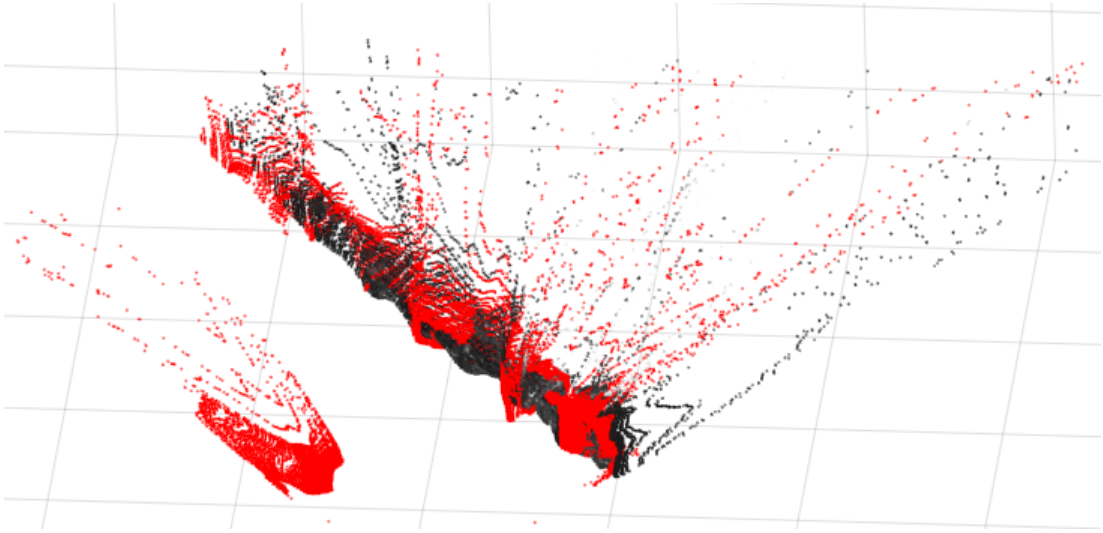
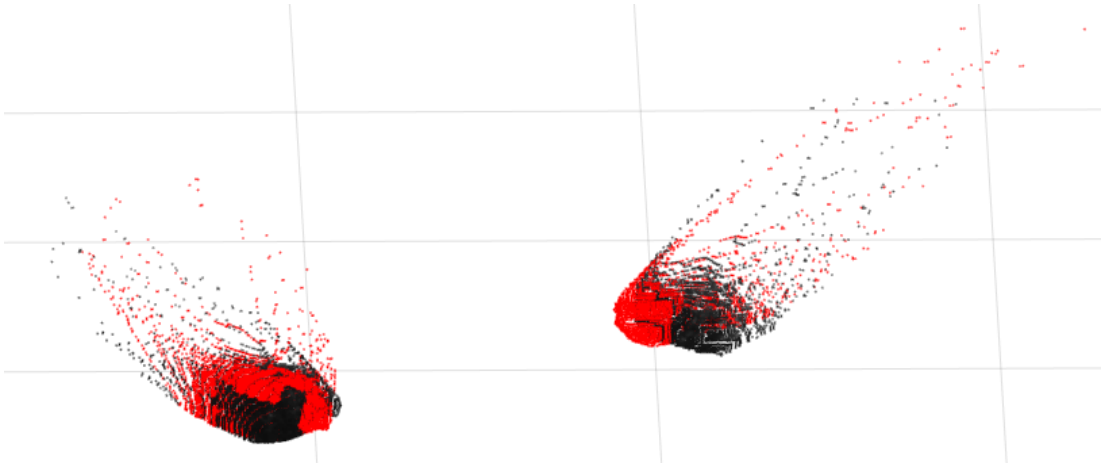


Figure 6.9: Rotation and Translation Error across 250 Samples

Some typical cases of successful registration results are shown in Figure 6.10. Here, the black point cloud is the static point cloud, and the red point cloud is the moving point cloud after registration using DUGMA. We could see there are strong noise and outliers at the edges. However, these outliers at the edges could contribute to a better registration result, especially when the shape is symmetric (See (b) Boxwood Ball in Figure 6.10).



(a) Hedge

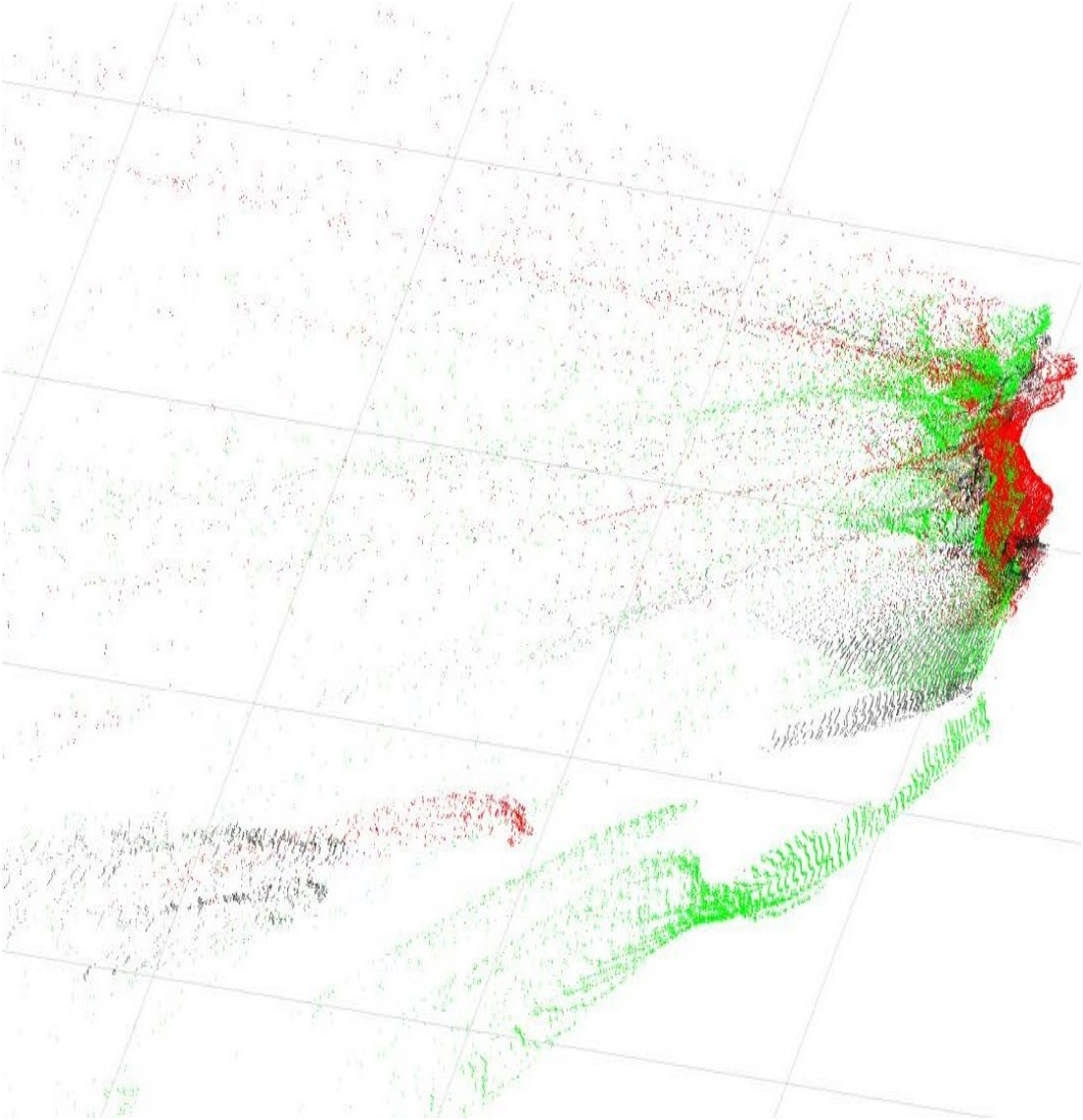


(b) Boxwood Ball

Figure 6.10: Some Successful Registration Samples

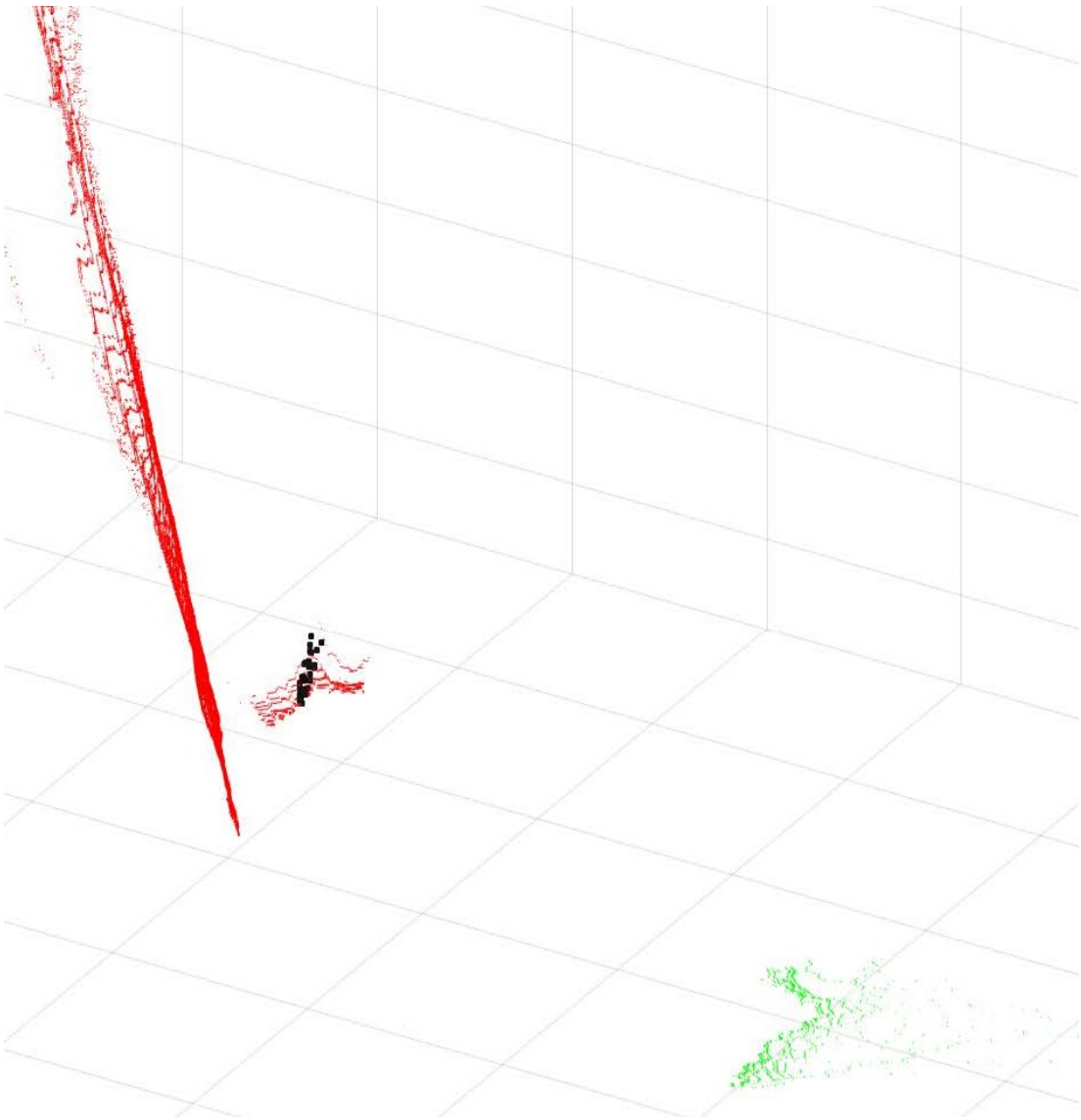
As for the unsuccessful registration samples, there are three main reasons. The first one is that the initial rotation angle is too big and DUGMA is a local registration algorithm. The second one is parameter setting, such as the definition of successful registration, the maximum iterations. The third one is that there is not enough overlapping areas for a successful registration. Some typical cases of unsuccessful registration results are shown in Figure 6.11. The black point cloud is the static point cloud and the red point cloud is the moving point cloud after registration using DUGMA. The green point cloud is the moving point cloud registered by the ground truth pose (so the black and green point cloud should overlap). In 'Failed Registration Sample 1', the registration error exceeded the successful registration definition by a small amount. In

