Vision-based Self-Supervised Depth Perception and Motion Control for Mobile Robots

by

Xiule Fan

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Applied Science
in
Mechanical and Mechatronics Engineering

Waterloo, Ontario, Canada, 2022

Author's Declaration

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

The advances in robotics have enabled many different opportunities to deploy a mobile robot in various settings. However, many current mobile robots are equipped with a sensor suite with multiple types of sensors. This expensive sensor suite and the computationally complex program to fully utilize these sensors may limit the large-scale deployment of these robots. The recent development of computer vision has enabled the possibility to complete various robotic tasks with simply camera systems. This thesis focuses on two problems related to vision-based mobile robots: depth perception and motion control.

Commercially available stereo cameras relying on traditional stereo matching algorithms are widely used in robotic applications to obtain depth information. Although their raw (predicted) disparity maps may contain incorrect estimates, they can still provide useful prior information towards more accurate predictions. We propose a data-driven pipeline to incorporate the raw disparity to predict high-quality disparity maps. The pipeline first utilizes a confidence generation component to identify raw disparity inaccuracies. Then a deep neural network, which consists of a feature extraction module, a confidence guided raw disparity fusion module, and a hierarchical occlusion-aware disparity refinement module, computes the final disparity estimates and their corresponding occlusion masks. The pipeline can be trained in a self-supervised manner, removing the need of expensive ground truth training labels. Experimental results on public datasets show that the pipeline has competitive accuracy with real-time processing rate. The pipeline is also tested with images captured by commercial stereo cameras to demonstrate its effectiveness in improving their raw disparity estimates.

After the stereo matching pipeline predicts the disparity maps, they are used by a proposed disparity-based direct visual servoing controller to compute the commanded velocity to move a mobile robot towards its target pose. Many previous visual servoing methods rely on complex and error-prone feature extraction and matching steps. The proposed visual servoing framework follows the direct visual servoing approach which does not require any extraction or matching process. Hence, its performance is not affected by the potential errors introduced by these steps. Furthermore, the predicted occlusion masks are also incorporated in the controller to address the occlusion problem inherited from a stereo camera setup. The performance of the proposed control strategy is verified by extensive simulations and experiments.

Acknowledgements

First of all, I would like to thank my supervisors, Professor Baris Fidan and Professor Soo Jeon. Their guidance and suggestions were extremely valuable to me as a beginner in academic research. More importantly, I am grateful that they were very supportive when I explored different research interests. I would also like to thank my thesis readers, Professor Yue Hu and Professor Fue-Sang Lien, for providing feedbacks to my work.

The next person I am grateful for is my partner, Jeniffer. She was my best supporter during my entire Master's study. She was always there to share my ups and downs in my everyday life. Without her, my time at Waterloo would not have been as enjoyable as it was.

Lastly, my greatest gratitude goes to my parents. They have always been an important and positive influence in my life. No words can compare to what they have taught me and how they have supported me.

Dedication

This thesis is dedicated to my loved ones.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation and Problem Definition

In domestic, commercial, and industrial settings, deploying intelligent mobile robots to complete various tasks (e.g., cleaning, social interaction, and material handling) has become more and more popular. In order to obtain accurate information from the environment and perform safe navigation, these robots are typically equipped with a sensor suite. The sensor suite may include some costly sensors, like a light detection and ranging (LiDAR) sensor. Additionally, utilizing multiple sensors to obtain high-quality information also requires careful calibration of these sensors. Furthermore, various sensor fusion techniques may be necessary to properly combine the captured information, which can lead to complex software design. All of the above disadvantages increase the difficulty of large-scale robot deployment.

Although the sensor suite with various types of sensors is traditionally needed for many robotic tasks, the recent development in computer vision has introduced new solutions to these tasks with the use of camera systems only. By reducing the robot's onboard sensors from a complete suite to just a few cameras, deploying an intelligent robotic system can become simpler and more cost efficient. In this thesis, we focus on the use of cameras in two specific robotic tasks: depth perception and motion control.

Previously, vision-based depth perception is often achieved by stereo cameras through various traditional stereo matching algorithms [41, 67]. Decades of research has improved the accuracy and real-time performance of these algorithms in many scenarios. However, their predictions, known as disparity, are still prone to erroneous or inaccurate estimates, especially in ambiguous (e.g. textureless, occluded, and highly reflective) regions.

Recently, using a deep neural network (DNN) trained with a large dataset of images has shown impressive results in vision-based depth perception. These data-driven depth perception methods are designed for either monocular or stereo cameras. As discussed later in Chapter 3, the stereo approaches are typically more reliable than the monocular ones. Among DNNs developed for stereo cameras, the ones trained in a supervised manner have outperformed traditional stereo matching algorithms and achieved state-of-the-art performance [51, 9, 103, 100, 14]. However, large datastes with ground truth labels are required to train these networks. These datasets are difficult and time consuming to collect. Hence, this shortcoming increases the difficulty of re-training a supervised approach to improve its accuracy when the input images are significantly different from its training images. To address this disadvantage, researchers have studied self-supervised stereo DNNs [109, 59, 104, 62, 95] that do not require any ground truth labels for training. However, these methods suffer from low accuracy and/or slow runtime, which limits their use in robotic applications. One of the focuses of this thesis is to develop a deep learning-based self-supervised stereo matching pipeline that can compute high-quality disparity estimates in real-time.

We further investigate vision-based motion control framework for mobile robots with the use of a stereo camera. Using image data to control a robot, known as visual servoing [46, 12], has been studied in the past decades. Visual servoing algorithms compute appropriate control signals by comparing a set of properties derived from image data to the reference ones. Traditionally, these algorithms [23, 38, 83, 11, 42, 18, 7] rely on various image processing approaches to extract image features and match them across different frames. These complex image processing algorithms may introduce errors, such as mismatched and unmatches features, into the controller. Therefore, the feature extraction and matching steps are often the limiting factors of the visual servoing algorithms [16, 87]. These limitations are addressed in some recent direct visual servoing approaches [16, 2, 3, 87] by utilizing features at constant pixel locations, which eliminates the need of feature extraction or matching. In these existing visual servoing algorithms, incorporating a stereo camera in the controller design is not a new concept [38, 7]. However, these stereo-based visual servoing approaches are still limited by the error-prone feature extraction and matching steps. Using a stereo camera in a direct visual servoing scheme is yet to be explored. Therefore, the other focus of this thesis is to design a direct visual servoing algorithm for a mobile robot that is equipped with a stereo camera by exploiting the estimation from the proposed stereo matching pipeline.

1.2 Contribution

To improve self-supervised stereo matching, we propose a data-driven occlusion-aware stereo matching pipeline with confidence guided raw disparity fusion. As pointed out previously, the disparity computed by a traditional stereo matching algorithm (raw disparity) may be erroneous at ambiguous regions. The estimates for other regions, however, are often accurate. A DNN can utilize these accurate raw disparity estimates as prior information to enhance its accuracy. In order to fully exploit this prior, our pipeline first computes a confidence map for a given raw disparity image. With the guidance of the computed confidence map, the raw disparity is used in a self-supervised DNN that predicts a more accurate disparity map and its corresponding occlusion mask. Since the DNN is based on a lightweight design, our stereo pipeline can achieve real-time processing rate. Experiments on various public and custom datasets demonstrate the effectiveness of the proposed pipeline.

We further design a direct visual servoing algorithm for a mobile robot based on the predicted disparity maps. The controller is derived according to the relationship between the stereo camera's velocity and the predicted disparity maps. In the proposed controller design, the predicted disparity maps are used to compute the control signals without the need of error-prone feature extraction or matching. Furthermore, the predicted occlusion maps are also incorporated into an occlusion-aware controller to address the occlusion problem originated from a stereo camera setup. The performance of the proposed controller is verified through extensive simulations and experiments.

1.3 Organization

The remaining of this thesis is organized in the following structure. In Chapter 2, background information and literature review related to depth estimation and visual servoing are given. The performance of various existing deep learning-based depth estimation algorithms is compared in Chapter 3. From this comparison, stereo approaches are identified as the more reliable solution. Based on this finding, Chapter 4 introduces and verifies the design of the proposed self-supervised stereo matching pipeline with confidence guided raw disparity fusion. After the discussions on the stereo pipeline, the design of the proposed direct disparity-based visual servoing framework is presented in Chapter 5 with verification through simulations and experiments. Lastly, the conclusion and recommended future work are given in Chapter 6.

Chapter 2

Background and Literature Review

2.1 Depth Estimation

2.1.1 Overview on Depth Estimation

In robotics, depth perception refers to the ability of a robot to detect its distance from other objects in the environment. Through decades of research and development, researchers and engineers have developed different approaches to capture depth information. Some of these methods rely on time-of-flight (ToF) sensors, while the others only require cameras.

Depth Estimation with Time-of-Flight Sensors

A ToF depth sensor consists of an emitter and receiver. The emitter emits light rays into the scene while the receiver captures the returning light rays. Either the time difference or phase shift between the emitted light and returning light is used to calculate the distance [1].

Common ToF depth sensors include Microsoft Kinect v2 sensors [24] and Velodyne's LiDAR systems [93]. There are some limitations associated with these ToF systems. For example, the receiver may capture multiple returning light rays from the same object, which leads to ambiguity in depth. Additionally, if the sensor or the objects in the scene move during a time period between a light ray's emission and return, the predicted depth may be inaccurate [26]. Furthermore, LiDAR systems can only compute sparse depth maps, and their high cost poses limitations on their use on robots [108].

Depth Estimation with Cameras

Computing depth with camera images is a popular alternative to ToF sensors since cameras are more cost-efficient and they can capture more semantic information [108]. Traditionally, stereo cameras are widely used for depth perception through a stereo matching algorithm. Recently, the advancements in deep learning has inspired researcher to revisit this problem with data-driven solutions.

A stereo camera is composed of two lenses to simulate human's binocular vision to infer depth. With a stereo camera, users can obtain two images of the same scene at the same time. If an object in the scene is visible in both images, the pixels associated to this object from both images form a corresponding pair. Given a pixel on one of the stereo images, finding its correspondence on the other view is the key to obtain stereo-based depth perception. Searching the correct correspondence is typically a computationally expensive process. To simplified this search, rectification is often performed on the stereo images so that the search is limited to a 1D scan line. This simplified search is referred as stereo matching. After the correct pixel correspondence is identified, the distance, known as disparity, between the pixels forming a corresponding pair can be found. Depth can then be computed from this disparity by

$$z = \frac{bf}{d}, \tag{2.1}$$

where z is the depth, b is the known camera baseline or distance between the two lenses, f is the camera's focal length in pixels, and d is the disparity in pixels [86]. Based on (2.1), estimating depth is equivalent to estimating disparity in a stereo setup since both baseline and focal length are usually available.

Although stereo cameras with a traditional stereo matching algorithm have been widely used in robotics, they still suffer from multiple problems that lead to incorrect, missing or noisy disparity estimates. Due to the physical setup of stereo cameras, some objects in the scene are only visible in one view but not the other, which is known as occlusion. The disparity estimates for these objects in the occluded regions are usually either incorrect or missing. Additionally, textureless regions in the images also impose difficulties on correct correspondence search [108]. Some active stereo cameras [52] attempt to solve the latter problem by adding textures to textureless regions by projecting artificial infrared light patterns to the scene. However, this method fails easily if the scene is flooded by strong natural illumination. The infrared lights from strong natural lighting can overwhelm the artificial patterns easily.

In addition to the traditional methods, the recent successes of deep learning in computer vision have opened the door for solving vision-based depth estimation with data-driven

approaches. Stereo matching algorithms based on DNNs have achieved state-of-the-art performance. The accuracy at occluded and textureless regions has also been improved [51]. Furthermore, DNNs even allow depth estimation from only monocular images captured by even cheaper monocular cameras.

2.1.2 Traditional Stereo Matching Algorithms

According to the analysis by Scharstein et al. [82], traditional stereo matching algorithms follow four general steps: matching cost computation, cost aggregation, disparity computation/optimization, and disparity refinement. At the matching cost computation step, each pixel or its neighborhood from one view of the stereo images is compared against all candidate correspondences on the other view. Then the matching cost is often aggregated within a support region in both the spatial domain and the disparity space to improve the robustness of stereo matching. A disparity map is regressed from the aggregated cost according to certain criteria. Lastly, the disparity map is optionally refined to achieve subpixel-level precision, detect occlusion, and/or remove noise.

Following these four general steps, traditional stereo matching algorithms can be categorized into local, global, and semi-global methods. Local methods [58, 105] usually adopt the winner-take-all strategy to select disparity from the matching cost. Global methods [56, 57] minimize a global energy function to achieve better accuracy at the cost of higher computational expenses. Semi-global matching (SGM) [41] combines the benefits of local and global methods by approximating global optimization in multiple local regions. Due to their accuracy and efficiency, SGM and its variants are widely used by commercial stereo cameras. For example, the Intel RealSense cameras [52] compute disparity using the AD-Census algorithm [67], which is a variant of SGM.

2.1.3 Deep Learning-based Depth Estimation

In recent years, deep learning has been widely used in various computer vision tasks, including vision-based depth estimation. These approaches often involve a carefully designed deep learning model (a DNN) and an optimization function (training loss). Prior to deploying (testing) these models in applications, they need to undergo a training process. In training, the model uses image data from a dataset to estimate the depth. Then the training loss is computed by the estimated depth and some supervisory signals. An optimization scheme, typically based on stochastic gradient descent [36], is used to update the

model parameters according to the training loss through back propagation. These data-driven depth estimation approaches can be categorized into monocular depth estimation and stereo depth estimation based on the types of input data. Alternatively, they also fall into either supervised or self-supervised category depending on whether ground truth depth labels are used as the supervisory signals or not.