'Failed Registration Sample 2', the static point cloud is one part of the moving point cloud and there is no overlapping areas between the static point cloud and the moving point cloud registered by the ground truth 6D pose. That is the reason why the estimated confidence is high, but the registration is completely wrong.



(a) Failed Registration Sample 1, Estimated $Confidence$=88.93%

Figure 6.11: *Cont.*



(b) Failed Registration Sample 2, Estimated $Confidence$=99.39%

Figure 6.11: Some Unsuccessful Registration Samples with High Estimated Confidence.

6.3 Conclusion

This chapter introduced the integration pipeline in a ROS system for TrimBot2020, and the real performance evaluation of the proposed supervised fusion algorithm and rigid point cloud registration for practical usage. The fusion pipeline includes the following: the rectified stereo intensity image input, getting two initial disparity maps from two stereo vision algorithms, fusing the initial disparity maps using our supervised disparity fusion method, converting the refined disparity map into 3D point cloud in the current view, registering multiple point clouds from different views with the proposed rigid

point cloud registration algorithm, and reconstructing the global 3D model of the whole garden. The real performance of the robot demonstrates the superiority of the proposed algorithms in this thesis, and the effectiveness of the fusion pipeline in the ROS system on the Trimbot. Succinctly, four proposed original algorithms (supervised & semi-supervised & unsupervised disparity fusion, point cloud registration) were introduced in the previous four chapters, and were evaluated on different datasets. Here, the proposed supervised disparity fusion algorithm and rigid point cloud registration algorithm were integrated into the navigation system of the Trimbot. The results demonstrate that the proposed approaches are not only successful with laboratory data, but also with real data. The code and technical report of the whole ROS pipeline on TrimBot2020 in this chapter will be released to the public to accelerate the commercial product development and related research progress. In the next chapter, overall conclusions about the whole thesis and a discussion on possible future work will be presented.

Chapter 7

Conclusion

In this chapter, a conclusion will be drawn about the whole thesis. As planned at the beginning, multiple initial depth maps were fused from different algorithms or sensors to obtain a refined depth map. The refined depth map was converted into a point cloud. Subsequently, a rigid point cloud registration algorithm was developed to obtain the relative 6D pose when there are many outliers, strong noise, and large occlusions. Finally, the whole pipeline was integrated into the ROS system on the robot called Trimbot.

A conclusion about the thesis accomplishments can be made first. This includes depth fusion, rigid point cloud registration, and practical application of the proposed algorithms. Second, the advantages and disadvantages of each proposed algorithm will be discussed. Finally, possible future extensions based on the current work will be presented.

7.1 Thesis Accomplishments and Critical Discussion

This thesis presents algorithms to solve existing problems in depth fusion and rigid point cloud registration. Further, the corresponding proposed methods have been integrated into the ROS system on a real robot to test their performance in a real application. More specifically, regarding depth fusion from multiple sources, it includes the supervised, semi-supervised, and unsupervised depth fusion algorithm. Regarding rigid point cloud registration, an uncertainty-based Gaussian mixture alignment has been developed to cope with the situation when there are many outliers, strong noise, and large occlusions. Besides the good performance achieved in the previous chapters, the algorithms also suffer from some problems. Regarding the advantages and disadvantages, a brief conclusion follows:

7.1.1 Depth Fusion

I have presented a supervised and semi-supervised depth fusion algorithm in Chapter 3 and unsupervised depth fusion algorithm in Chapter 4. These three algorithms are able to refine the initial depth maps to some extent. The energy functions to their refiner network are elegantly defined. All of them include the common smoothness term (Equation 3.3 or Equation 4.4), which can make the neural network generalise more easily and inpaint the black holes in the initial disparity map. However, the smoothness term also results in blurry phenomenon at edges. The proposed supervised and semi-supervised depth fusion method were compared with [14, 29, 59, 67, 68, 69, 73, 94, 97, 114] to achieve the state-of-art performance in general depth fusion area. The proposed unsupervised method is the first work to do unsupervised depth fusion and achieved the state-of-art performance compared with [29, 67, 68, 97, 114, 119] when realizing the stereo-stereo depth fusion.

Supervised Depth Fusion: The proposed supervised depth fusion can achieve a good performance; however, it requires a large quantity of labelled data to train. When this algorithm is adapted to a real environment, the neural network is usually trained on a synthetic dataset, and fine-tuned on a real dataset. By doing this, it can generalise better when the number of labelled data is limited.

Semi-supervised Depth Fusion: The proposed semi-supervised depth fusion can achieve a comparable performance to the supervised method, but with less labelled data. However, when training its neural network, it is usually slower than the supervised method, because additional training on unlabelled data would cost some extra time and

global memory on the GPU.

Unsupervised Depth Fusion: The proposed unsupervised depth fusion can generalise more successfully compared with the previous two methods, because it could use more real unlabelled data to train and there is no domain generation problem. However, it is difficult to use it in all the depth fusion tasks. In Equation 4.1, the confidence of each pixel w_{u_s} needs to be estimated in advance. An incorrect or coarse confidence estimation will influence the initialisation of the refined depth map to some extent, which will result in poor fusion results.

7.1.2 Rigid Point Cloud Registration

The uncertainty-based Gaussian mixture alignment in Chapter 5 improved the accuracy and robustness when there are many outliers, strong noise, and large occlusions. The proposed algorithm achieved the state-of-art performance once (Nov. 2017) compared with the top algorithms [28, 76, 86, 103, 156, 162, 169]). Currently HGMR [45] achieved the state-of-art performance. Additionally, the computation cost of the proposed algorithm is large, which precludes it from being used in real-time, even on a GPU (Nvidia GTX1080Ti). The current proposed algorithm is sensitive to the local minimum, although it enjoys a wider convergence basin compared with the traditional local algorithms. Additionally, uncertainty estimation is difficult, which limits the generality of this algorithm.

7.1.3 Practical Usage on the Real Robot

When the depth acquisition method [97] and the supervised depth fusion method were tuned, it was time-consuming to tune the parameters to make them generalise from the synthetic datasets to the real environment. Additionally, limited by the fixed focal length and baseline of the stereo vision cameras, the valid working distance was within 10 m, which could not recover the structure of the remote parts of the scenes. Although some disparity post-processing at the edges was conducted, there were still some misestimated depths at the edge. Those outliers will become obvious when the refined depth map is projected to a point cloud.

Given that the point cloud registration algorithm [118] is slow, it is impossible to run the whole pipeline in real-time. Except for the point cloud registration component, the other components could achieve real-time performance. The structure of the plants in the real garden is usually symmetric (eg: boxwood ball), which decreases the performance

of the point cloud registration algorithm because of the local minimum. Additionally, the accumulated error will increase drastically when the rigid point cloud registration algorithm totally fails.

7.2 Possible Future Extensions

In the near future, the current work could be extended by changing the input data quality (depending on the sensor) and the algorithms.

7.2.1 Sensor

Different sensors have different properties, which will influence the system overall performance. Integration of multiple dedicated sensors is a good solution to a robust production. For example, in the near future, integration of the ToF and stereo vision sensor is a good option to achieve a cheap and highly accurate indoor depth sensor. Different choices for the baseline and focal length in the stereo vision camera will result in different valid working distances.

7.2.2 Algorithm

Depth Fusion: The depth fusion algorithms will be extended to more kinds of depth fusion for different scenes, such as ToF-stereo depth fusion and Ultrasonic-stereo depth fusion. Currently, the depth fusion algorithm is single-task learning, which can only output depth information. The information in different domains (such as semantic meaning, depth, and surface normal) share common information (to some extent). To save computation resources and utilise the common information among different domains, the current depth fusion algorithms in this thesis could possibly be extended to the multi-task learning problem by outputting the semantic meaning, surface normal, and depth. Additionally, it is not evident whether the proposed adversarial networks in this thesis suffer from the mode collapse problem, which will be investigated in the future research.

Rigid Point Cloud Registration: The current rigid point cloud registration algorithm in this thesis only uses the 3D position of each point for registration. In actuality, more information can be added into the cost function. For example, the RGB information, surface normal, or semantic meaning information could be regarded as features. By considering feature matching and probability alignment simultaneously, the registration

accuracy could be improved (in my opinion). Additionally, the downsampling of the initial point cloud would also influence the registration speed and accuracy. The semantic meaning could be utilised to achieve different downsampling for different objects. For example, different sampling rates could be used for ground or trees in the garden. Finally, the initialisation of the 6D pose is also important. Using the existing hardware (such as an IMU), to provide a coarse global initialised 6D pose to the registration algorithm is a good solution to help avoid local minima and increase the speed.

The Current System: The current system in Chapter 6 can not work in all situations. For example, when the rigid point cloud registration component fails in some special cases, the system has a risk of breaking down. In the near future, fusing multiple 6D poses from different sources (such as structure from motion, IMU, and RGB-D slam) is a robust and accurate solution in real environments.