Depth Estimation with Monocular Images

Deep learning-based monocular depth estimation was first explored in a supervised manner. The early design in [22] first predicts a coarse depth map with a monocular red-green-blue (RGB) image input. The coarse depth map is later refined with the RGB image as a guidance. Some researchers integrated DNNs with other algorithms like conditional random fields [60] and random forest [79] for better depth prediction. Jung et al. [50] used an adversarial network to predict more realistic depth maps. Wofk et al. [98] designed a lightweight model to enable monocular depth estimation on embedded systems.

One common problem in supervised learning is the difficulty in collecting the training dataset. Dense and accurate ground truth depth labels are required in this training scheme. These labels are typically time consuming to collect [29]. To remedy this challenge, self-supervised models are proposed.

There are two different approaches in self-supervised monocular depth estimation. The first one utilizes stereo images in training and only predicts depth with monocular images in testing. Garg et al. [29] predicted a depth map by using the left stereo image as an input. The predicted depth map is converted to disparity by reversing (2.1). Using the disparity map and the right stereo image, the algorithm constructs a synthetic left stereo image. The photometric difference between the actual and synthetic left images is the main supervisory signal to train this model. Godard et al. [34] followed the similar idea with an additional left-right consistency check to improve the quality of the predicted depth maps.

An alternative approach is to use monocular videos as training data and design an algorithm to mimic structure from motion. Zhou et al. [111] provided the first solution based on this approach. Their model consists of two components: a depth network and a pose network. The depth network computes a depth map from an RGB image captured at a particular time step. Using the same RGB image as a target and a temporally adjacent RGB image as a source, the pose network predicts the camera transformation between these two frames. By using the camera transformation and predicted depth, the source image can be projected to the same reference frame as the target image. The photometric difference between the original target image and the projected image is used

to train the model. Building upon this pioneering work, Casser et al. [8] explicitly modeled object motion to predict depth in a dynamic environment. Chen et al. [13] incorporated online optimization into their model. Godard et al. [35] modified the loss function to further improve the accuracy. Some other works derived from [111] attempt to tackle more difficult indoor scenarios with many ambiguous regions by either using optical flow [110] or superpixels [101] as additional information for training.

Although self-supervised approaches remove the reliance on ground truth datasets that are challenging to collect, they introduce other limitations. For examples, the approaches relying on stereo images for training require a stereo camera to collect the training data, which introduces cost in addition to the monocular camera. On the other hand, self-supervised models trained with monocular videos only require one monocular camera for data collection, but they suffer from scale ambiguity. This problem arises from the predicted camera transformation that is only unique up to scale. To remedy this limitation, previous works often scale the predicted depth maps such that their median depth is the same as the median of the ground truth depth maps [111]. From this perspective, ground truth supervision is still used in these methods.

Depth Estimation with Stereo Images

Similar to monocular depth estimation, deep learning-based stereo depth estimation was also first introduced as a supervised approach. The early attempt from [102] replaces the matching cost computation step in SGM with a convolutional neural network (CNN). Kendall et al. [51] designed an end-to-end DNN to mimic the four general steps for stereo matching from [82]. Their model first extracts high-level features from the input stereo images with multiple 2D convolutional layers. The extracted left features are concatenated with the right features to form a matching cost volume with the latter shifted according to the disparity candidates. The cost aggregation step is replaced by applying 3D convolutions to the cost volume. Lastly, disparity is regressed via a differentiable soft argmin operation. In this method, explicit reasoning is utilized to search the correct correspondence and disparity according to the left features and all right candidate features.

By following the framework outlined in [51], researchers have designed numerous networks to improve the accuracy in data-driven stereo matching. Chang and Chen [9] adopted spatial pyramid pooling [39, 106] to incorporate more contextual information into their model to improve accuracy at ambiguous regions. Zhang et al. [103] redesigned the cost aggregation module to reduce the number of expensive 3D convolutional layers in [9]. Khamis et al. [53] used a compact matching cost volume and a hierarchical refinement module guided by the input images to improve the real-time performance of their method

while maintaining high accuracy. Xu and Zhang [100] replaced all 3D convolutions with 2D deformable convolutions [17], which further increases accuracy and decreases runtime. Huang et al. [45] utilized deformable convolutions to perform fast occlusion-aware stereo matching. Cheng et al. [14] found the best network design in training through neural architecture search and reached state-of-the-art accuracy.

Similar to supervised monocular depth estimation, supervised stereo matching also suffers from the challenging process to collect training datasets with ground truth. Therefore, self-supervised training is also another popular research direction. To realize a self-supervised method, Zhou et al. [109] iteratively selected regions on stereo images with high confidence to complete stereo matching. Joung et al. [49] also relied on confidential matches and included additional smoothness constraints to handle non-matching pixels. Li and Yuan [59] first predicted occlusion masks for the image inputs and then performed occlusion-aware disparity prediction. Tonioni et al. [90] designed a modular network to allow online adaptation in a self-supervised manner. Wang et al. [95] incorporated a parallax attention mechanism into their network. While significant advancements have been made in self-supervised stereo matching, these methods still have low accuracy and/or low frame rate.

To further improve the model's performance, researchers have proposed some alternative approaches to include more information into either the model itself or the training process. Both Dovesi et al. [20] and Zhang et al. [104] jointly predicted semantic segmentation and disparity with their proposed models while the semantic cues help refine disparity. Liu et al. [62] used both stereo and temporally adjacent images to train their network. Additionally, there have been a few attempts to incorporate disparity estimates from a traditional stereo matching algorithm into a DNN. Ferrera et al. [25] fused disparity maps obtained by a traditional stereo matching algorithm into a light DNN to predict more accurate disparity in real-time. Instead of treating disparity from a traditional algorithm as an input to the DNN, Tonioni et al. [89] used a DNN to compute the confidence map of this raw disparity. The confidence map can filter its corresponding raw disparity map to obtain more reliable supervisory signal to train a stereo matching model. Wang et al. [94] designed a novel pyramid voting module to compute accurate and semi-dense disparity maps as the training labels for their DNN. All of these methods other than [25] do not attempt to incorporate raw disparity as the model's input. Although this approach has been investigated in [25], their model is designed as a supervised method. The potential of fusing raw disparity into a self-supervised stereo DNN is yet to be explored.

2.2 Stereo Matching Confidence

As mentioned previously, raw disparity computed by a traditional stereo matching algorithm may contain errors at ambiguous regions. In some applications, it is necessary to identify these inaccuracies by quantifying the confidence of the raw disparity map. Recently, Poggi et al. [76] provided a comprehensive review on different approaches to estimate stereo matching confidence. Some of the confidence measures require the matching cost properties [37, 43, 54], e.g., the matching cost for different disparity candidates. Other methods are based on the consistency between the disparity maps computed from the left stereo images and from the right stereo images [21, 43], respectively. Additionally, some methods focus on analyzing certain features of the predicted disparity maps themselves only [74, 84]. Furthermore, deep learning has also been applied to disparity confidence generation [28, 77, 91].

2.3 Visual Servoing

Visual servoing algorithms utilize image data to generate the control actions to move a robot equipped with a camera. Both [46] and [12] have provided comprehensive overviews of these approaches. The basic formulation of visual servoing algorithms [46] follows

$$\mathbf{v} = -\lambda \mathcal{L}_{\mathbf{s}}^{+} \left(\mathbf{s} - \mathbf{s}_{ref} \right), \tag{2.2}$$

where $\mathbf{v}$ denotes the velocity of the camera capturing the image, $\lambda > 0$ denotes the controller gain, $\mathcal{L}_{\mathbf{s}}^{+}$ is the pseudoinverse of an interaction matrix $\mathcal{L}_{\mathbf{s}}$, $\mathbf{s}$ is a vector of image properties derived from the image data, $\mathbf{s}_{ref}$ denotes these properties' desired values captured at the robot's target pose. Note that $\mathbf{v} \in \mathbb{R}^6$ if the camera is free to move in a 3D space with six degrees of freedom (DOF) and $\mathbf{v} \in \mathbb{R}^3$ if the camera motion is constrained on a 2D plane. The former scenario is common in applications with a robot manipulator, while the latter is typically seen on a mobile robot. The interaction matrix describes the relationship between the rate of change of the image property vector $\dot{\mathbf{s}}$ and the camera velocity as

$$\dot{\mathbf{s}} = \mathcal{L}_{\mathbf{s}} \mathbf{v}. \tag{2.3}$$

Depending on how the image-based properties are used by the controller, these algorithms can be categorized into two classes: position-based visual servoing (PBVS) and image-based visual servoing (IBVS).

2.3.1 Position-based Visual Servoing

In PBVS, a pose estimation algorithm uses images captured by a camera to compute estimates for pose control. Essentially, the image-based property vector $\mathbf{s}$ used in (2.2) is the estimated pose and $\mathbf{s}^*$ is the target pose in PBVS algorithms. By considering the image coordinates of several feature points on an object, Wilson et al. [97] utilized an extended Kalman filter (EKF) to predict the relative pose between the object and the camera mounted on a robot. The difference between the estimated relative pose and the reference relative pose is treated as a feedback signal to control the robot. PBVS is later extended in [88] to ensure the object needed for image-based pose estimation remains in the camera's field of view during the entire motion. The designed controller then moves the camera towards its target pose according to this criterion. Although PBVS algorithms can exploit image data to compute control actions for pose control, their performance is dependent on accurate camera calibration and high-quality pose estimates [7, 10].

2.3.2 Image-based Visual Servoing

To apply IBVS algorithms, a reference image is first captured at the robot's target pose. The difference between the image properties based on the same image features at the current frame and at the reference frame serves as an error signal for a controller to generate the appropriate control actions. A popular choice of such a property vector $\mathbf{s}$ in IBVS is the image coordinates of a set of image features. To identify the same image features at both the current and target frames, image processing techniques are adopted to extract these features and match them across different frames.

Traditional IBVS approaches have been designed according to different image features. In the early work [23], simple visual features (e.g., points, straight lines, and circles) are extracted for visual servoing. In [38], a stereo camera captures a pair of stereo views for the same corner feature. Based on this shared feature and the epipolar constraint, a control algorithm is proposed. Shademan and Janabi-Sharifi [83] adopted the Scale-Invariant Feature Transform (SIFT) [63] for robust feature selection and matching in their control framework. Chaumette [11] first performed segmentation on the images and computed the image moments for these segments. The moments then serve as the image features in visual servoing. The uses of image moments and SIFT are combined in [42]. The Speeded-Up Robust Features (SURF) [4] are also considered in visual servoing [18]. Cai et al. [7] used color as the image features in their visual servoing approach designed for an uncalibrated stereo camera. Although the above IBVS methods show that images can be used for motion control without the need of pose estimation, the complex image processing

algorithms to extract and match image features can introduce errors, such as mismatched and unmatched features. Hence, feature extraction and matching are often the bottleneck in IBVS [16, 87].

Recently, direct visual servoing has been studied to address the above limitation. In these approaches, the property based on image features is obtained according to all pixels in the image. Since all pixels are considered, feature extraction or matching is no longer necessary. Researchers have explored the use of color intensities [16], image histograms [3], photometric moments [2], wavelet coefficients [73], and images in the frequency domain [65] for direct visual servoing. Note that depth from the camera to the objects in the scene is typically required in these IBVS schemes. Therefore, Teuliere and Marchand [87] exploited the depth information generated by a red-green-blue-depth (RGB-D) camera directly in their visual servoing design. Their work has shown that depth maps from an RGB-D camera already contain useful information to achieve the control goal. Besides RGB-D cameras, stereo cameras are also widely used in robotics for depth perception. However, applying the stereo setup in direct visual servoing is yet to be studied.

Lastly, it is important to point out that the above visual servoing algorithms typically assume the camera is integrated with a robot manipulator capable of 6-DOF motion. The motion of the camera attached to a mobile robot, on the other hand, are constrained to a 2D plane with only 3 DOFs. Some researchers have extended different IBVS algorithms to mobile robots, such as visual servoing using SIFT features [27] and using depth maps [99, 61].

Chapter 3

Comparison of Existing Depth Estimation Approaches

As outlined in Chapter 2, various data-driven vision-based depth estimation algorithms have been proposed. This chapter provides a comparison between some selected models and summarizes their advantages and disadvantages. This summary serves as the foundation and reference for the proposed depth estimation pipeline.

Six different models with open-source implementation, including Monodepth2 [35], P^2Net [101], PSMNet [9], GA-Net [103], StereoNet [53], and AANet [100], are selected for this comparison. The first two approaches predict depth from monocular images. Monodepth2 provides a simple solution to achieve state-of-the-art performance, while P^2Net is designed for more accurate depth estimation in indoor environments. Although Monodepth2 and P^2N are trained in a self-supervised manner, their reported accuracy has surpassed many supervised methods. Among the selected stereo models, PSMNet adopts spatial pyramid pooling [39, 106] to increase accuracy of stereo matching at ambiguous regions. Through a sophisticated cost aggregation module, GA-Net estimates highly accurate disparity. Both StereoNet and AANet predict disparity maps with high accuracy and high frame rate.

The reported accuracies of these six models are based on different datasets and different error metrics. This inconsistency in evaluation makes comparing these methods directly according to their published results difficult. Hence, to facilitate better comparison, these models are evaluated by two custom datasets collected by an Intel D435 camera [52] with the use of each model's pretrained parameters provided by the model's authors/developers.

3.1 Datasets

Both datasets were collected indoors with the camera mounted on a Turtlebot2 [69]. Each instance of the datasets consists of a pair of stereo infrared (IR) images, a depth map computed by the onboard stereo matching algorithm, and an RGB image. All of these images were recorded to a laptop when the Turtlebot navigated in the indoor environments. Synchronization was performed offline to group the images into different instances according to their timestamps. Note that the depth image aligns with the left IR image, while the RGB image does not line up with either of the stereo views. The depth estimation models only use the left or both IR images to infer depth, depending on whether the method is a monocular or stereo algorithm. The RGB image just provides a better visualization of the scene.