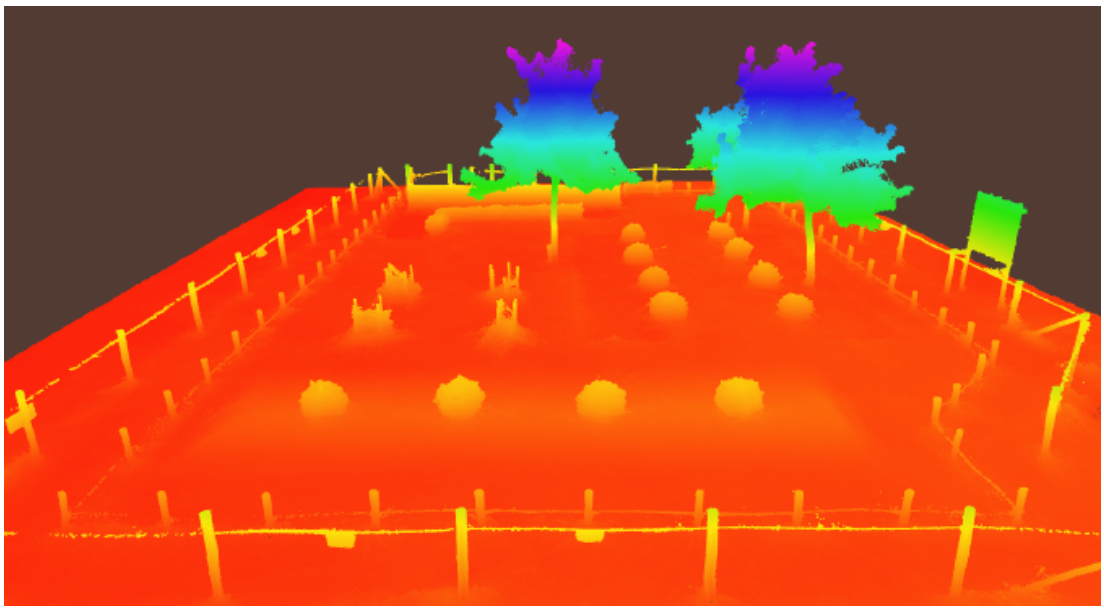
Appendix A

Dataset Description

A.1 Description of Trimbot2020 Garden Dataset

We make use of the TrimBot2020 Garden 2017 dataset, which was used for the semantic reconstruction challenge of the 2017 ICCV (International Conference on Computer Vision) workshop '3D Reconstruction meets Semantics' [129]. The dataset consists of a 3D laser scan of the garden as well as multiple traversals of the robot through the garden (see Figure A.1). In addition to the challenge dataset (2 camera pairs), we included all 5 camera pairs (Figure A.2) , obtaining total 1270 sample pairs. Robot poses for the traversals were recorded in the coordinate system of the laser scanner using a Topcon laser tracker. The results were subsequently refined using Structure-from-Motion [132]. The quantitative evaluation is performed only on a subset of pixels which correspond to static non-ground areas (the grass on the ground surface yields noisy GT measurements as well as other moving parts like tree branches).

The accuracy of stereo depth map estimates depends on the distance of the cameras to the scene, with the uncertainty growing quadratically with the distance. In contrast, the uncertainty grows only linearly in the disparity space (measured in pixels). As is common [56, 133], we measured the accuracy of the stereo algorithms by comparing their estimated disparity values with the ground truth disparity values provided by the laser scanner.



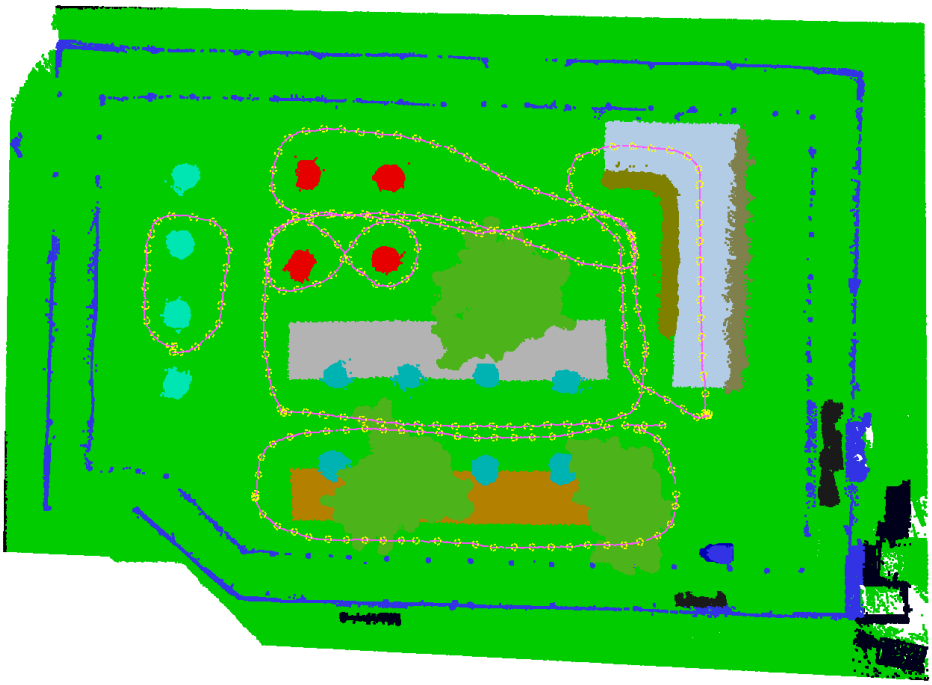


Figure A.1: Trimbot Garden 2017 GT dataset [129]. **Above**: point cloud with color-encoded height. **Below**: semantic point cloud with trajectories (magenta line) and camera centers (yellow).

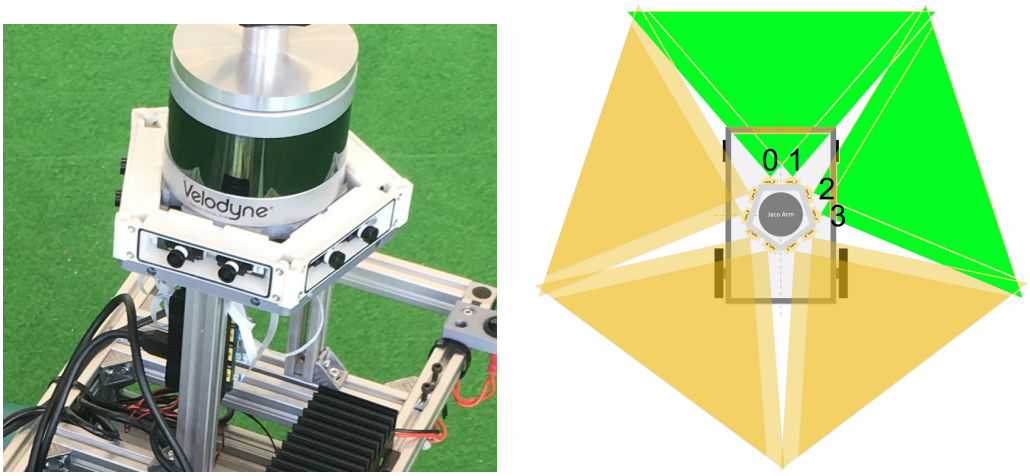


Figure A.2: Trimbot Garden 2017 GT dataset [129]. **Left**: Pentagonal camera rig mounted on the robot with five stereo pairs. **Right**: Top view of camera rig with test set pairs (green field of veiw) and training set pairs (yellow field of view).

Appendix B

DUGMA in 2D Registration

The method proposed in Chapter 5 can be also used for 2D registration. 14,700 trials (similar to those in Section 5.3.1) have been conducted using 100 2D fish models from the Gatorbait100[1] database, and one hundred 2D point clouds with different shapes have been used (such as face, umbrella, and computer) to test sensitivity to shapes. All the results are equally robust.

B.1 Simulation on 2D Fish Model

Similar to the simulation on the 3D models, 14,700 similar trials have been conducted using 100 2D fish models from the Gatorbait100 database. Here, approximately 100 points were randomly extracted from each initial image as the 2D point cloud model. Figure B.1 shows 6 examples with completely different fish shapes from the 100 2D fish models.

[1] A. Rangarajan. (14 November. 2017). https://www.cise.ufl.edu/~anand/publications.html