In this chapter, these two datasets are referred to as the Flight Arena dataset and the Hallway dataset according to where they were recorded. The Flight Arena dataset includes 19 frames with items commonly found in an office/lab setting. There are 105 frames in the Hallway dataset with difficult scenarios, such as large textureless regions and specular reflection. Figure 3.1 shows some sample images from these two datasets.

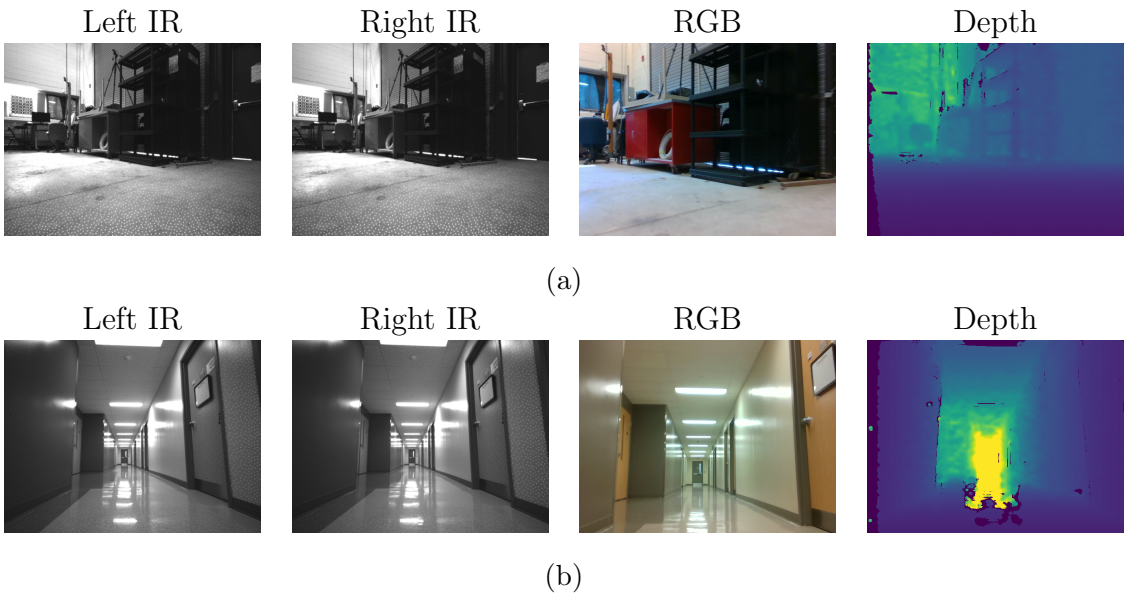


Figure 3.1: Sample images from the (a) Flight Arena dataset and (b) Hallway dataset.

3.2 Results

The results of each model are reported in the following error metrics: L1 error, root mean squared error (RMSE), and percentage of inliers. Denote a predicted depth map as $\mathbf{Z} \in \mathbb{R}_{>0}^{H \times W}$ and a ground truth depth map as $\mathbf{Z}^* \in \mathbb{R}_{>0}^{H \times W}$, where H and W are the height and width of these images, respectively. Due to the lack of ground truth depth images in either dataset, the depth maps computed by the camera are treated as the pseudo ground truth $\mathbf{Z}^*$ for evaluation. By considering an arbitrary pixel location $\mathbf{p}$ on the depth maps, the L1 error is given by

$$L1 = \frac{1}{HW} \sum_{\mathbf{p}} \left| \mathbf{Z}(\mathbf{p}) - \mathbf{Z}^*(\mathbf{p}) \right|, \tag{3.1}$$

The RMSE is

$$RMSE = \sqrt{\frac{\sum_{\mathbf{p}} \left(\mathbf{Z}(\mathbf{p}) - \mathbf{Z}^*(\mathbf{p}) \right)^2}{HW}}. \tag{3.2}$$

The percentage of inliers δ_ϵ with respect to a predefined threshold ϵ [22] is defined as

$$\delta_\epsilon = \frac{\sum_{\mathbf{p}} \mathbf{1}_\epsilon \left(\max \left(\frac{\mathbf{Z}(\mathbf{p})}{\mathbf{Z}^*(\mathbf{p})}, \frac{\mathbf{Z}^*(\mathbf{p})}{\mathbf{Z}(\mathbf{p})} \right) \right)}{HW}, \ \mathbf{1}_\epsilon(x) = \begin{cases} 1, \ x < \epsilon \\ 0, \ \text{otherwise} \end{cases}, \tag{3.3}$$

where $\max(\cdot)$ returns the larger value of the two input arguments.

In addition to accuracy evaluation, memory consumption and processing rate for these algorithms are also important considerations for their applications in robotics. This is because the hardware on a robot is often limited while the algorithms typically need to process information in real-time to ensure the robot's safe operation. In this comparison, the memory consumption is quantified by giga multiply-accumulate-operations (GMAC) and number of model parameters. In computing, 1 MAC is equivalent to performing one multiplication operation followed by one summation operation. GMAC quantifies how many operations are needed to execute the model once with one input instance. The processing rate of the model is measured by frame rate in frames per second (fps).

The results from accuracy evaluation on the Flight Arena dataset and the Hallway dataset are shown in Table 3.1a and Table 3.1b, respectively. Both tables indicate that the stereo methods can compute more accurate results than the monocular ones. During training, the monocular methods implicitly map features extracted by the DNNs with depth. If the features in the test image are significantly different from the ones from the training dataset, it may be difficult for these models to find the accurate depth. However,

stereo algorithms utilize explicit reasoning to search correspondences for even previously unseen features, resulting in more robust predictions.

	lower the better		higher the better		
Method	$L1$ (m)	$RMSE$ (m)	$\delta_{1.25}$	$\delta_{1.25^2}$	$\delta_{1.25^3}$
Monodepth2 [35]	0.5388	0.8748	0.5713	0.8285	0.9143
P^{2}Net [101]	0.6862	1.0344	0.4323	0.7194	0.8654
PSMNet [9]	0.3793	0.6749	0.8426	0.9464	0.9841
GA-Net [103]	0.1818	**0.3800**	**0.9622**	**0.9900**	0.9952
StereoNet [53]	**0.1780**	0.3925	0.9439	0.9895	**0.9971**
AANet [100]	0.1793	0.4076	0.9540	0.9871	0.9940

(a)

	lower the better		higher the better		
Method	$L1$ (m)	$RMSE$ (m)	$\delta_{1.25}$	$\delta_{1.25^2}$	$\delta_{1.25^3}$
Monodepth2 [35]	1.0201	1.5577	0.3297	0.6987	0.8687
P^{2}Net [101]	1.1119	1.5942	0.2991	0.5947	0.8005
PSMNet [9]	0.6733	1.2482	0.7275	0.9012	0.9721
GA-Net [103]	**0.2776**	**0.6739**	**0.9531**	**0.9899**	**0.9926**
StereoNet [53]	0.4490	0.8399	0.8220	0.9419	0.9799
AANet [100]	0.3433	0.8127	0.9173	0.9746	0.9901

(b)

Table 3.1: Accuracy evaluation on (a) the Flight Arena dataset and (b) the Hallway dataset.

By comparing the two monocular methods, it is surprising to find that P^{2}Net is less accurate than Monodepth2 even though P^{2}Net is specifically designed for indoor applications. This result may be explained by the camera setup used in this comparison. One advantage of P^{2}Net is to enforce the planar constraint at textureless regions that are commonly seen in indoor scenarios. The Intel D435 camera used to record the datasets projects random IR patterns into the scene. These IR patterns add textures to the textureless regions to facilitate better stereo matching by the camera. Due to the lack of textureless regions in

the recorded IR images, P^{2}Net may not demonstrate its full potential.

Among all stereo methods, GA-Net, StereoNet, and AANet are significantly more accurate than PSMNet. When inferring depth using the Flight Arena dataset, GA-Net, StereoNet, and AANet all have similar accuracy. However, GA-Net outperforms the other two methods considerably in the Hallway dataset as shown in Table 3.1b. This may be caused by the more frequent presence of specular reflection in the Hallway dataset than in the Flight Arena dataset as shown in Figure 3.1. This comparison suggests that GA-Net is the most accurate method among all four stereo approaches.

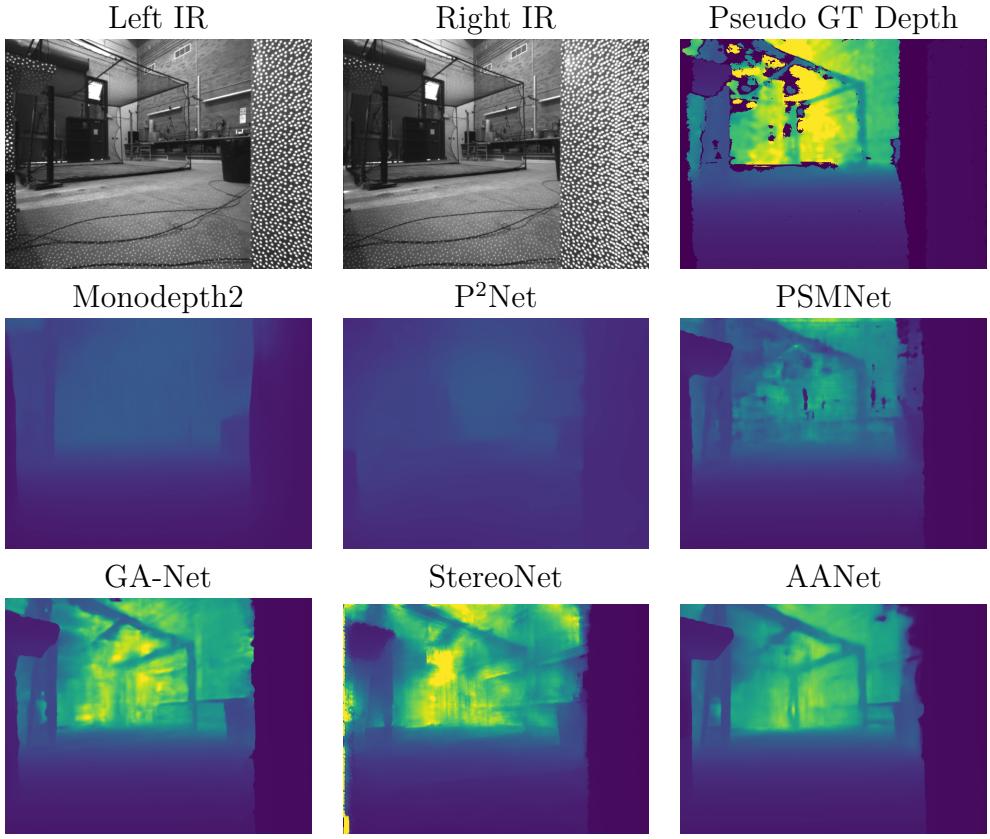


Figure 3.2: Sample IR stereo images from the Flight Arena Dataset, pseudo ground truth depth maps, and depth maps predicted by different algorithms.

Qualitative results for sample images from both datasets are shown in Figure 3.2 and Figure 3.3. Although Monodepth2 and P^{2}Net manage to preserve objects closed to the camera (e.g., the column in Figure 3.2 and the wall in Figure 3.3), they fail to predict

accurate distance for objects far away from the camera. All four stereo approaches capture the general structure of the scene but the performance among them also vary. PSMNet, GA-Net and AANet can preserve more details from the scene, which can be observed from the more defined frame structure shown in Figure 3.2 and smoother wall and floor transition in Figure 3.3. On the other hand, GA-Net and StereoNet can make better predictions for objects far way from the camera. This is supported by comparing the distribution of yellow regions in each of the predicted depth maps against the distribution in the pseudo ground depth map.

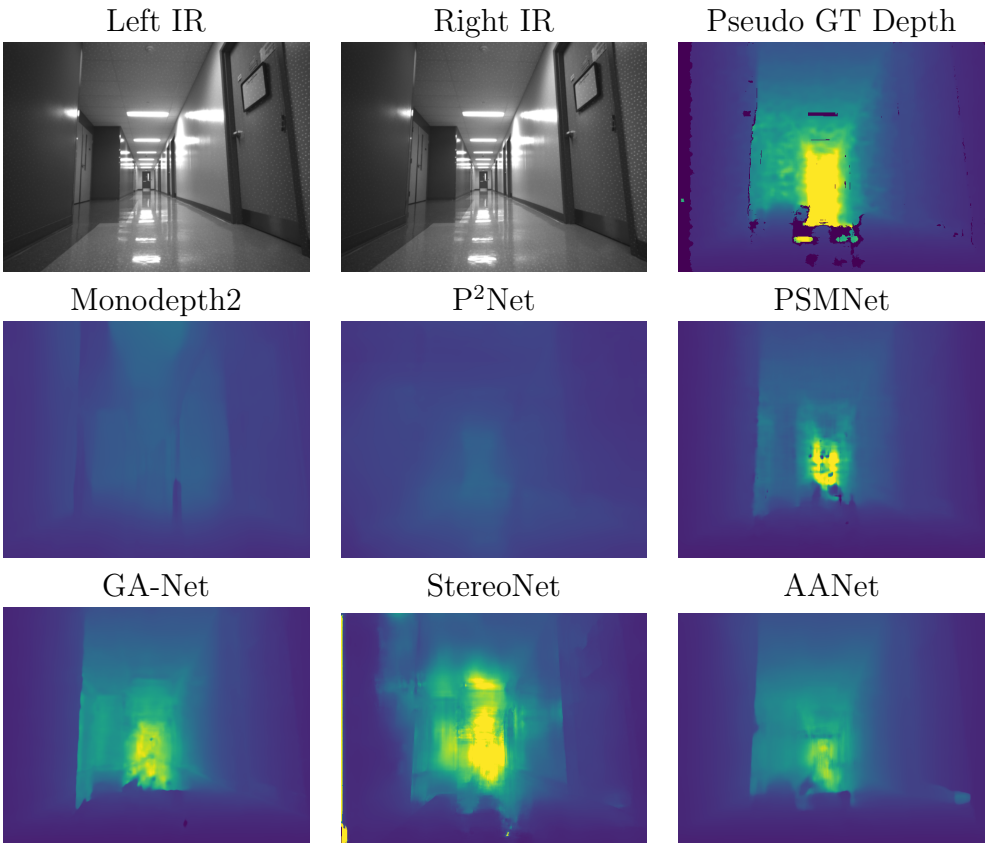


Figure 3.3: Sample IR stereo images from the Hallway Dataset, pseudo ground truth depth maps, and depth maps predicted by different algorithms.