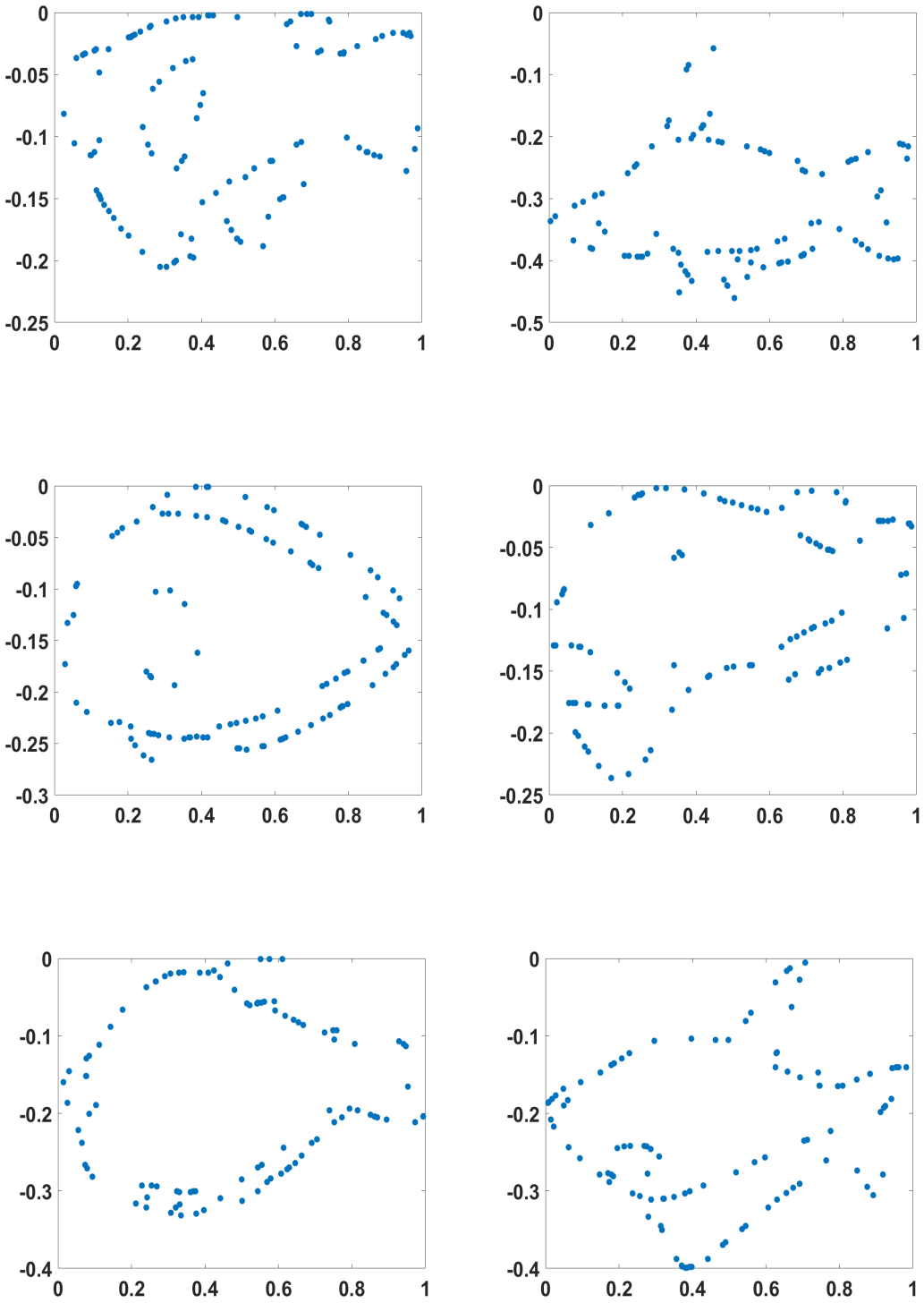
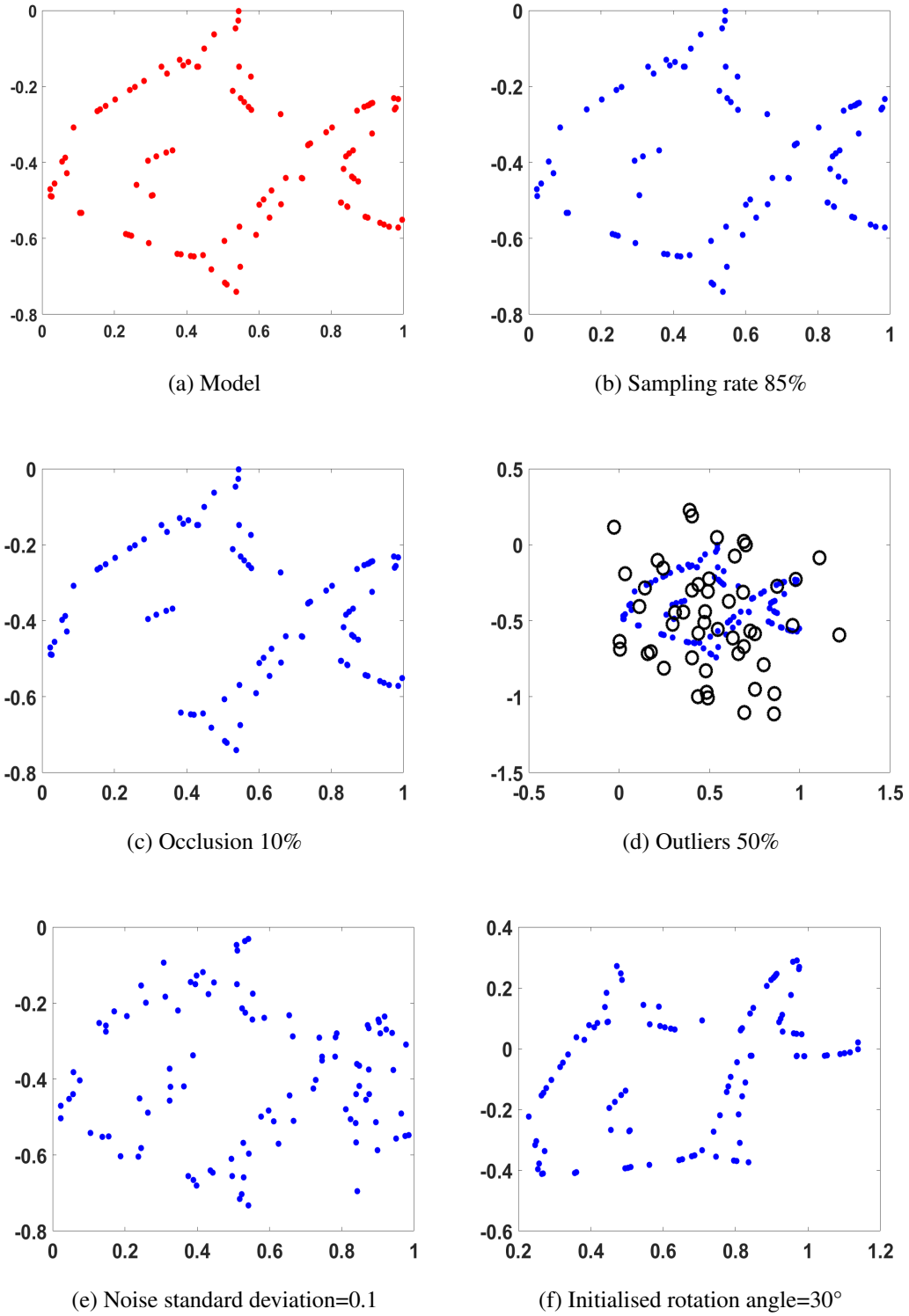


Figure B.1: 2D models of various fish with different shapes.

Figure B.2 illustrates the effect from the various factors which will be explored to assess the performance of the algorithm. In all 2D experiments, the sampling rate was set to 90% and 85% for the fixed and moving point clouds, respectively. The remaining

factors were drawn from the uniform distribution from Table B.1 randomly when they were not the controlled variable.



(a) Model

(b) Sampling rate 85%

(c) Occlusion 10%

(d) Outliers 50%

(e) Noise standard deviation=0.1

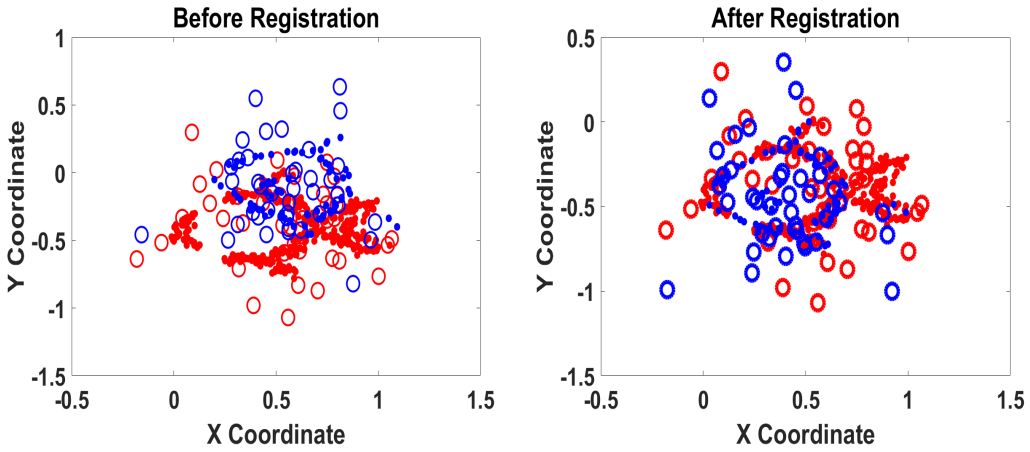
(f) Initialised rotation angle=30°

Figure B.2: Different influences from various factors.

Table B.1: Range for random factors in 2D part.

Initial rotation angle	[-15°, 15°]
Outliers	[0, 50] that is, [0, 50%]
Noise standard deviation	[0, 0.05] times radius of point cloud
Occlusion	[0, 0.05] that is: [0, 5%]

Figure B.3 shows a pair of point clouds before and after registration.



(a) Before Registration (b) After registration

Figure B.3: The scene before and after registration.

When the rotation is the controlled variable, it will range from [-60°, 60°] and the step is equal to 8°. Figure B.4 shows the estimated error (mean error and its standard deviation) with the change of the initial rotation. It is obvious that from [-40°, 40°] the accuracy and robustness of CPD and our method are nearly the same, while the standard ICP behaves less effectively. If the maximum iteration value is increased (set to 200 rather than 100), the proposed method will become more accurate and robust from [-40°, 40°] while CPD and standard ICP cannot, in that they obtain their local minimum. Beyond -40°or 40°, the proposed algorithm finds a local minimum and breaks down.

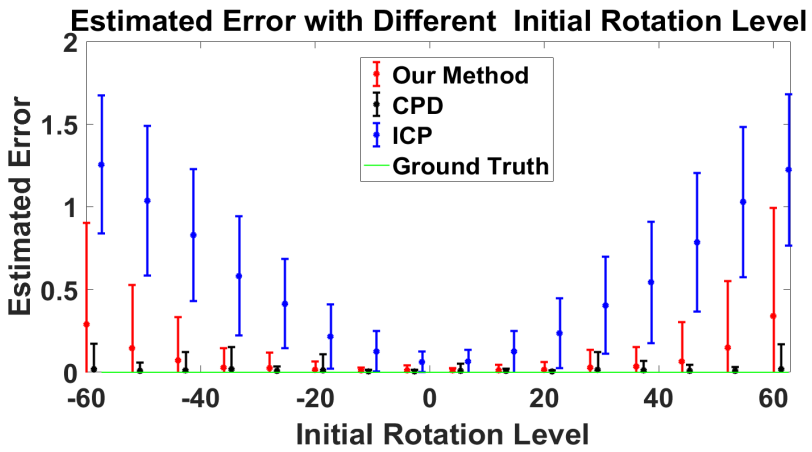


Figure B.4: Standard deviation of estimated error versus initial rotations.

When noise level is the controlled variable, it will range from [0, 0.30] and its step is equal to 0.03. From Fig B.5, it is evident that with the increase of noise level, the error increases slightly. However, it is still very accurate and robust compared with CPD, because the variances of all the noise on each axis have been stored in the covariances, which will be used by the proposed system to estimate the error for each point.

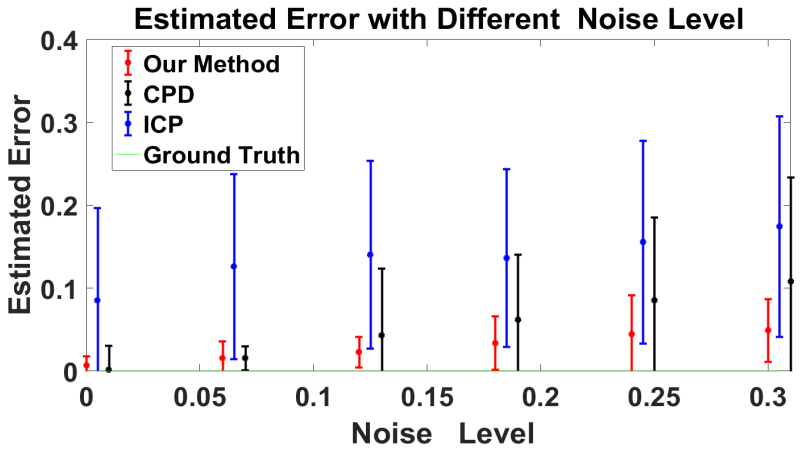


Figure B.5: Estimated error with different noise levels.