In addition to accuracy comparison, Table 3.2 outlines the memory consumption and frame rate of each approach. The frame rate is measured on a desktop with an NVIDIA GTX 1660 Super GPU. The results show that the monocular methods can run significantly

faster than the stereo approaches with fewer operations to compute a depth map. However, the monocular models may consume more memory space than the stereo alternatives since the monocular models use more parameters. Among all stereo approaches, StereoNet consumes the least amount of memory with the highest frame rate.

	lower the better		higher the better
Method	GMAC	# of Param.	Frame rate (fps)
Monodepth2 [35]	5.18	14.84M	24.22
P^2Net [101]	**4.66**	14.84M	**25.01**
PSMNet [9]	601.54	5.22M	1.836
GA-Net [103]	1211.14	6.58M	0.1843
StereoNet [53]	32.83	**399.81k**	12.14
AANet [100]	118.21	3.44M	5.416

Table 3.2: Memory consumption and processing rate of each model.

All of the above observations from the comparison results can be summarized into the following points. The monocular methods can process images at a higher processing rate with the expense of more memory consumption. However, they fail to predict depth maps with reasonable accuracy. Therefore, monocular depth estimation is excluded for further considerations. On the other hand, the stereo methods are more superior than the monocular ones in terms of accuracy. Although, GA-Net is the most accurate model among all four stereo methods studied here, its frame rate is extremely low, which hinders its use in robotic applications. By taking accuracy, memory, and runtime into considerations, StereoNet offers a good balance between these factors. Therefore, it is used as a baseline for further development.

Chapter 4

Self-Supervised Stereo Vision with Raw Disparity Fusion

In the previous chapter, StereoNet [53] has been chosen as a baseline model. One limitation of StereoNet is its dependence on supervised training. Supervised training requires dense and accurate ground truth labels that are difficult and time consuming to obtain. When the model needs to be deployed in an unknown environment, its accuracy may degrade considerably. It is not always feasible to gather a dataset with ground truth information for re-training in a timely manner. Self-supervised training is a potential solution for this shortcoming. However, the strong supervisory signals provided by ground truth labels are not available in this training scheme. Therefore, the accuracy of self-supervised approaches is often lower than that of supervised methods.

To enable more accurate self-supervised stereo matching, an important task is to identify useful supervisory signals other than the ground truth labels. Previous studies [89, 25] have shown that raw disparity generated by a traditional stereo matching algorithm contains strong prior information for stereo DNN. Although these works utilize certain ground truth information, this finding may still be beneficial to self-supervised methods. Additionally, for a robot equipped with a commercially available stereo camera, the camera can often compute this prior disparity information on its onboard computer through a traditional stereo matching algorithm. Therefore, incorporating raw disparity in a self-supervised stereo matching approach may also be applicable to robotic applications.

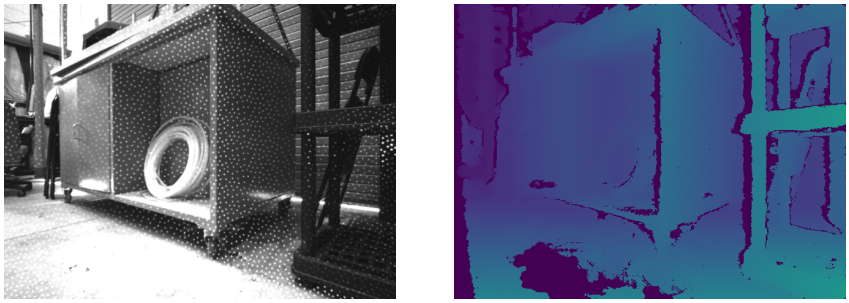
To fully explain and demonstrate the proposed design for self-supervised stereo matching, this chapter first presents discussions on raw disparity maps and how they can provide the proper prior information. Then the general architecture of the proposed approach is

presented. The details of each component in the pipeline are given after the introduction of the high-level overview. The performance of the pipeline is evaluated on publicly available datasets and custom datasets collected by commercial stereo cameras.

4.1 Observations on Raw Disparity Maps

Commercial stereo cameras use pre-designed algorithms to compute raw disparity/depth maps that are typically aligned with the left views. These algorithms are usually based on traditional stereo matching methods due to hardware limitations. For example, the Intel D435 [52] camera predicts depth from a variation of the AD-Census algorithm [52].

(a)



(b)

Figure 4.1: Sample left stereo views and their corresponding raw disparity maps computed by (a) the SGBM algorithm and (b) an Intel D435 camera.

Sample raw disparity computed by traditional stereo matching algorithms are given in Figure 4.1. Figure 4.1(a) shows a sample left RGB image from the Scene Flow dataset [66] and the left disparity map computed by the semi-global block matching (SGBM) [71] algorithm from the open-source OpenCV library [5]. Figure 4.1(b) shows the left IR image and its corresponding raw disparity map obtained from an Intel D435 camera. Note that

the camera's output is the left depth map. The raw disparity map is converted from the depth map by reversing (2.1).

Both samples show that the raw disparity maps from a traditional stereo matching algorithm or from a commercial stereo camera contain both correct and incorrect information. The raw disparity maps can capture the general disparity/depth of the scene. However, there are many incorrect or missing estimates in ambiguous (e.g., textureless or occluded) regions. At textureless regions (e.g., the gray cube in Figure 4.1(a) and the floor area in Figure 4.1(b)), the algorithms fail to find correct correspondence for many pixels, resulting in missing disparity estimates. Occluded areas present either at object boundaries or the leftmost areas of the left raw disparity maps. These regions are only visible in the left stereo view but not in the right view. Hence, the algorithms cannot make correct predictions for these pixels, which also causes either missing or incorrect disparity estimates.

Although the raw disparity maps struggle at textureless and occluded regions, their ability to capture the general structure of the scene can still provide useful information. If the erroneous or the missing disparity estimates are ignored or filtered out in Figure 4.1, the remaining estimates contain accurate information, which can serve as strong prior knowledge for a stereo DNN.

4.2 General Architecture of the Proposed Pipeline

To fully utilize the raw disparity information, a self-supervised stereo matching pipeline with confidence guided raw disparity fusion is proposed. It is assumed that a pair of rectified stereo images and the corresponding left raw disparity map are available as inputs to the pipeline. Note that the raw disparity map is restricted to the left view only because many stereo cameras only provide disparity information for the left view. In this proposed pipeline as shown in Figure 4.2, there are two main components: a confidence generation process and a DNN named CRD-Fusion.

In the pipeline, the confidence generation process first computes the confidence map given the raw disparity and stereo images. The confidence map provides an quantitative measures on the accuracy of the raw disparity. It is later used to remove inaccurate raw disparity estimates to obtain more reliable prior information. Next, the stereo images, raw disparity map and the confidence map are utilized by the self-supervised CRD-Fusion network. The DNN computes a more accurate disparity map with its corresponding occlusion mask.

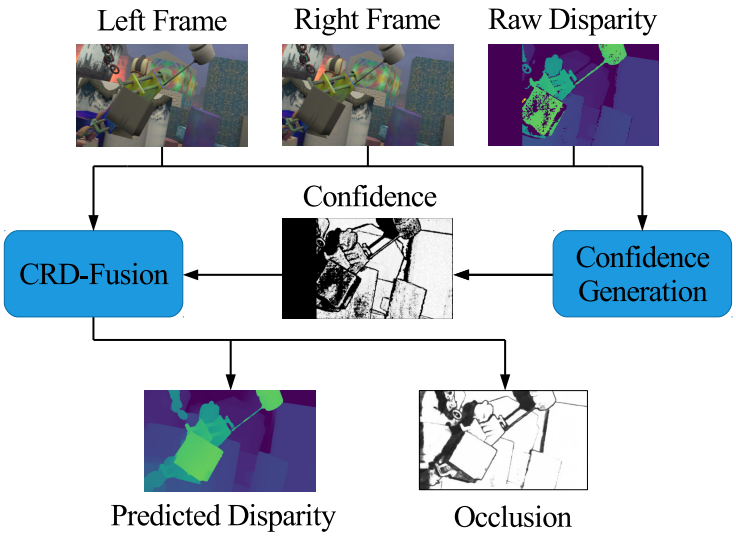


Figure 4.2: Illustration of the proposed self-supervised stereo matching pipeline with confidence guided raw disparity fusion.

4.3 Confidence Generation from Raw Disparity

Previous analysis on raw disparity maps has shown that they contain both accurate and inaccurate disparity information. Including inaccurate raw disparity into the CRD-Fusion network may cause erroneous final predictions. Therefore, the purpose of the confidence generation step is to quantify the raw disparity accuracy at each pixel location in the form of a confidence map.

Many of the confidence measures outlined in [76] have limitations which prevent their use in this pipeline. The methods relying on matching cost properties [37, 43, 54] require access to each pixel's matching cost at different disparity candidates. Typically, users can only retrieve the stereo images and the left raw disparity maps from a commercial stereo camera since the stereo matching algorithm adopted by the camera is a black box to the users. Intermediate values like matching cost are not accessible. Hence, these methods can not be used in our pipeline. Additionally, methods depending on both left and right disparity maps [21, 43] are also not applicable since the right disparity maps are not usually available to the users. Lastly, learning-based methods [28, 77, 91] are typically computationally expensive. Using them in the proposed pipeline may cause the entire pipeline to be too computationally intensive, especially given that the downstream stereo matching process is achieved by a DNN. Furthermore, re-training may be required if the

input data are significantly different from the training data.

Based on the above constraints and limitations, two confidence measures are chosen for the proposed use case: zero-mean sum of absolute difference (ZSAD) [37] and mean disparity deviation (MND) [74]. Neither of these methods require an expensive neural network. Moreover, computing ZSAD only needs the stereo images and the left raw disparity maps, while computing MND is only based on the raw disparity.

ZSAD is a confidence measure based on the difference between one stereo view and a reconstructed view obtained from the other stereo image and disparity. Assume the left rectified RGB image $\mathbf{I}^l$, right rectified RGB image $\mathbf{I}^r$, and left raw disparity map $\tilde{\mathbf{D}}$ are available to the user. Additionally, all of these images have a spatial resolution of $H \times W$. The ZSAD of a left-view pixel $\mathbf{p}^l = (j, i)$, where $j \in \{1, ..., H\}$ and $i \in \{1, ..., W\}$, is

$$\mathbf{ZSAD}\left(\mathbf{p}^l\right) = \sum_{\mathbf{q} \in N_z\left(\mathbf{p}^l\right)} \left| \mathbf{I}^l\left(\mathbf{q}\right) - \mu\left(\mathbf{I}^l\left(\mathbf{p}^l\right)\right) - \mathbf{I}^r\left(\mathbf{q}^r\right) + \mu\left(\mathbf{I}^r\left(\mathbf{p}^r\right)\right) \right|, \tag{4.1}$$

where $N_z(\cdot)$ denotes a window centered at the specified pixel with a preset size for ZSAD calculations, $\mu\left(\mathbf{I}^l\left(\cdot\right)\right)$ and $\mu\left(\mathbf{I}^r\left(\cdot\right)\right)$ denote the average intensities of $N_z(\cdot)$ in the left view and right view, respectively. The right-view pixel $\mathbf{p}^r$ corresponding to $\mathbf{p}^l$ is

$$\mathbf{p}^r = \mathbf{p}^l - \tilde{\mathbf{D}}\left(\mathbf{p}^l\right) = \left(j, i - \tilde{\mathbf{D}}\left(\mathbf{p}^l\right)\right). \tag{4.2}$$

Similarly, for a selected pixel $\mathbf{q}$ within the neighborhood of $N_z\left(\mathbf{p}^l\right)$ in the left view, its correspondence $\mathbf{q}^r$ from the right view is

$$\mathbf{q}^r = \mathbf{q} - \tilde{\mathbf{D}}\left(\mathbf{q}\right). \tag{4.3}$$

One problem arising from (4.2) and (4.3) is that the pixel indices from the left view ($\mathbf{p}^l$ and $\mathbf{q}$) are whole numbers while their corresponding raw disparities ($\tilde{\mathbf{D}}\left(\mathbf{p}^l\right)$ and $\tilde{\mathbf{D}}\left(\mathbf{q}\right)$) may have subpixel-level precision. Therefore, the indices of the resulting pixel correspondences ($\mathbf{p}^r$ and $\mathbf{q}^r$) are not whole numbers. In order to obtain the intensity of these pixel correspondences ($\mathbf{I}^r\left(\mathbf{p}^r\right)$ and $\mathbf{I}^r\left(\mathbf{q}^r\right)$) from the right image, bilinear interpolation [48] is used. Bilinear interpolation samples the pixels with whole-number indices adjacent to $\mathbf{p}^r$ and $\mathbf{q}^r$ and then performs weighted average to obtain the intensity at these pixel correspondences.

Since ZSAD in (4.1) is a similarity score according to pixel intensity, it may be affected by illumination and contrast of the stereo images. To minimize these effects, a simple normalization strategy is used to rescale the ZSAD score as

$$\mathbf{ZSAD}_n\left(\mathbf{p}^l\right) = \frac{\mathbf{ZSAD}\left(\mathbf{p}^l\right)}{\mu\left(\mathbf{ZSAD}\left(\mathbf{I}^l, \mathbf{I}^r, \tilde{\mathbf{D}}\right)\right)}, \tag{4.4}$$

where $\mu\left(\mathbf{ZSAD}\left(\mathbf{I}^l, \mathbf{I}^r, \tilde{\mathbf{D}}\right)\right)$ denotes the average of the entire ZSAD map obtained from $\mathbf{I}^l$, $\mathbf{I}^r$, and $\tilde{D}$.