When occlusion is the controlled variable, it ranges from [0, 0.3] (from 0 to 30%) and the step size is equal to 0.03. The random occlusion part affects the proposed model significantly when the occlusion rate is greater than 15%, because the missing long segments make it easy to converge to local minima, as shown in Figure B.6.

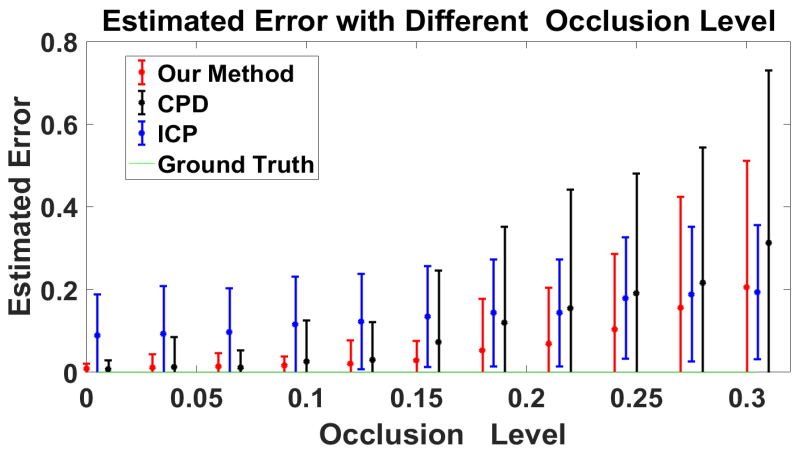


Figure B.6: Estimated error with different occlusion levels.

When the number of outliers is the controlled variable, it ranges from [20, 200] (that is, from 20% to 200%) and the step is 20. The covariance for outliers is allowed to be very large to represent that they have a very low certainty. By doing this, the outliers will be filtered by the proposed system automatically. Therefore, the experimental result looks much better than the comparison algorithms, as shown in Figure B.7.

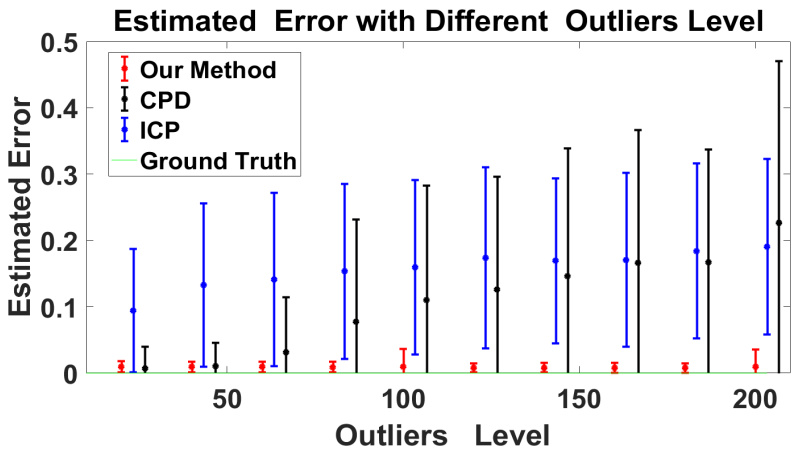


Figure B.7: Estimated error with different outlier levels.

In the 2D part, 14,700 trials were attempted, and the total time for each algorithm is listed in table B.2. Standard ICP used the minimum time, mainly because it initially dropped in the local minimum.

Table B.2: Total runing time in 2D part.

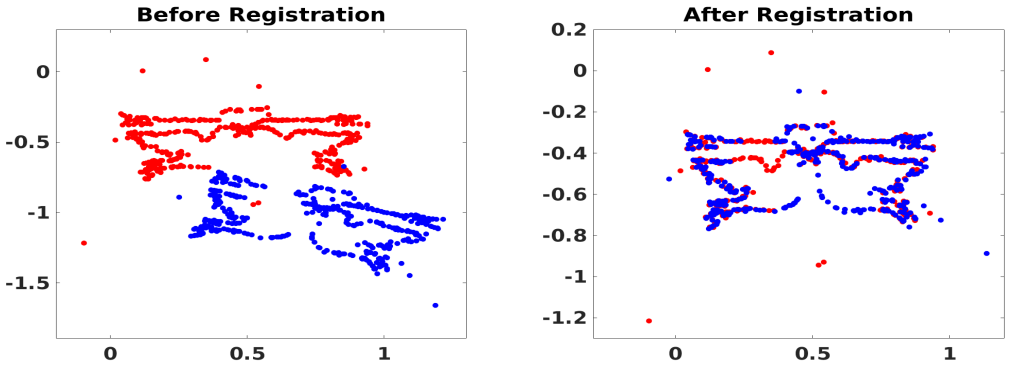
ICP	54.30 sec
CPD	31.10 min
Our Method	432.12 min

B.2 Simulation on 100 Different 2D Models

Now, an additional 100 different 2D shapes will be used to test the robustness to different shapes in 2D registration. The random parameters are set as follows (See Table B.3). More qualitative results can be observed in Figure B.8.

Table B.3: Range for random factors for More Different 2D Models.

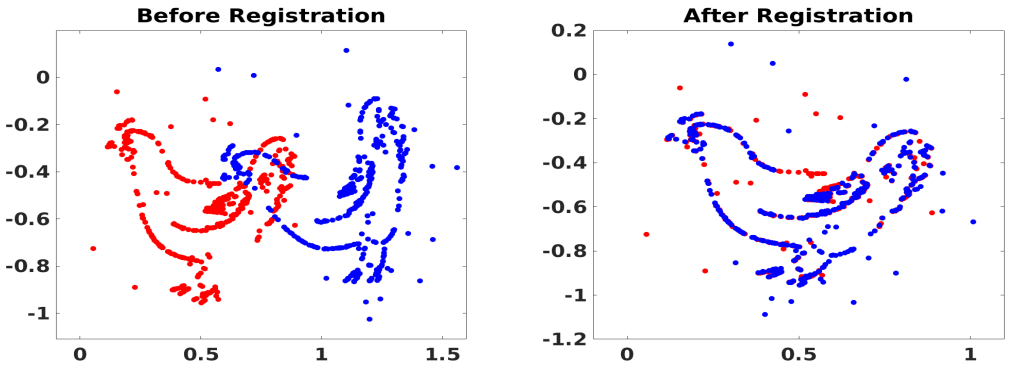
Initial rotation angle	[-30°, 30°]
Outliers	[0, 5%]
Noise standard deviation	[0, 10%] times radius of point cloud
Occlusion	[0, 10%]



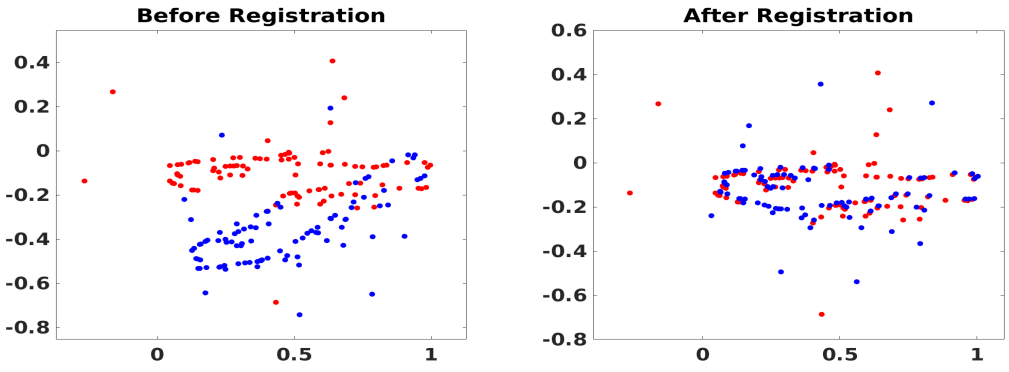
(a) Butterfly Input

(b) Butterfly Output

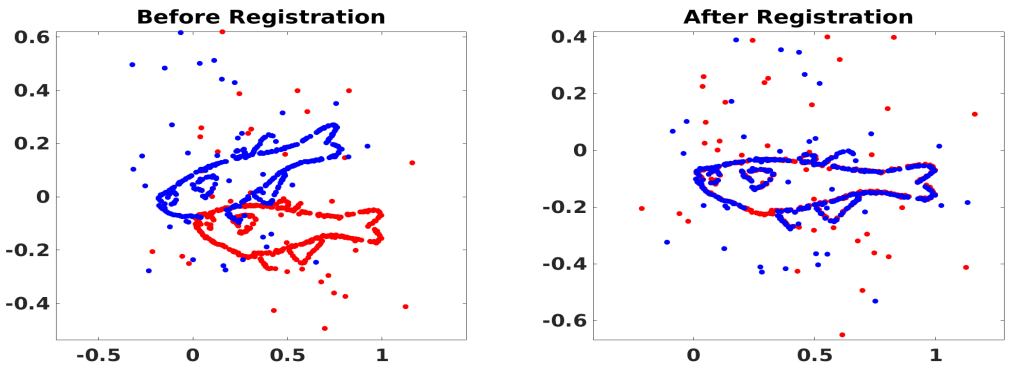
Figure B.8: *Cont.*



(c) Cock Input (d) Cock Output



(e) Sparse Input (f) Sparse Output



(g) Dense Input (h) Dense Output

Figure B.8: Some successful registrations in the 2D plane.

Regarding each shape, 10 trials were run. A model was chosen from '2D_SHAPE_MODEL' in the new dataset, and transformed by the random factors above in each trial. The ex-

perimental results again show that the algorithm is robust to different shapes, as shown in Figure B.9. The x-axis is the order of the shape and the y-axis is the corresponding error. 5 models have much larger errors compared with the rest because their shapes are symmetric and DUGMA got to local minima.

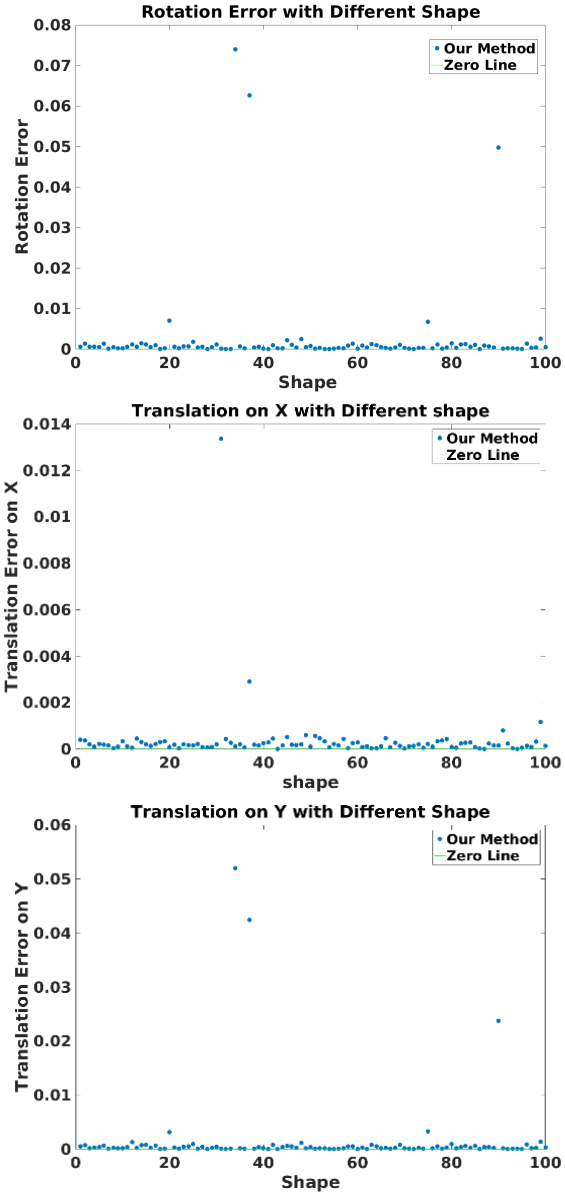


Figure B.9: Robustness Test to 2D Shapes

Bibliography

[1] The code for pix2pix algorithm. https://phillipi.github.io/pix2pix/. Accessed: 2019-01-13.

[2] Depth map description in wikipedia. https://en.wikipedia.org/wiki/Depth_map. Accessed: 2019-07-23.

[3] Function 'disparity' in matlab software. https://www.mathworks.com/help/vision/ref/disparity.html. Accessed: 2019-01-13.

[4] Lidar sensor description. https://en.wikipedia.org/wiki/Lidar. Accessed: 2019-01-13.

[5] Markov random field description in wikipedia. https://en.wikipedia.org/wiki/Markov_random_field. Accessed: 2019-07-23.

[6] Opencv library. https://opencv.org. Accessed: 2019-01-13.

[7] Project wesite of 3dmatch. http://3dmatch.cs.princeton.edu/. Accessed: 2019-01-13.

[8] Radar sensor description. https://en.wikipedia.org/wiki/Radar. Accessed: 2019-01-13.

[9] Sigmoid function in tensorflow platform. https://www.tensorflow.org/api_docs/python/tf/math/sigmoid. Accessed: 2019-01-13.

[10] Structured-light sensor description. https://en.wikipedia.org/wiki/Structured-light_3D_scanner. Accessed: 2019-01-13.

[11] Tof sensor description. https://en.wikipedia.org/wiki/Time-of-flight_camera. Accessed: 2019-01-13.

[12] Ultrasonic sensor description. https://en.wikipedia.org/wiki/Ultrasonic_transducer. Accessed: 2019-01-13.

[13] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: A system for large-scale machine learning. In *OSDI*, volume 16, pages 265–283, 2016.

[14] Gianluca Agresti, Ludovico Minto, Giulio Marin, and Pietro Zanuttigh. Deep learning for confidence information in stereo and tof data fusion. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 697–705, 2017.

[15] Gianluca Agresti, Ludovico Minto, Giulio Marin, and Pietro Zanuttigh. Stereo and tof data fusion by learning from synthetic data. *Information Fusion*, 49:161–173, 2019.

[16] Gianluca Agresti, Henrik Schaefer, Piergiorgio Sartor, and Pietro Zanuttigh. Unsupervised domain adaptation for tof data denoising with adversarial learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5584–5593, 2019.

[17] Gianluca Agresti, Henrik Schaefer, Piergiorgio Sartor, and Pietro Zanuttigh. Unsupervised domain adaptation for tof data denoising with adversarial learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5584–5593, 2019.

[18] Filippo Aleotti, Fabio Tosi, Matteo Poggi, and Stefano Mattoccia. Generative adversarial networks for unsupervised monocular depth prediction. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 0–0, 2018.

[19] Yasin Almalioglu, Muhamad Risqi U Saputra, Pedro PB de Gusmao, Andrew Markham, and Niki Trigoni. Ganvo: Unsupervised deep monocular visual odometry and depth estimation with generative adversarial networks. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 5474–5480. IEEE, 2019.

[20] Yasuhiro Aoki, Hunter Goforth, Rangaprasad Arun Srivatsan, and Simon Lucey. Pointnetlk: Robust & efficient point cloud registration using pointnet. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7163–7172, 2019.

[21] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *International Conference on Machine Learning*, pages 214–223, 2017.

[22] Amira Belhedi, Adrien Bartoli, Steve Bourgeois, Kamel Hamrouni, Patrick Sayd, and Vincent Gay-Bellile. Noise modelling and uncertainty propagation for tof sensors. In *European Conference on Computer Vision*, pages 476–485. Springer, 2012.

[23] Paul J Besl, Neil D McKay, et al. A method for registration of 3-d shapes. *IEEE Transactions on pattern analysis and machine intelligence*, 14(2):239–256, 1992.

[24] Christopher M Bishop. *Neural networks for pattern recognition*. Oxford university press, 1995.

[25] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale GAN training for high fidelity natural image synthesis. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*, 2019.

[26] Álvaro Parra Bustos and Tat-Jun Chin. Guaranteed outlier removal for point cloud registration with correspondences. *IEEE transactions on pattern analysis and machine intelligence*, 40(12):2868–2882, 2018.

[27] Dylan Campbell and Lars Petersson. An adaptive data representation for robust point-set registration and merging. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4292–4300, 2015.

[28] Dylan Campbell and Lars Petersson. Gogma: Globally-optimal gaussian mixture alignment. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5685–5694, 2016.

[29] Jia-Ren Chang and Yong-Sheng Chen. Pyramid stereo matching network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5410–5418, 2018.

[30] Baoliang Chen, Cheolkon Jung, and Zhendong Zhang. Variational fusion of time-of-flight and stereo data for depth estimation using edge-selective joint filtering. *IEEE Transactions on Multimedia*, 20(11):2882–2890, 2018.

[31] Po-Yi Chen, Alexander H Liu, Yen-Cheng Liu, and Yu-Chiang Frank Wang. Towards scene understanding: Unsupervised monocular depth estimation with semantic-aware representation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2624–2632, 2019.

[32] Yang Chen and Gérard Medioni. Object modelling by registration of multiple range images. *Image and vision computing*, 10(3):145–155, 1992.

[33] Xuelian Cheng, Yiran Zhong, Yuchao Dai, Pan Ji, and Hongdong Li. Noise-aware unsupervised deep lidar-stereo fusion. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6339–6348, 2019.

[34] Dmitry Chetverikov, Dmitry Svirko, Dmitry Stepanov, and Pavel Krsek. The trimmed iterative closest point algorithm. In *Pattern Recognition, 2002. Proceedings. 16th International Conference on*, volume 3, pages 545–548. IEEE, 2002.

[35] Haili Chui and Anand Rangarajan. A new point matching algorithm for non-rigid registration. *Computer Vision and Image Understanding*, 89(2):114–141, 2003.

[36] Antonia Creswell, Tom White, Vincent Dumoulin, Kai Arulkumaran, Biswa Sengupta, and Anil A Bharath. Generative adversarial networks: An overview. *IEEE Signal Processing Magazine*, 35(1):53–65, 2018.

[37] Arun CS Kumar, Suchendra M Bhandarkar, and Mukta Prasad. Monocular depth prediction using generative adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 300–308, 2018.

[38] Carlo Dal Mutto, Pietro Zanuttigh, and Guido Maria Cortelazzo. Probabilistic tof and stereo data fusion based on mixed pixels measurement models. *IEEE transactions on pattern analysis and machine intelligence*, 37(11):2260–2272, 2015.

[39] Carlo Dal Mutto, Pietro Zanuttigh, and Guido Maria Cortelazzo. Probabilistic tof and stereo data fusion based on mixed pixels measurement models. *IEEE transactions on pattern analysis and machine intelligence*, 37(11):2260–2272, 2015.

[40] Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the royal statistical society. Series B (methodological)*, pages 1–38, 1977.

[41] Haowen Deng, Tolga Birdal, and Slobodan Ilic. Ppf-foldnet: Unsupervised learning of rotation invariant 3d local descriptors. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 602–618, 2018.

[42] Haowen Deng, Tolga Birdal, and Slobodan Ilic. 3d local features for direct pairwise registration. *arXiv preprint arXiv:1904.04281*, 2019.

[43] Emily L Denton, Soumith Chintala, Rob Fergus, et al. Deep generative image models using a laplacian pyramid of adversarial networks. In *Advances in neural information processing systems*, pages 1486–1494, 2015.

[44] Alexey Dosovitskiy, Philipp Fischer, Eddy Ilg, Philip Hausser, Caner Hazirbas, Vladimir Golkov, Patrick Van Der Smagt, Daniel Cremers, and Thomas Brox. *Proc* ks.
the IEEE international conference on computer vision, pages 2758–2766, 2015.

[45] Benjamin Eckart, Kihwan Kim, and Jan Kautz. Hgmr: Hierarchical gaussian mixtures for adaptive 3d registration. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 705–721, 2018.

[46] David Eigen, Christian Puhrsch, and Rob Fergus. Depth map prediction from a single image using a multi-scale deep network. In *Advances in neural information processing systems*, pages 2366–2374, 2014.

[47] Gil Elbaz, Tamar Avraham, and Anath Fischer. 3d point cloud registration for localization using a deep neural network auto-encoder. In *CVPR*, 2017.

[48] Abdelrahman Eldesokey, Michael Felsberg, and Fahad Shahbaz Khan. Confidence propagation through cnns for guided sparse depth regression. *IEEE transactions on pattern analysis and machine intelligence*, 2019.

[49] Jakob Engel, Vladlen Koltun, and Daniel Cremers. Direct sparse odometry. *IEEE transactions on pattern analysis and machine intelligence*, 40(3):611–625, 2018.

[50] Jakob Engel, Thomas Schöps, and Daniel Cremers. Lsd-slam: Large-scale direct monocular slam. In *European conference on computer vision*, pages 834–849. Springer, 2014.

[51] Jakob Engel, Jurgen Sturm, and Daniel Cremers. Semi-dense visual odometry for a monocular camera. In *Proceedings of the IEEE international conference on computer vision*, pages 1449–1456, 2013.

[52] José M Fácil, Alejo Concha, Luis Montesano, and Javier Civera. Single-view and multi-view depth fusion. *IEEE Robotics and Automation Letters*, 2(4):1994–2001, 2017.

[53] Martin A Fischler and Robert C Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[54] Andrew W Fitzgibbon. Robust registration of 2d and 3d point sets. *Image and Vision Computing*, 21(13):1145–1153, 2003.