Calculating confidence based on ZSAD only may lead to incorrect estimates, especially at textureless regions. Figure 4.3 shows an image patch cropped from the RGB image shown in Figure 4.1(a). The image patch is located at a textureless planar region with smooth ground truth disparity. When a traditional stereo matching algorithm attempts to find a match for the left-view pixels in this textureless region, many disparity candidates may have similar matching costs. This is because the color intensities of the right-view pixels corresponding to these disparity candidates may be visually similar to the left-view pixels. Therefore, the algorithm can easily select the incorrect candidates according to the ambiguous matching costs, which results in the noisy raw disparity shown in Figure 4.3. When calculating ZSAD for this textureless patch, the right image is sampled according to the raw disparity. Even with the presence of the inaccurate raw disparity esitmates, the reconstructed image patch may be almost visually identical to the original patch in the left view. ZSAD will therefore incorrectly assign high confidence level to this noisy raw disparity patch.

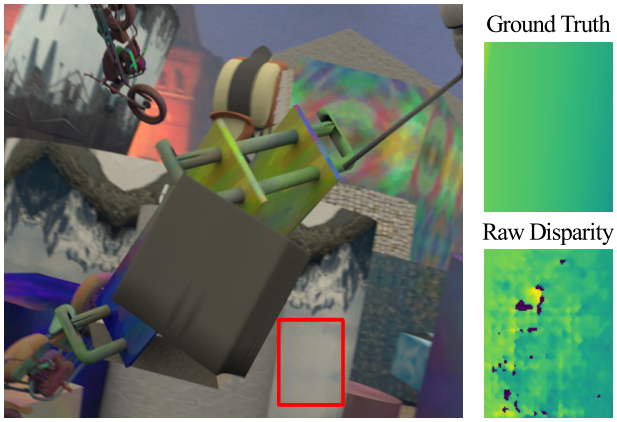


Figure 4.3: Image patch of a textureless region and its corresponding ground truth and raw disparity.

To properly calculate the confidence at textureless regions, we exploit the observation that raw disparity at these regions may be noisy. MND quantifies the smoothness of a disparity map and it is well suited in this use case. MND for pixel $\mathbf{p}^l$ is found by

$$\mathbf{MND}\left(\mathbf{p}^l\right) = -\left|\tilde{\mathbf{D}}\left(\mathbf{p}^l\right) - \mu\left(\tilde{\mathbf{D}}\left(\mathbf{p}^l\right)\right)\right|, \tag{4.5}$$

where $\mu\left(\tilde{\mathbf{D}}\left(\mathbf{p}^l\right)\right)$ is the mean raw disparity within a window $N_m\left(\mathbf{p}^l\right)$ centered at $\mathbf{p}^l$ with a preset size.

The normalized ZSAD in (4.4) and MND in (4.5) do not limit their values within a certain range. Typically, confidence score is constrained between 0 and 1. Additionally, MND is applicable to textureless regions only. Therefore, ZSAD and MND need to be combined in a texture-aware way to form a confidence map between 0 and 1. The proposed method to compute this confidence $\mathbf{Cf} \in \mathbb{R}^{H\times W}$ is

$$\mathbf{Cf}\left(\mathbf{p}^l\right) = \omega\left(\mathbf{p}^l\right) e^{-\gamma_m \mathbf{MND}\left(\mathbf{p}^l\right)} + \left(1-\omega\left(\mathbf{p}^l\right)\right) e^{-\gamma_z \mathbf{ZSAD}_n\left(\mathbf{p}^l\right)}, \tag{4.6}$$

where $\omega\left(\mathbf{p}^l\right)$ is a texture-aware guidance term. According to the definitions (4.1) and (4.5), $\mathbf{ZSAD}\left(\mathbf{p}^l\right) \geq 0$ and $\mathbf{MND}\left(\mathbf{p}^l\right) \leq 0$. Additionally, if their magnitudes are smaller, it implies that $\tilde{\mathbf{D}}\left(\mathbf{p}^l\right)$ is more accurate according to these measures. Hence, the constants that control the sensitivity of MND and ZSAD, respectively, are $\gamma_m < 0$ and $\gamma_z > 0$.

The texture-aware guidance term ensures that MND is considered as the more important measure for textureless regions while ZSAD is more applicable to areas with more textures. Since the gradient of image intensity is a strong indicator of textures within the image, the guidance term depends on this gradient as

$$\omega\left(\mathbf{p}^l\right) = e^{-\gamma_s \sqrt{\left(\frac{\partial}{\partial x}\mathbf{I}^l\left(\mathbf{p}^l\right)\right)^2 + \left(\frac{\partial}{\partial y}\mathbf{I}^l\left(\mathbf{p}^l\right)\right)^2}}, \tag{4.7}$$

where $\gamma_s > 0$ is a constant, $\frac{\partial}{\partial x}\mathbf{I}^l\left(\mathbf{p}^l\right)$ and $\frac{\partial}{\partial y}\mathbf{I}^l\left(\mathbf{p}^l\right)$ denote the image gradients with respect to the horizontal and vertical direction, respectively. In the actual implementation, these image gradients are approximated by Sobel filters.

Lastly, confidence scores for invalid pixels are replaced by zero. Two types of invalid pixels are considered in the confidence generation step. As shown in Figure 4.1, the traditional stereo matching algorithms cannot find disparity for some pixels. These pixels with missing raw disparity information are treated as invalid. Additionally, if the confidence scores for certain pixels are low, the raw disparity estimates for these pixels are likely incorrect. To prevent these pixels from affecting the downstream process, pixels with their confidence score lower than a threshold λ_C are also considered as invalid.

4.4 CRD-Fusion Network

To predict better disparity while exploiting the strong prior information provided by the raw disparity from a traditional stereo matching algorithm/stereo camera, the proposed

DNN needs to be effective and fast so that it can run in parallel with the camera for real-time robotic applications. The proposed CRD-Fusion network, as shown in Figure 4.4, is inspired by StereoNet [53] due to its good accuracy and high inference rate.

The network consists of three modules: feature extraction, confidence guided raw disparity fusion, and hierarchical occlusion-aware disparity refinement. The feature extraction module first extracts the high-level features from a pair of rectified stereo images. In the confidence guided raw disparity fusion module, the initial matching cost is constructed using the extracted features. After cost aggregation, the raw disparity and the confidence map are fused into the network to compute an initial disparity map at low resolution. The initial disparity map is gradually upsampled and refined by an occlusion-aware scheme using multiple refinement stages with the high-level features as guidance. At the end, the final predicted disparity map and its corresponding occlusion mask at full resolution are obtained. To train this network, a self-supervised training strategy is designed.

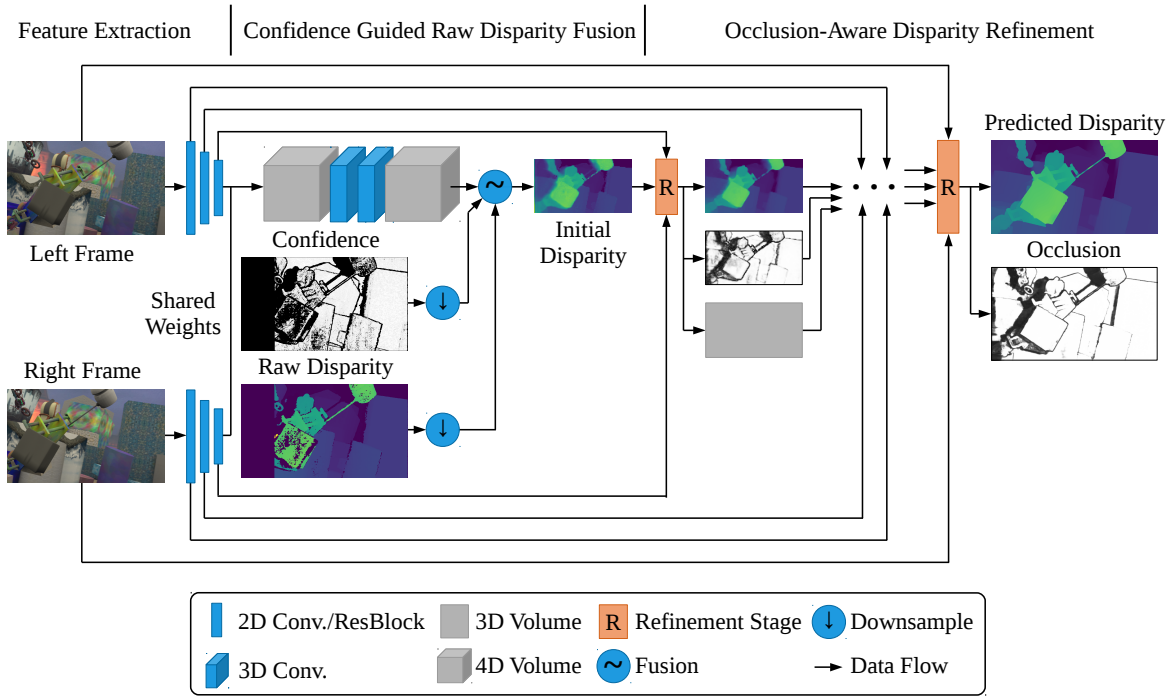


Figure 4.4: Overview of the CRD-Fusion network.

4.4.1 Feature Extraction

There are two purposes for the feature extraction module: downsampling and extraction of high-level features. Downsampling an input RGB image can lower the computational load required to predict the initial disparity map. Furthermore, color intensity in an RGB image may not be descriptive enough to obtain accurate stereo matching. On the other hand, high-level features, which contain more contextual information, can lead to more robust performance [51].

In the original StereoNet design, the feature extraction module first downsamples the input image by using three 2D convolutional layers with large receptive fields. After downsampling, the image tensor is at 1/8 of its original resolution. Then multiple residual blocks [40] are applied to the small image tensor to further extract high-level features. This approach can capture more contextual information at the lowest image resolution, but not necessarily at higher resolution. This design is suitable for StereoNet since it produces initial disparity maps at 1/8 resolution and does not require high-level features at the intermediate scales (i.e., 1/2 and 1/4 resolutions) in the model.

In the CRD-Fusion network, the feature extraction module is modified so that high-level features at different image resolutions can be used in the disparity refinement modules. The module consists of multiple stages with the same design. By using different number of stages, high-level features at different resolutions can be extracted.

Given an input image, the proposed feature extraction module needs to extract features $\{\mathbf{F}_0, \mathbf{F}_1, ..., \mathbf{F}_K\}$, where $\mathbf{F}_k \in \mathbb{R}^{H/2^k \times W/2^k \times C_k}$ for $k \in \{0, 1, ..., K\}$, and C_k is the number of feature channels at scale k. Note that feature $\mathbf{F}_0$ at the original resolution $k = 0$ is essentially the input image. Additionally, $C_0 = 3$ is used to account for the color channels in an RGB image.

A feature extraction stage k is responsible for processing feature $\mathbf{F}_k$ to compute the feature at the next scale $\mathbf{F}_{k+1}$. Each stage follows the design outlined in Table 4.1. First, a 5×5 2D convolutional layer with stride 2 reduces the spatial resolution of $\mathbf{F}_k$ by half. Then a residual block with 3×3 convolutions and an additional 3×3 convolutional layer extract more features to obtain $\mathbf{F}_{k+1}$. All of the convolutional layers except for the last one in each stage are followed by batch normalization [47] and the leaky ReLU [64] activation function. The number of feature channels C_k is fixed at 32 for all $k \neq 0$. There are $K - 1$ extraction stages in total to obtain K features.

The same feature extraction module is applied to both left and right stereo images. This can help reduce the memory footprint of the model and ensure the features extracted from both views are consistent. A convolution operation applies the same kernel/filter to

Layer	Description	Input Size	Output Size
1	5×5 conv., stride 2	$H/2^k \times W/2^k \times C_k$	$H/2^{k+1} \times W/2^{k+1} \times C_{k+1}$
2	3×3 residual block	$H/2^{k+1} \times W/2^{k+1} \times C_{k+1}$	$H/2^{k+1} \times W/2^{k+1} \times C_{k+1}$
3	3×3 conv.	$H/2^{k+1} \times W/2^{k+1} \times C_{k+1}$	$H/2^{k+1} \times W/2^{k+1} \times C_{k+1}$

Table 4.1: Structure of the feature extraction stage k. All convolutions (conv.) except for layer 3 are followed by batch normalization and leaky ReLU. $C_0 = 3$ and $C_k = 32$ if $k \neq 0$.

its inputs at all pixel locations. No matter where an object is located in an image, the same features are extracted from this object. In a stereo setting, the same non-occluded objects are visible in both left and right views. Using the same feature extraction module on both images will lead to two feature maps with the same non-occluded high-level features but at different locations.

After applying the feature extraction module to both input images $\mathbf{I}^l$ and $\mathbf{I}^r$, two sets of features $\left\{\mathbf{F}_0^l, \mathbf{F}_1^l, ..., \mathbf{F}_K^l\right\}$ and $\{\mathbf{F}_0^r, \mathbf{F}_1^r, ..., \mathbf{F}_K^r\}$ are computed. These features are used in the following modules to predict disparity.

4.4.2 Confidence Guided Raw Disparity Fusion

In a typical stereo-based deep learning model, a matching cost given a pre-defined maximum disparity range is generated by using the extracted features. To reduce the computational cost, only features at low resolution are used. Additionally, if the maximum number of disparity candidates at the original image resolution is D_0, the corresponding number of disparity candidates at an arbitrary scale k is

$$D_k = \frac{D_0}{2^k}. \tag{4.8}$$

Equation (4.8) shows that using a low image resolution reduces the disparity range, which can further lower the computational cost. The matching cost generation step in CRD-Fusion also follows this procedure to construct the cost at the lowest resolution, which corresponds to the resolution scale K.