[55] Charles Freundlich, Michael Zavlanos, and Philippos Mordohai. Exact bias correction and covariance estimation for stereo vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3296–3304, 2015.

[56] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[57] Jason Geng. Structured-light 3d surface imaging: a tutorial. *Advances in Optics and Photonics*, 3(2):128–160, 2011.

[58] Silvio Giancola, Matteo Valenti, and Remo Sala. Metrological qualification of the kinect v2™ time-of-flight camera. In *A Survey on 3D Cameras: Metrological Comparison of Time-of-Flight, Structured-Light and Active Stereoscopy Technologies*, pages 41–60. Springer, 2018.

[59] Clément Godard, Oisin Mac Aodha, and Gabriel J Brostow. Unsupervised monocular depth estimation with left-right consistency. In *CVPR*, volume 2, page 7, 2017.

[60] Steven Gold, Chien-Ping Lu, Anand Rangarajan, Suguna Pappu, and Eric Mjolsness. New algorithms for 2d and 3d point matching: Pose estimation and correspondence. In *Advances in Neural Information Processing Systems*, pages 957–964, 1995.

[61] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[62] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of wasserstein gans. In *Advances in Neural Information Processing Systems*, pages 5769–5779, 2017.

[63] Kin Gwn Lore, Kishore Reddy, Michael Giering, and Edgar A Bernal. Generative adversarial networks for depth map estimation from rgb video. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 1177–1185, 2018.

[64] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, second edition, 2004.

[65] Lionel Heng, Benjamin Choi, Zhaopeng Cui, Marcel Geppert, Sixing Hu, Benson Kuan, Peidong Liu, Rang Nguyen, Ye Chuan Yeo, Andreas Geiger, et al. Project autovision: Localization and 3d scene perception for an autonomous vehicle with a multi-camera system. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 4695–4702. IEEE, 2019.

[66] George Heritage and Andy Large. *Laser scanning for the environmental sciences*. John Wiley & Sons, 2009.

[67] Heiko Hirschmuller. Accurate and efficient stereo processing by semi-global matching and mutual information. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 2, pages 807–814. IEEE, 2005.

[68] Dominik Honegger, Torsten Sattler, and Marc Pollefeys. Embedded real-time multi-baseline stereo. In *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, pages 5245–5250. IEEE, 2017.

[69] Luis Horna and Robert B Fisher. 3d plane labeling stereo matching with content aware adaptive windows. In *VISIGRAPP (6: VISAPP)*, pages 162–171, 2017.

[70] Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[71] Po-Han Huang, Kevin Matzen, Johannes Kopf, Narendra Ahuja, and Jia-Bin Huang. Deepmvs: Learning multi-view stereopsis. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[72] Du Q Huynh. Metrics for 3d rotations: Comparison and analysis. *Journal of Mathematical Imaging and Vision*, 35(2):155–164, 2009.

[73] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. *CVPR*, 2017.

[74] Felix Järemo Lawin, Martin Danelljan, Fahad Shahbaz Khan, Per-Erik Forssén, and Michael Felsberg. Density adaptive point set registration. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3829–3837, 2018.

[75] Maximilian Jaritz, Raoul De Charette, Emilie Wirbel, Xavier Perrotton, and Fawzi Nashashibi. Sparse and dense data with cnns: Depth completion and semantic segmentation. In *2018 International Conference on 3D Vision (3DV)*, pages 52–60. IEEE, 2018.

[76] Bing Jian and Baba C Vemuri. Robust point set registration using gaussian mixture models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(8):1633–1645, 2011.

[77] Alexia Jolicoeur-Martineau. The relativistic discriminator: a key element missing from standard GAN. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*, 2019.

[78] Takeo Kanade and Masatoshi Okutomi. A stereo matching algorithm with an adaptive window: Theory and experiment. In *Proceedings. 1991 IEEE International Conference on Robotics and Automation*, pages 1088–1095. IEEE, 1991.

[79] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*, 2018.

[80] Michal Kelemen, Ivan Virgala, Tatiana Kelemenová, L'ubica Miková, Peter Frankovskỳ, Tomáš Lipták, and Milan Lörinc. Distance measurement via using of ultrasonic sensor. *Journal of automation and control*, 3(3):71–74, 2015.

[81] Alex Kendall, Hayk Martirosyan, Saumitro Dasgupta, Peter Henry, Ryan Kennedy, Abraham Bachrach, and Adam Bry. End-to-end learning of geometry and context for deep stereo regression. In *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[82] Leonid Keselman, John Iselin Woodfill, Anders Grunnet-Jepsen, and Achintya Bhowmik. Intel realsense stereoscopic depth cameras. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 1–10, 2017.

[83] Sunok Kim, Seungryong Kim, Dongbo Min, and Kwanghoon Sohn. Laf-net: Locally adaptive fusion networks for stereo confidence estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 205–214, 2019.

[84] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[85] Yann LeCun, Léon Bottou, Yoshua Bengio, Patrick Haffner, et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[86] Huan Lei, Guang Jiang, and Long Quan. Fast descriptors and correspondence propagation for robust global point cloud registration. *IEEE Transactions on Image Processing*, 2017.

[87] Marc Levoy, J Gerth, B Curless, and K Pull. The stanford 3d scanning repository. *URL http://www-graphics. stanford. edu/data/3dscanrep*, 2005.

[88] Zhengfa Liang, Yiliu Feng, Yulan Guo, Hengzhu Liu, Wei Chen, Linbo Qiao, Li Zhou, and Jianfeng Zhang. Learning for disparity estimation through feature constancy. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2811–2820, 2018.

[89] Yinlong Liu, Chen Wang, Zhijian Song, and Manning Wang. Efficient global point cloud registration by matching rotation invariant features through translation search. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 448–463, 2018.

[90] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

[91] Bruce D Lucas, Takeo Kanade, et al. An iterative image registration technique with an application to stereo vision. 1981.

[92] Will Maddern and Paul Newman. Real-time probabilistic fusion of sparse 3d lidar and dense stereo. In *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*, pages 2181–2188. IEEE, 2016.

[93] Xudong Mao, Qing Li, Haoran Xie, Raymond YK Lau, Zhen Wang, and Stephen Paul Smolley. Least squares generative adversarial networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2794–2802, 2017.

[94] Giulio Marin, Pietro Zanuttigh, and Stefano Mattoccia. Reliable fusion of tof and stereo depth driven by confidence measures. In *European Conference on Computer Vision*, pages 386–401. Springer, 2016.

[95] Giulio Marin, Pietro Zanuttigh, and Stefano Mattoccia. Reliable fusion of tof and stereo depth driven by confidence measures. In *European Conference on Computer Vision*, pages 386–401. Springer, 2016.

[96] Larry Matthies and STEVENA Shafer. Error modeling in stereo navigation. *IEEE Journal on Robotics and Automation*, 3(3):239–248, 1987.

[97] Nikolaus Mayer, Eddy Ilg, Philip Hausser, Philipp Fischer, Daniel Cremers, Alexey Dosovitskiy, and Thomas Brox. A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In *Proceedings*

of the IEEE Conference on Computer Vision and Pattern Recognition, pages 4040–4048, 2016.

[98] Moritz Menze, Christian Heipke, and Andreas Geiger. Joint 3d estimation of vehicles and scene flow. In *ISPRS Workshop on Image Sequence Analysis (ISA)*, 2015.

[99] Luke Metz, Ben Poole, David Pfau, and Jascha Sohl-Dickstein. Unrolled generative adversarial networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, 2017.

[100] Zhe Min and Max Q-H Meng. Robust generalized point set registration using inhomogeneous hybrid mixture models via expectation maximization. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 8733–8739. IEEE, 2019.

[101] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*, 2018.

[102] Raul Mur-Artal, Jose Maria Martinez Montiel, and Juan D Tardos. Orb-slam: a versatile and accurate monocular slam system. *IEEE transactions on robotics*, 31(5):1147–1163, 2015.

[103] Andriy Myronenko and Xubo Song. Point set registration: Coherent point drift. *IEEE transactions on pattern analysis and machine intelligence*, 32(12):2262–2275, 2010.

[104] Rahul Nair, Kai Ruhl, Frank Lenzen, Stephan Meister, Henrik Schäfer, Christoph S Garbe, Martin Eisemann, Marcus Magnor, and Daniel Kondermann. A survey on time-of-flight stereo fusion. In *Time-of-Flight and Depth Imaging. Sensors, Algorithms, and Applications*, pages 105–127. Springer, 2013.

[105] Chuong V Nguyen, Shahram Izadi, and David Lovell. Modeling kinect sensor noise for improved 3d reconstruction and tracking. In *3D Imaging, Modeling, Processing, Visualization and Transmission (3DIMPVT), 2012 Second International Conference on*, pages 524–530. IEEE, 2012.

[106] Guang-Yu Nie, Ming-Ming Cheng, Yun Liu, Zhengfa Liang, Deng-Ping Fan, Yue Liu, and Yongtian Wang. Multi-level context ultra-aggregation for stereo matching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3283–3291, 2019.

[107] Luis Enrique Ortiz, Elizabeth V Cabrera, and Luiz M Gonçalves. Depth data error modeling of the zed 3d vision sensor from stereolabs. *ELCVIA: electronic letters on computer vision and image analysis*, 17(1):0001–15, 2018.

[108] Zhaoqing Pan, Weijie Yu, Xiaokai Yi, Asifullah Khan, Feng Yuan, and Yuhui Zheng. Recent progress on generative adversarial networks (gans): A survey. *IEEE Access*, 7:36322–36333, 2019.

[109] Kihong Park, Seungryong Kim, and Kwanghoon Sohn. High-precision depth estimation with the 3d lidar and stereo fusion. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2156–2163. IEEE, 2018.

[110] Kihong Park, Seungryong Kim, and Kwanghoon Sohn. High-precision depth estimation using uncalibrated lidar and stereo fusion. *IEEE Transactions on Intelligent Transportation Systems*, 2019.

[111] Kihong Park, Seungryong Kim, and Kwanghoon Sohn. High-precision depth estimation using uncalibrated lidar and stereo fusion. *IEEE Transactions on Intelligent Transportation Systems*, 2019.

[112] Andrea Pilzer, Dan Xu, Mihai Puscas, Elisa Ricci, and Nicu Sebe. Unsupervised adversarial depth estimation using cycled generative networks. In *2018 International Conference on 3D Vision (3DV)*, pages 587–595. IEEE, 2018.

[113] Andrea Pilzer, Dan Xu, Mihai Puscas, Elisa Ricci, and Nicu Sebe. Unsupervised adversarial depth estimation using cycled generative networks. In *2018 International Conference on 3D Vision (3DV)*, pages 587–595. IEEE, 2018.

[114] Matteo Poggi and Stefano Mattoccia. Deep stereo fusion: combining multiple disparity hypotheses with deep-learning. In *3D Vision (3DV), 2016 Fourth International Conference on*, pages 138–147. IEEE, 2016.

[115] Matteo Poggi, Davide Pallotti, Fabio Tosi, and Stefano Mattoccia. Guided stereo matching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 979–988, 2019.

[116] François Pomerleau, Francis Colas, Roland Siegwart, and Stéphane Magnenat. Comparing icp variants on real-world data sets. *Autonomous Robots*, 34(3):133–148, 2013.

[117] Can Pu and Robert B Fisher. Udfnet: Unsupervised disparity fusion with adversarial networks. In *2019 26th IEEE International Conference on Image Processing (ICIP)*. IEEE, 2019.

[118] Can Pu, Nanbo Li, Radim Tylecek, and Bob Fisher. Dugma: Dynamic uncertainty-based gaussian mixture alignment. In *2018 International Conference on 3D Vision (3DV)*, pages 766–774. IEEE, 2018.

[119] Can Pu, Runzi Song, Radim Tylecek, Nanbo Li, and Robert B Fisher. Sdf-man: Semi-supervised disparity fusion with multi-scale adversarial networks. *Remote Sensing*, 11(5):487, 2019.

[120] Mihai Marian Puscas, Dan Xu, Andrea Pilzer, and Nicu Sebe. Structured coupled generative adversarial networks for unsupervised monocular depth estimation. *arXiv preprint arXiv:1908.05794*, 2019.

[121] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 652–660, 2017.

[122] Jiaxiong Qiu, Zhaopeng Cui, Yinda Zhang, Xingdi Zhang, Shuaicheng Liu, Bing Zeng, and Marc Pollefeys. Deeplidar: Deep surface normal guided depth prediction for outdoor scene from sparse lidar data and single color image. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3313–3322, 2019.

[123] Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y Ng. Ros: an open-source robot operating system. In *ICRA workshop on open source software*, volume 3, page 5. Kobe, Japan, 2009.

[124] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[125] Fabio Remondino and David Stoppa. *TOF range-imaging cameras*, volume 68121. Springer, 2013.

[126] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[127] Radu Bogdan Rusu, Nico Blodow, and Michael Beetz. Fast point feature histograms (fpfh) for 3d registration. In *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, pages 3212–3217. IEEE, 2009.

[128] Muhamad Risqi U Saputra, Andrew Markham, and Niki Trigoni. Visual slam and structure from motion in dynamic environments: A survey. *ACM Computing Surveys (CSUR)*, 51(2):37, 2018.

[129] Torsten Sattler, Radim Tylecek, Thomas Brox, Marc Pollefeys, and Robert B Fisher. 3d reconstruction meets semantics–reconstruction challenge 2017. In *ICCV Workshop, Venice, Italy, Tech. Rep*, 2017.

[130] Daniel Scharstein, Heiko Hirschmüller, York Kitajima, Greg Krathwohl, Nera Nešić, Xi Wang, and Porter Westling. High-resolution stereo datasets with subpixel-accurate ground truth. In *German conference on pattern recognition*, pages 31–42. Springer, 2014.

[131] Johannes L Schonberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4104–4113, 2016.

[132] Johannes L. Schönberger and Jan-Michael Frahm. Structure-From-Motion Revisited. In *CVPR*, 2016.

[133] Thomas Schöps, Johannes L. Schönberger, Silvano Galliani, Torsten Sattler, Konrad Schindler, Marc Pollefeys, and Andreas Geiger. A Multi-View Stereo Benchmark with High-Resolution Images and Multi-Camera Videos. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[134] Aleksandr Segal, Dirk Haehnel, and Sebastian Thrun. Generalized-icp. In *Robotics: science and systems*, volume 2, page 435, 2009.

[135] Akihito Seki and Marc Pollefeys. Patch based confidence prediction for dense disparity map. In *BMVC*, 2016.

[136] Shreyas S Shivakumar, Ty Nguyen, Steven W Chen, and Camillo J Taylor. Dfusenet: Deep fusion of rgb and sparse depth information for image guided dense depth completion. *arXiv preprint arXiv:1902.00761*, 2019.

[137] Merrill Ivan Skolnik. Introduction to radar systems. *New York, McGraw Hill Book Co., 1980. 590 p.*, 1980.

[138] Julian Straub, Trevor Campbell, Jonathan P. How, and John W. Fisher, III. Efficient global point cloud alignment using bayesian nonparametric mixtures. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[139] Federico Tombari, Samuele Salti, and Luigi Di Stefano. Unique signatures of histograms for local surface description. In *European conference on computer vision*, pages 356–369. Springer, 2010.

[140] Alessio Tonioni, Oscar Rahnama, Thomas Joy, Luigi Di Stefano, Thalaiyasingam Ajanthan, and Philip HS Torr. Learning to adapt for stereo. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9661–9670, 2019.

[141] Alessio Tonioni, Fabio Tosi, Matteo Poggi, Stefano Mattoccia, and Luigi Di Stefano. Real-time self-adaptive deep stereo. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 195–204, 2019.

[142] Yanghai Tsin and Takeo Kanade. A correlation-based approach to robust point set registration. In *European conference on computer vision*, pages 558–569. Springer, 2004.

[143] Benjamin Ummenhofer, Huizhong Zhou, Jonas Uhrig, Nikolaus Mayer, Eddy Ilg, Alexey Dosovitskiy, and Thomas Brox. Demon: Depth and motion network for learning monocular stereo. In *IEEE Conference on computer vision and pattern recognition (CVPR)*, volume 5, page 6, 2017.

[144] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[145] Jayakorn Vongkulbhisal, Beñat Irastorza Ugalde, Fernando De la Torre, and Joao P Costeira. Inverse composition discriminative optimization for point cloud registration. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2993–3001, 2018.

[146] Chen Wang, Danfei Xu, Yuke Zhu, Roberto Martín-Martín, Cewu Lu, Li Fei-Fei, and Silvio Savarese. Densefusion: 6d object pose estimation by iterative dense fusion. *arXiv preprint arXiv:1901.04780*, 2019.

[147] Rui Wang, Stephen M Pizer, and Jan-Michael Frahm. Recurrent neural network for (un-) supervised learning of monocular video visual odometry and depth. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5555–5564, 2019.

[148] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. High-resolution image synthesis and semantic manipulation with conditional gans. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8798–8807, 2018.

[149] Yue Wang and Justin M Solomon. Deep closest point: Learning representations for point cloud registration. *arXiv preprint arXiv:1905.03304*, 2019.

[150] Zhengwei Wang, Qi She, and Tomas E. Ward. Generative adversarial networks: A survey and taxonomy. *CoRR*, abs/1906.01529, 2019.

[151] Andreas Wedel, Annemarie Meißner, Clemens Rabe, Uwe Franke, and Daniel Cremers. Detection and segmentation of independently moving objects from dense scene flow. In *International Workshop on Energy Minimization Methods in Computer Vision and Pattern Recognition*, pages 14–27. Springer, 2009.

[152] Alex Wong and Stefano Soatto. Bilateral cyclic constraint and adaptive regularization for unsupervised monocular depth prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5644–5653, 2019.

[153] Oliver J Woodford, Minh-Tri Pham, Atsuto Maki, Frank Perbet, and Björn Stenger. Demisting the hough transform for 3d shape recognition and registration. *International Journal of Computer Vision*, 106(3):332–341, 2014.

[154] Stephen J Wright. *Primal-dual interior-point methods*. SIAM, 1997.

[155] Gengshan Yang, Joshua Manela, Michael Happold, and Deva Ramanan. Hierarchical deep stereo matching on high-resolution images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5515–5524, 2019.

[156] Jiaolong Yang, Hongdong Li, Dylan Campbell, and Yunde Jia. Go-ICP: a globally optimal solution to 3D ICP point-set registration. *IEEE transactions on pattern analysis and machine intelligence*, 38(11):2241–2254, 2016.

[157] Yanchao Yang, Alex Wong, and Stefano Soatto. Dense depth posterior (ddp) from single image and sparse range. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3353–3362, 2019.

[158] Zi Jian Yew and Gim Hee Lee. 3dfeat-net: Weakly supervised local 3d features for point cloud registration. In *European Conference on Computer Vision*, pages 630–646. Springer, 2018.

[159] Anestis Zaganidis, Li Sun, Tom Duckett, and Grzegorz Cielniak. Integrating deep semantic segmentation into 3-d point cloud registration. *IEEE Robotics and Automation Letters*, 3(4):2942–2949, 2018.

[160] Pietro Zanuttigh, Giulio Marin, Carlo Dal Mutto, Fabio Dominio, Ludovico Minto, and Guido Maria Cortelazzo. *Time-of-flight and structured light depth cameras*. Springer, 2016.

[161] Jure Zbontar and Yann LeCun. Stereo matching by training a convolutional neural network to compare image patches. *Journal of Machine Learning Research*, 17:1–32, 2016.

[162] Andy Zeng, Shuran Song, Matthias Nießner, Matthew Fisher, Jianxiong Xiao, and Thomas Funkhouser. 3dmatch: Learning local geometric descriptors from rgb-d reconstructions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1802–1811, 2017.

[163] Feihu Zhang, Victor Prisacariu, Ruigang Yang, and Philip HS Torr. Ga-net: Guided aggregation net for end-to-end stereo matching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 185–194, 2019.

[164] Zhenxin Zhang, Lan Sun, Ruofei Zhong, Dong Chen, Zhihua Xu, Cheng Wang, Cheng-Zhi Qin, Haili Sun, and Roujing Li. 3-d deep feature construction for mobile laser scanning point cloud registration. *IEEE Geoscience and Remote Sensing Letters*, 2019.

[165] Junbo Jake Zhao, Michaël Mathieu, and Yann LeCun. Energy-based generative adversarial networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, 2017.

[166] Huizhong Zhou, Benjamin Ummenhofer, and Thomas Brox. Deeptam: Deep tracking and mapping. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 822–838, 2018.

[167] Zhiyong Zhou, Jian Zheng, Yakang Dai, Zhe Zhou, and Shi Chen. Robust non-rigid point set registration using student's-t mixture model. *PloS one*, 9(3):e91381, 2014.

[168] Jihua Zhu, Congcong Jin, Zutao Jiang, Siyu Xu, Minmin Xu, and Shanmin Pang. Robust point cloud registration based on both hard and soft assignments. *Optics & Laser Technology*, 110:202–208, 2019.

[169] Jihua Zhu, Di Wang, Xiuxiu Bai, Huimin Lu, Congcong Jin, and Zhongyu Li. Registration of point clouds based on the ratio of bidirectional distances. In *3D Vision (3DV), 2016 Fourth International Conference on*, pages 102–107. IEEE, 2016.

[170] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2223–2232, 2017.