Given the high-level features $\mathbf{F}_K^l$ and $\mathbf{F}_K^r$, the matching cost $\mathbf{c}$ is formed according to D_K disparity candidates. The matching cost $\mathbf{c}$ is a 4D tensor with dimension of $H/2^K \times W/2^K \times D_K \times C_K$. For a pixel $\mathbf{p}_K^l = (j_K, i_K)$ where $j_K \in \left\{1, ..., H/2^K\right\}$ and $i_K \in \left\{1, ..., W/2^K\right\}$ from the left feature $\mathbf{F}_K^l$, its matching cost $\mathbf{c}\left(\mathbf{p}_K^l, d_k\right) \in \mathbb{R}^{C_K}$ given a disparity candidate d_K is defined by

$$\mathbf{c}\left(\mathbf{p}_K^l, d_k\right) = \mathbf{F}_K^l\left(\mathbf{p}_K^l\right) - \mathbf{F}_K^r\left(\mathbf{p}_K^l - d_K\right), \, d_K \in \{0, 1, ..., D_K - 1\}, \tag{4.9}$$

where the pixel correspondence is similar to (4.2) as

$$\mathbf{p}_K^l - d_K = (j_K, i_K - d_K)\,. \tag{4.10}$$

In the case when $i_K < d_K$, the resulting pixel correspondence $\mathbf{p}_K^l - d_K$ is outside of $\mathbf{F}_K^r$, which causes $\mathbf{F}_K^r\left(\mathbf{p}_K^l - d_K\right)$ in (4.9) to become invalid. To address this problem, the cost $\mathbf{c}\left(\mathbf{p}_K^l, d_k\right)$ when $i_K < d_K$ is explicitly set to zero. Note that the matching cost $\mathbf{c}$ is constructed by fixing $\mathbf{F}_K^l$ and shifting $\mathbf{F}_K^r$. Hence, $\mathbf{c}$ is the difference between reference left-view pixels and all candidate right-view pixels. The disparity map inferred from this matching cost is therefore the left disparity map.

As pointed out in [51], it is important to aggregate the cost volume in three dimensions: height, width and candidate disparities. This can help the model learn more contextual information and minimize multi-modal distribution in the cost volume. In the context of deep learning, this aggregation step is achieved by applying 3D convolutional layers to the cost volume. By following [53], a lightweight cost aggregation design is adopted.

The aggregation follows the layers shown in Table 4.2 sequentially. The first four 3D convolutaional layers followed by batch normalization and the leaky ReLU activation function maintain the same number of feature channels at C_K. The last convolution which does not utilize any normalization or activation function combines all of the feature channels. Then the dimension of the feature channels is removed. The resulting aggregated cost volume is therefore $\hat{\mathbf{c}} \in \mathbb{R}^{H/2^K \times W/2^K \times D_K}$.

Layer	Description	Input Size	Output Size
1	$3 \times 3 \times 3$ conv.	$H/2^K \times W/2^K \times D_K \times C_K$	$H/2^K \times W/2^K \times D_K \times C_K$
2	$3 \times 3 \times 3$ conv.	$H/2^K \times W/2^K \times D_K \times C_K$	$H/2^K \times W/2^K \times D_K \times C_K$
3	$3 \times 3 \times 3$ conv.	$H/2^K \times W/2^K \times D_K \times C_K$	$H/2^K \times W/2^K \times D_K \times C_K$
4	$3 \times 3 \times 3$ conv.	$H/2^K \times W/2^K \times D_K \times C_K$	$H/2^K \times W/2^K \times D_K \times C_K$
5	$3 \times 3 \times 3$ conv.	$H/2^K \times W/2^K \times D_K \times C_K$	$H/2^K \times W/2^K \times D_K \times 1$
6	Dim. reduction	$H/2^K \times W/2^K \times D_K \times 1$	$H/2^K \times W/2^K \times D_K$

Table 4.2: Layers used in cost aggregation, including five 3D convolutional layers (conv.) and a dimension (dim.) reduction operation. Layers 1 to 4 are followed by batch normalization and leaky ReLU activation.

After cost aggregation, the aggregated cost volume can be used to regress a disparity map at resolution scale K. The soft argmin operation proposed in[51] is designed for this task. The operation begins with a conversion from the aggregated matching cost $\hat{\mathbf{c}}$ to a

probability distribution $\hat{\mathbf{P}} \in \mathbb{R}^{H/2^K \times W/2^K \times D_K}$ for all disparity candidates at scale K. This conversion is given by

$$\hat{\mathbf{P}}\left(\mathbf{p}_K^l, d_K\right) = \sigma\left(-\hat{\mathbf{c}}\left(\mathbf{p}_K^l, d_K\right)\right), \tag{4.11}$$

where $\sigma(\cdot)$ is the softmax [6] operator which yields an exponentially normalized probability. The softmax operator in this case is defined as

$$\sigma\left(-\hat{\mathbf{c}}\left(\mathbf{p}_K^l, d_K\right)\right) = \frac{e^{-\hat{\mathbf{c}}\left(\mathbf{p}_K^l, d_K\right)}}{\sum_{d_i=0}^{D_K-1} e^{-\hat{\mathbf{c}}\left(\mathbf{p}_K^l, d_i\right)}}. \tag{4.12}$$

After the probability distribution is obtained, an $H/2^K \times W/2^K$ preliminary disparity map $\hat{\mathbf{D}}$ is regressed by the following weighted summation:

$$\hat{\mathbf{D}}\left(\mathbf{p}_K^l\right) = \sum_{d_K=0}^{D_K-1} d_K \hat{\mathbf{P}}\left(\mathbf{p}_K^l, d_K\right). \tag{4.13}$$

The preliminary disparity map is only dependent on the deep features extracted in the previous module. The prior knowledge from raw disparity, especially the more accurate raw disparity estimates as identified by the confidence map, is not exploited yet. To incorporate this prior information, a confidence guided raw disparity fusion step is proposed to improve the preliminary disparity map. The preliminary disparity $\hat{\mathbf{D}}$ is regressed at the resolution scale K while $\tilde{\mathbf{D}}$ and $\mathbf{Cf}$ are computed at the original resolution. To ensure their resolutions are consistent, $\tilde{\mathbf{D}}$ and $\mathbf{Cf}$ are resized to resolution scale K by nearest downsampling to obtain the low-resolution raw disparity $\tilde{\mathbf{D}}_K$ and low-resolution confidence $\mathbf{Cf}_K$. Note that after downsampling, $\tilde{\mathbf{D}}_K$ must be divided by 2^K to properly scale the disparity values. Once the resolutions are consistent, the fusion step is carried out by

$$\mathbf{D}_i\left(\mathbf{p}_K^l\right) = \mathbf{Cf}_K\left(\mathbf{p}_K^l\right)\tilde{\mathbf{D}}_K\left(\mathbf{p}_K^l\right) + \left(1 - \mathbf{Cf}_K\left(\mathbf{p}_K^l\right)\right)\hat{\mathbf{D}}\left(\mathbf{p}_K^l\right) \tag{4.14}$$

to compute the $H/2^K \times W/2^K$ initial disparity map $\mathbf{D}_i$ that combines the information from both the deep features and raw disparity. The fusion scheme (4.14) exploits the accurate raw disparity when possible. When the raw disparity is not available or its corresponding confidence map has low scores, (4.14) focuses more on the preliminary disparity based on the deep features.

4.4.3 Hierarchical Occlusion-Aware Disparity Refinement

The initial disparity map $\mathbf{D}_i$ is at low resolution and does not contain many details. To obtain a more accurate disparity map at higher resolution, the initial disparity map needs

to be upsampled and refined [53]. StereoNet adopts a hierarchical refinement module which upsamples the initial disparity map gradually to the original image resolution by doubling the resolution at each step. Additionally, it uses the left image as guidance to recover detailed disparity in the prediction.

Although the refinement module in [53] can successfully estimate accurate disparity, it lacks the ability to address occlusion. This missing function is not problematic in [53] since it is designed as a supervised model. To train a stereo model in a supervised manner, the ground truth disparity values are provided. These values also include the correct disparity at occluded regions. By training with these ground truth labels, the model implicitly learns how to make accurate predictions at occluded regions. Therefore, StereoNet only needs a simple refinement module to obtain high-quality results. On the other hand, the proposed pipeline is designed as a self-supervised approach with no knowledge of ground truth information in training. To achieve more accurate results, it is necessary to address occlusion explicitly.

Recently, MaskFlowNet [107] has been proposed to solve the optical flow prediction problem in a self-supervised manner. The model adopts a hierarchical design to upsample low-resolution predicted optical flow maps gradually and then refine the upsampled maps. Similar to stereo matching, occlusion is also present in optical flow prediction. To mitigate this problem, the model predicts a soft occlusion mask at each stage of refinement. The occlusion mask is considered in the next refinement stage. A similar design has also been extended for supervised stereo matching [45].

Inspired by [53] and [107], the CRD-Fusion network utilizes a hierarchical occlusion-aware refinement module. Similar to the feature extraction module, the refinement module also consists of multiple stages. By changing the number of refinement stages, the model can be applied to different image resolutions.

The proposed refinement module follows the arrangement shown in Table 4.3. There are K refinement stages in the refinement module. Each stage is responsible for doubling the resolution of the input disparity map while providing additional detailed disparity estimates through a refinement block. With K stages, the module can refine the initial disparity map $\mathbf{D}_i$, which is regressed at the resolution scale K, and obtain the refined disparity at the original image resolution.

The general structure of a disparity refinement stage k can be visualized in Figure 4.5 with additional description in Table 4.3. The inputs of the stage include refined disparity $\mathbf{D}_{k+1}$, predicted occlusion mask $\mathbf{O}_{k+1}$, and disparity residual feature $\mathbf{R}_{k+1}$ at scale $k+1$ from the previous refinement stage $k+1$, as well as high-level features $\mathbf{F}_k^l$ and $\mathbf{F}_k^r$ from the feature extraction module. The stage computes the refined disparity $\mathbf{D}_k$, occlusion mask

Stage	Input	Description	Output
K	$\mathbf{F}_K^l$, $\mathbf{F}_K^r$, $\mathbf{D}_i$	Correlation	$\mathbf{Corr}_K$
	$\mathbf{Corr}_K$, $\mathbf{F}_K^l$, $\mathbf{D}_i$	Refinement block K	$\mathbf{R}_K$
	$\mathbf{R}_K$	3×3 conv, added to $\mathbf{D}_i$	$\mathbf{D}_K$
	$\mathbf{R}_K$	3×3 conv	$\mathbf{O}_K$
$K-1$	$\mathbf{D}_K$, $\mathbf{O}_K$	Upsample	$\mathbf{D}_{K-1}^{'}$, $\mathbf{O}_{K-1}^{'}$
	$\mathbf{R}_K$	Upsample, 3×3 conv	$\mathbf{R}_{K-1}^{''}$
	$\mathbf{R}_{K-1}^{''}$	3×3 conv	$\mathbf{R}_{K-1}^{'}$
	$\mathbf{F}_{K-1}^l$, $\mathbf{F}_{K-1}^r$, $\mathbf{D}_{K-1}^{'}$, $\mathbf{O}_{K-1}^{'}$, $\mathbf{R}_{K-1}^{'}$	Correlation	$\mathbf{Corr}_{K-1}$
	$\mathbf{Corr}_{K-1}$, $\mathbf{F}_{K-1}^l$, $\mathbf{R}_{K-1}^{''}$, $\mathbf{D}_2^{'}$	Refinement block $K-1$	$\mathbf{R}_2$
	$\mathbf{R}_{K-1}$	3×3 conv, added to $\mathbf{D}_{K-1}^{'}$	$\mathbf{D}_{K-1}$
	$\mathbf{R}_{K-1}$	3×3 conv	$\mathbf{O}_{K-1}$
		.....	
0	$\mathbf{D}_1$, $\mathbf{O}_1$	Upsample	$\mathbf{D}_0^{'}$, $\mathbf{O}_0^{'}$
	$\mathbf{R}_1$	Upsample, 3×3 conv	$\mathbf{R}_0^{''}$
	$\mathbf{R}_0^{''}$	3×3 conv	$\mathbf{R}_0^{'}$
	$\mathbf{F}_0^l$, $\mathbf{F}_0^r$, $\mathbf{D}_0^{'}$, $\mathbf{O}_0^{'}$, $\mathbf{R}_0^{'}$	Correlation	$\mathbf{Corr}_0$
	$\mathbf{Corr}_0$, $\mathbf{F}_0^l$, $\mathbf{R}_0^{''}$, $\mathbf{D}_0^{'}$	Refinement block 0	$\mathbf{R}_0$
	$\mathbf{R}_0$	3×3 conv, added to $\mathbf{D}_0^{'}$	$\mathbf{D}_0$
	$\mathbf{R}_0$	3×3 conv	$\mathbf{O}_0$

Table 4.3: Description of the hierarchical occlusion-aware disparity refinement module.

$\mathbf{O}_k$, and disparity residual feature $\mathbf{R}_k$ at scale k.

The first step in the refinement stage is to ensure the inputs are at the same resolution. Among all of the inputs, $\mathbf{D}_{k+1}$, $\mathbf{O}_{k+1}$, and $\mathbf{R}_{k+1}$ are at scale $k+1$, while $\mathbf{F}_k^l$ and $\mathbf{F}_k^r$ are at scale k. Hence, the first three inputs need to be upsampled. $\mathbf{D}_{k+1}$ and $\mathbf{O}_{k+1}$ are resized by simply applying a bilinear upsampling operator $U(\cdot)$ to them as

$$\mathbf{D}_k^{'} = 2U\left(\mathbf{D}_{k+1}\right), \tag{4.15}$$

$$\mathbf{O}_k^{'} = U\left(\mathbf{O}_{k+1}\right). \tag{4.16}$$

After resizing, the upsampled disparity $\mathbf{D}_k^{'}$ and occlusion mask $\mathbf{O}_k^{'}$ are at the resolution scale k. Note that a multiplier of 2 applied to $\mathbf{D}_{k+1}$ is needed to ensure the disparity values are correct after upsampling. Upsampling of the disparity residual feature $\mathbf{R}_{k+1}$ is more complicated. As described later, $\mathbf{R}_{k+1}$ is of dimension $H/2^{k+1} \times W/2^{k+1} \times C_{k+1}^{\mathbf{R}}$, where the number of channels $C_{k+1}^{\mathbf{R}}$ is large. To reduce the computational load, $\mathbf{R}_{k+1}$ is first

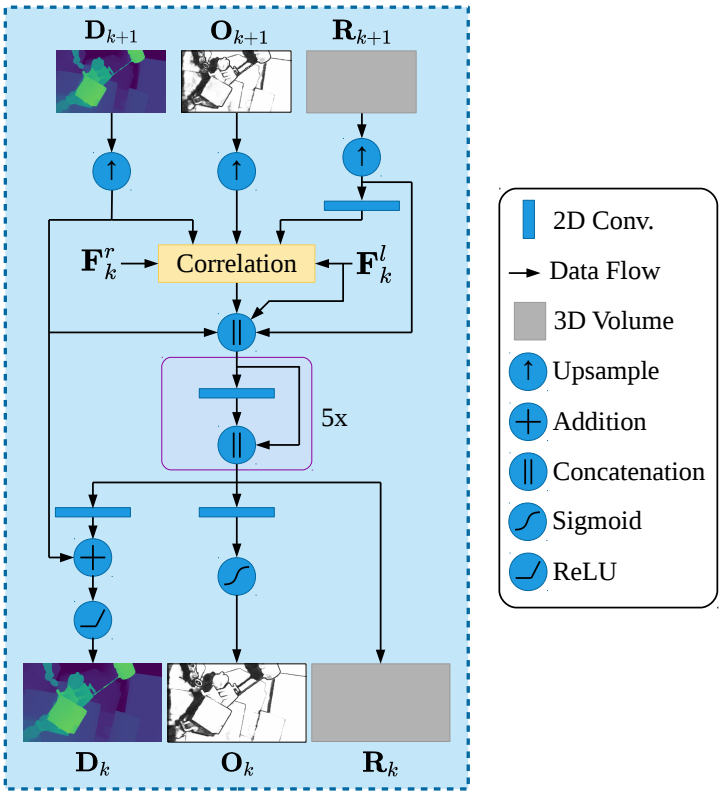


Figure 4.5: Overview of refinement stage k.

bilinearly upsampled and then processed by a 3×3 2D convolution for channel reduction to obtain residual feature $\mathbf{R}_k^{''} \in \mathbb{R}^{H/2^k \times W/2^k \times 16}$. The 2D convolution here is followed by batch normalization and leaky ReLU activation. Furthermore, a second 3×3 2D convolution without batch normalization or activation function computes another disparity residual feature $\mathbf{R}_k^{'} \in \mathbb{R}^{H/2^k \times W/2^k \times C_k}$ from $\mathbf{R}_k^{''}$. $\mathbf{R}_k^{'}$ shares the same number of channels C_k as the high-level features $\mathbf{F}_k^l$ and $\mathbf{F}_k^r$.

Following input resizing, the next step evaluates the accuracy of the upsampled disparity $\mathbf{D}_k^{'}$ in the form of a correlation score [19]. Correlation quantifies the similarity of its inputs, which are high-level features in the context of the refinement module. Prior to constructing the correlation score, the right high-level feature $\mathbf{F}_k^r$ is first bilinearly sampled to build a synthetic left high-level feature $\hat{\mathbf{F}}_k^l$ according to $\mathbf{D}_k^{'}$. Since only non-occluded regions are visible in both views, feature reconstruction is not applicable to occluded pixels. Therefore,

the occlusion-aware sampling scheme follows

$$\hat{\mathbf{F}}_k^l = S\left(\mathbf{F}_k^r, \mathbf{D}_k^{'}\right) \otimes \mathbf{O}_k^{'} \oplus \mathbf{R}_k^{'}, \tag{4.17}$$

where $S(\cdot)$ is the bilinear sampler, $\otimes$ and $\oplus$ are element-wise tensor multiplication and addition, respectively. The multiplication with the upsampled occlusion mask $\mathbf{O}_k^{'}$ enforces sampling at non-occluded regions only. The addition of the disparity residual feature $\mathbf{R}_k^{'}$ results in better occlusion prediction in the later steps according to [107]. For a pixel $\mathbf{p}_k^l = (j_k, i_k)$, where $j_k \in \left\{1, ..., H/2^k\right\}$ and $i_k \in \left\{1, ..., W/2^k\right\}$, the correlation measures a patch-wise similarity between $\mathbf{F}_k^l$ and $\hat{\mathbf{F}}_k^l$ to evaluate the accuracy of $\mathbf{D}_k^{'}\left(\mathbf{p}_k^l\right)$ as

$$\mathbf{Corr}_k^{'}\left(\mathbf{p}_k^l\right) = \sum_{o \in (-m,m)\times(-m,m)} \left\langle \mathbf{F}_k^l\left(\mathbf{p}_k^l + o\right), \hat{\mathbf{F}}_k^l\left(\mathbf{p}_k^l + o\right) \right\rangle. \tag{4.18}$$

The image patches of interest are centered at $\mathbf{p}_k^l$ in $\mathbf{F}_k^l$ and $\hat{\mathbf{F}}_k^l$ with the size of $(2m+1) \times (2m+1)$.

One disadvantage of the correlation calculated according to (4.17) and (4.18) is that it only explores the similarity of the two input feature volumes according to the predicted disparity. Without sufficient refinement, the predicted disparity map may contain inaccurate values, which may lead to low similarity. In spite of the presence of inaccuracies, the predicted disparity can still offer a heuristic about the approximate value of the accurate estimate. If a range of disparity values according to the predicted disparity are considered in the correlation step, it is more likely for the correlation score to reflect information of the accurate disparity. By following this observation and [19], the sampling rule and correlation are modified to

$$\hat{\mathbf{F}}_{k,d'}^l = S\left(\mathbf{F}_k^r, \mathbf{D}_k^{'} + d'\right) \otimes \mathbf{O}_k^{'} \oplus \mathbf{R}_k^{'}, \tag{4.19}$$

$$\mathbf{Corr}_{k,d'}\left(\mathbf{p}_k^l, d'\right) = \sum_{o \in (-m,m)\times(-m,m)} \left\langle \mathbf{F}_k^l\left(\mathbf{p}_k^l + o\right), \hat{\mathbf{F}}_{k,d'}^l\left(\mathbf{p}_k^l + o\right) \right\rangle, \tag{4.20}$$

where $d' \in [-d_{range}, d_{range}]$ is a disparity offset from a pre-defined range. After the correlation for each disparity offset is computed, the correlation scores $\mathbf{Corr}_{k,d'} \in \mathbb{R}^{H/2^k \times W/2^k}$ for all offsets are concatenated together to form a correlation score volume $\mathbf{Corr}_k \in \mathbb{R}^{H/2^k \times W/2^k \times (2d_{range}+1)}$. In the actual implementation, the image patch and disparity range are fixed to $m = 1$ and $d_{range} = 4$. Hence, the correlation score is applied to 3×3 image patches and the final correlation volume is $\mathbf{Corr}_k \in \mathbb{R}^{H/2^k \times W/2^k \times 9}$.

After $\mathbf{Corr}_k$ is obtained, a refinement block as shown in Table 4.4 is used to exploit more contextual information to compute a new disparity residual feature $\mathbf{R}_k$. Inside the refinement block, the correlation score $\mathbf{Corr}_k \in \mathbb{R}^{H/2^k \times W/2^k \times 9}$, left high-level feature $\mathbf{F}_k^l \in \mathbb{R}^{H/2^k \times W/2^k \times C_k}$, upsampled disparity residual feature $\mathbf{R}_k^{''} \in \mathbb{R}^{H/2^k \times W/2^k \times 16}$, and the upsampled disparity $\mathbf{D}_k^{'} \in \mathbf{R}^{H/2^k \times W/2^k \times 1}$ are concatenated together. The number of channels Ch_k for the volume after concatenation depends on the number of channels of the inputs. For example, if all four inputs are used, $Ch_k = 9 + C_k + 16 + 1 = 26 + C_k$. The concatenated volume undergoes a series of five combinations of 2D convolution and concatenation. In each combination, a 3×3 2D convolution followed by batch normalization and leaky ReLU activation processes the input volume. Then the output from the convolution is concatenated with the input volume along the channel dimension. The number of output channels is 32 for the first three convolutions and 16 for the remaining two. Similar to [53], some of the convolutional layers are dilated convolutions as shown in Table 4.4 in order to increase the receptive field. After all five convolutions and concatenations, the disparity residual feature $\mathbf{R}_k$ is obtained and it contains $C_k^{\mathbf{R}} = Ch_k + 3 \times 32 + 2 \times 16 = 128 + Ch_k$ channels. As pointed out earlier, the number of channels $C_k^{\mathbf{R}}$ is high for $\mathbf{R}_k$. Reducing it in the next stage of refinement can reduce the computational load.

Layer	Input	Description	Output Size	Output
1	$\mathbf{Corr}_k$, $\mathbf{F}_k^l$, $\mathbf{R}_k^{''}$, $\mathbf{D}_k^{'}$	Concatenation	$H/2^k \times W/2^k \times Ch_k$	concat1
2	concat1	3×3 conv	$H/2^k \times W/2^k \times 32$	conv2
3	concat1, conv2	Concatenation	$H/2^k \times W/2^k \times (32 + Ch_k)$	concat3
4	concat3	3×3 conv, $d = 2$	$H/2^k \times W/2^k \times 32$	conv4
5	concat3, conv4	Concatenation	$H/2^k \times W/2^k \times (64 + Ch_k)$	concat5
6	concat5	3×3 conv, $d = 4$	$H/2^k \times W/2^k \times 32$	conv6
7	concat5, conv6	Concatenation	$H/2^k \times W/2^k \times (96 + Ch_k)$	concat7
8	concat7	3×3 conv	$H/2^k \times W/2^k \times 16$	conv8
9	concat7, conv8	Concatenation	$H/2^k \times W/2^k \times (112 + Ch_k)$	concat9
10	concat9	3×3 conv	$H/2^k \times W/2^k \times 16$	conv10
11	concat9, conv10	Concatenation	$H/2^k \times W/2^k \times (128 + Ch_k)$	$\mathbf{R}_k$

Table 4.4: Description of the refinement block k (d - dilation). The number of channels Ch_k depends on the inputs.

The disparity residual feature $\mathbf{R}_k$ from the refinement block is used to estimate both a disparity residual $\Delta\mathbf{D}_k$ and a predicted occlusion mask at the current scale k. One

3×3 convolution is applied to $\mathbf{R}_k$ to compress the number of channels to calculate $\Delta\mathbf{D}_k \in \mathbb{R}^{H/2^k \times W/2^k \times 1}$, and a separate 3×3 convolution also processes $\mathbf{R}_k$ to find the occlusion mask with the size of $\mathbb{R}^{H/2^k \times W/2^k \times 1}$. The residual disparity and the upsampled disparity are added together to obtain the refined disparity at scale k as

$$\mathbf{D}_k = \text{ReLU}\left(\mathbf{D}_k^{'} + \Delta\mathbf{D}_k\right), \tag{4.21}$$

where the ReLU operation ensures $\mathbf{D}_k$ is non-negative. Similarly, the occlusion mask from the convolutional layer undergoes a sigmoid operator so that the predicted occlusion mask $\mathbf{O}_k$ is constrained between 0 and 1. Lastly, the refined disparity $\mathbf{D}_k$, occlusion mask $\mathbf{O}_k$, and disparity residual feature $\mathbf{R}_k$ act as the inputs for the next refinement stage at scale $k-1$.

In the actual implementation, the refinement module starts from refinement stage $k = K$ and ends at stage $k = 0$. At stage K, the upsampling steps are ignored with $\mathbf{D}_K^{'}$ initialized as $\mathbf{D}_i$, $\mathbf{O}_K^{'}$ initialized as a tensor of 1's, $\mathbf{R}_K^{'}$ initialized as a tensor of 0's, and $\mathbf{R}_K^{''}$ initialized as an empty tensor. Because of this initialization scheme and $C_K = 32$, the number of channels $Ch_K = 9 + 32 + 1 = 42$ and $C_K^{\mathbf{R}} = 128 + 42 = 170$ for this scale. For $k \in \{1, 2, ..., K-1\}$, all four inputs are used in the corresponding refinement block and $C_k = 32$. Hence, the number of channel is $Ch_k = 26 + 32 = 58$ and $C_k^{\mathbf{R}} = 128 + 58 = 186$. When $k = 0$, the feature $\mathbf{F}_0^l$ is essentially the input left image $\mathbf{I}^l$. Therefore, $C_0 = 3$ leads to $Ch_0 = 26 + 3 = 29$ and $C_0^{\mathbf{R}} = 128 + 29 = 157$. The refined disparity $\mathbf{D}_0$ and occlusion mask $\mathbf{O}_0$ at this scale are at the original image resolution. They are used as the final outputs of the pipeline.

4.4.4 Loss Function

Training a neural network involves calculating a loss function, computing the gradient of the loss function with respect to the network parameters, and updating the parameters based on the gradient. The self-supervised training loss to train the CRD-Fusion network is

$$L = \frac{1}{(K+1)HW} \sum_{k=0}^{K} \left(\frac{1}{2^k} \left(\alpha_d L_{d,k} + \alpha_p L_{p,k} + \alpha_s L_{s,k} + \alpha_o L_{o,k} \right) \right), \tag{4.22}$$

where L_d is the disparity supervision loss according to the raw disparity, L_p is the photometric loss, L_s is the smoothness loss for the predicted disparity map, L_o is the occlusion loss, and α's are the weights for each term. The subscript k denotes that the loss is computed with $\mathbf{D}_k$ and/or $\mathbf{O}_k$ from refinement stage k. Including the scaling factor $1/2^k$ help

emphasize the losses at higher resolution. Similar to [35], all refined disaprity maps $\mathbf{D}_k$ and predicted occlusion masks $\mathbf{O}_k$ are bilinearly upsampled to the original resolution as $\mathbf{D}_k^0$ and $\mathbf{O}_k^0$ for loss computation. Note that no ground truth disparity maps are used in this loss function, which makes it suitable for self-supervision. The definition of each loss term is given below.

Raw Disparity Supervision Loss

The raw disparity map contains rich information of the actual disparity values. Hence, it is a strong supervisory signal to train the network. However, the raw disparity map may also include inaccurate disparity estimates. This error must be removed to provide appropriate supervision. The proposed raw disparity supervision loss incorporates the confidence map to filter out the inaccurate areas in loss computation. This loss follows

$$L_{d,k} = \sum_{\mathbf{p}^l} \mathbf{Cf}\left(\mathbf{p}^l\right) \Delta_s \left(\mathbf{D}_k^0\left(\mathbf{p}^l\right) - \tilde{D}\left(\mathbf{p}^l\right)\right), \tag{4.23}$$

where $\Delta_s\left(x\right)$ is the smooth L1 loss [33] given by

$$\Delta_s\left(x\right) = \begin{cases} \frac{1}{2}x^2, \; x < 1 \\ |x| - \frac{1}{2}, \text{ otherwise} \end{cases} \tag{4.24}$$

Photometric Loss

The raw disparity supervision loss can only focus on regions with accurate raw disparity. At areas with inaccurate or missing raw disparity information, additional supervision is necessary to improve their disparity estimates. Another commonly used loss in depth estimation is the photometric loss. The photometric loss compares the difference between a reference image and a source image warped to the reference image's frame. In the context of stereo matching, the warping process involves reconstructing one stereo view from another under the condition of disparity. This warping process can be achieved by the bilinear sampler $S(\cdot)$. Since only non-occluded pixels are visible in both views, occluded pixels should be removed from the photometric loss computation. Additionally, the raw disparity supervision loss already provides strong supervision at pixels with accurate raw disparity estimates. The photometric loss should therefore allocate more emphasis on pixels without useful raw disparity information. Based on these considerations, the photometric loss is

inspired by [34, 59, 95] as

$$L_{p,k} = \sum_{\mathbf{p}^l} \left(\frac{\alpha}{2} \left(1 - SSIM \left(\mathbf{I}^l \left(\mathbf{p}^l \right), \hat{\mathbf{I}}_k^l \left(\mathbf{p}^l \right) \right) \right) + (1-\alpha) \left\| \mathbf{I}^l \left(\mathbf{p}^l \right) - \hat{\mathbf{I}}_k^l \left(\mathbf{p}^l \right) \right\| \right) \mathbf{O}_k^0 \left(\mathbf{p}^l \right) \left(1 - \mathbf{Cf} \left(\mathbf{p}^l \right) \right), \tag{4.25}$$

where the constant α is set to 0.85, $SSIM(\cdot)$ is the structural similarity index measure [96] to quantify the similarity between two images, and $\hat{\mathbf{I}}_k^l$ is the reconstructed left view by

$$\hat{\mathbf{I}}_k^l = S \left(\mathbf{I}^r, \mathbf{D}_k^0 \right). \tag{4.26}$$

Disparity Smoothness Loss

Minimizing only the raw disparity supervision loss and photometric loss may cause the predicted disparity map to be noisy. The raw disparity map can only provide sparse supervision to the model, while the photometric loss may result in incorrect matches at ambiguous regions. Additionally, estimation at occluded areas is not addressed in these two losses. Including the disparity smoothness loss can reduce the noisiness of the predicted disparity maps. Moreover, it can fill in disparity at occluded regions with disparity from non-occluded pixels. However, disparity discontinuities do exist in many scenarios due to the presence of multiple objects in the scene. Boundaries of these objects provide a good indication of where disparity discontinuities are located, and these boundaries often reveal themselves as change in color intensity in the image. Therefore, the smoothness loss follows the edge-aware design in [35, 34] as

$$L_{s,k} = \sum_{\mathbf{p}^l} \left| \frac{\partial}{\partial x} \mathbf{D}_k^0 \left(\mathbf{p}^l \right) \right| e^{-\left\| \frac{\partial}{\partial x} \mathbf{I}^l \left(\mathbf{p}^l \right) \right\|} + \left| \frac{\partial}{\partial y} \mathbf{D}_k^0 \left(\mathbf{p}^l \right) \right| e^{-\left\| \frac{\partial}{\partial y} \mathbf{I}^l \left(\mathbf{p}^l \right) \right\|}. \tag{4.27}$$

Occlusion Loss

Since the predicted occlusion mask is part of the photometric loss computation, minimizing $L_{p,k}$ may cause the occlusion mask to become all 0's. When the predicted occlusion mask is zero, the resulting photometric loss is also zero even though inaccurate predicted disparity $\mathbf{D}_k^0$ will still lead to nonidentical $\mathbf{I}^l$ and $\hat{\mathbf{I}}_k^l$. Therefore, a zero occlusion mask may bring inaccurate information to the training process. A behavior similar to this has been observed in [111] where the model predicts an explainability mask that can converge to zero easily.

To remedy this problem, a binary cross entropy loss between the explainability mask and a constant mask with 1's is used. The binary cross entropy loss pushes the explainability mask towards the non-zero direction. This same approach is adopted here and the occlusion loss is given by

$$L_{o,k} = \sum_{\mathbf{p}^l} -\ln \mathbf{O}_k^0 \left(\mathbf{p}^l\right). \tag{4.28}$$

4.5 Experiments

4.5.1 Datasets

The proposed pipeline is evaluated on multiple datasets, including three public datasets and two custom datasets, to demonstrate its effectiveness. The public datasets are SceneFlow [66], KITTI 2015 [68], and KITTI 2012 [32]. These datasets are widely used by other works on stereo matching. Evaluating the proposed pipeline on these datasets is more convenient for comparison with other existing approaches. The two custom datasets were collected by commercially available stereo cameras that are commonly seen in robotic applications. Since the pipeline is designed for use with a commercial stereo camera, evaluating it on these custom datasets can verify its performance with respect to this goal.

The SceneFlow dataset is a synthetic dataset with 35,454 frames in the training set and 4,370 frames in the test set. The ground truth disparity maps are provided in both sets. The image size for all images is 540 $\times$ 960. The dataset contains very challenging scenes with textureless regions, large disparity, and large occlusion.

Both KITTI 2015 and KITTI 2012 are real-word datasets collected in driving scenarios. The stereo images were captured by two cameras mounted on a car. KITTI 2015 includes 200 training frames and 200 test frames, while KITTI 2012 provides 194 training frames and 195 test frames. Ground truth disparity is provided for training images in the form of sparse disparity maps obtained from a LiDAR. To evaluate the model's performance on test images, the predicted results must be submitted to the official KITTI server. The image size varies in these two datasets with an average image size of 375 $\times$ 1240.

The two custom datasets were collected by an Intel RealSense D435 camera [52] and a Zed Mini camera [85] mounted on a Turtlebot 2 [69], respectively, as shown in Figure 4.6. In the dataset collected by the RealSense camera, there are 5,546 training frames and 1,387 testing frames. The resolution of all frames is 480 $\times$ 640. Note that the stereo images from the RealSense camera are infrared images with only one color channel. Since the pipeline

requires the input stereo images to have three color channels, the same infrared images are concatenated three times along the channel dimension. Additionally, the camera only provides depth maps instead of disparity. The raw disparity maps are computed based on the depth maps by reversing (2.1) with known camera baseline and focal length. In addition to the RealSense dataset, the dataset recorded by the Zed camera includes 2,967 training frames and 742 testing frames. The image resolution of this dataset is 720×1280. No ground truth data is available in these custom datasets. Therefore, the test frames are only used for qualitative evaluation.

Figure 4.6: System setup for custom dataset collection.

4.5.2 Implementation Details

The raw disparity maps for public datasets are computed by the SGBM algorithm [71] from the OpenCV library due to its simplicity and ease of use, while the ones for custom datasets are provided by the cameras. To fully exploit the benefits of parallel computing, the confidence generation step and the CRD-Fusion network are implemented with a deep learning framework, PyTorch [75]. The pipeline is trained and evaluated on an NVIDIA V100 GPU, unless otherwise stated. After the raw disparity maps are available for all input frames, the confidence generation step is carried out. The confidence maps are computed according to the parameters outlined in Table 4.5a.

Once the stereo RGB images, raw disparity maps, and confidence maps are available, the CRD-Fusion network is ready to be trained and evaluated. Prior to sending the train-

γ_z	0.24	γ_m	-2.0
γ_s	0.01	λ_C	0.8
N_z size	3×3	N_m size	5×5

(a)

Parameter	Scene Flow/Custom datasets	KITTI 2012/2015
α_d	0.7	8.5
α_p	3	0.8
α_s	0.45	0.05
α_o	0.75	0.3

(b)

Table 4.5: Constant parameters for the proposed pipeline: (a) parameters used in confidence generation; (b) parameters to train CRD-Fusion on different datasets.

ing images to the network, data augmentation is performed for all input images. The brightness, contrast, saturation, and hue of the RGB images are randomly altered. Their pixel intensities are also normalized by the mean and variance of images from the large ImageNet [81] dataset. All training images are then randomly cropped to an input size of 256×512. Note that in evaluation, only intensity normalization is performed, and the images are padded such that their size is divisible by 2^k for $k \in \{0, 1, ..., K\}$. This padding strategy ensures the network can downsample and upsample the images properly.

The CRD-Fusion network is set up with $K = 3$ and $D_0 = 192$ disparity candidates at the full resolution. These settings result in $D_K = 24$ for cost volume construction and 1/8 resolution as the lowest image resolution in the model. The negative slope of all leaky ReLU activation function is fixed at 0.2. The model is trained with a batch size of 8 using an Adam [55] optimizer.

The model parameters are first randomly initialized and then trained with images from the Scene Flow dataset for 15 epochs with an initial learning rate of 0.001, which is later multiplied by 0.1 at the 10th epoch. Training on the KITTI 2012/2015 datasets and custom datasets does not start from scratch. These datasets contain significantly fewer images than Scene Flow does. Training from scratch using these smaller datasets is likely to lead to unsatisfactory results. Instead, the model trained with Scene Flow is further fine-tuned by these smaller datasets separately for evaluation on each of them. When fine-tuning the model on KITTI 2012/2015, training is performed for 1000 epochs with the learning rate at 0.0001. The learning rate is then halved every 200 epochs. When training with the RealSense dataset, the model is fine-tuned for 30 epochs with 0.0001 learning rate that is

divided by 2 at the 15th epoch. Lastly, data from the Zed dataset is applied to the model for 50 epochs with an initial learning rate of 0.0001, which is also halved at the 25th epoch. The weighting terms for the loss function are shown in Table 4.5b given different datasets.

4.5.3 Confidence Map Evaluation

The confidence map generation scheme proposed in (4.6) is evaluated with the Scene Flow test set. The evaluation is performed by first dividing all pixels into multiple sets $\{\mathcal{R}_1, \mathcal{R}_2, \mathcal{R}_3, ...\}$ according to each pixel's corresponding confidence score. For example, the confidence scores for all pixels in an arbitrary set $\mathcal{R}_i$ all fall within a pre-defined range. The endpoint error (EPE) for all pixels in $\mathcal{R}_i$ is then computed according to the raw disparity and ground truth via

$$EPE_i = \frac{1}{|\mathcal{R}_i|} \sum_{\mathbf{q}_i} \left| \tilde{\mathbf{D}}(\mathbf{q}_i) - \mathbf{D}^*(\mathbf{q}_i) \right|, \ \mathbf{q}_i \in \mathcal{R}_i, \tag{4.29}$$

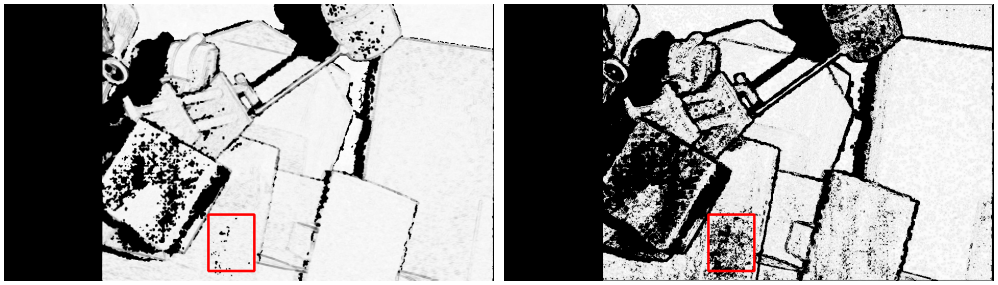
where $\mathbf{D}^*$ is the ground truth disparity map.

The evaluation results for five confidence ranges are shown in Table 4.6. Based on the results, the pixels with lower confidence scores generally have higher disparity errors. When the confidence scores reach 0.8, the EPE is as low as 0.65 px. From this evaluation, it can be concluded that the proposed confidence generation scheme is effective in quantifying the accuracy of raw disparity. Additionally, the chosen confidence threshold $\lambda_C = 0.8$ can be used to select accurate raw disparity while filtering out inaccuracies.

Confidence Range	0-0.2	0.2-0.4	0.4-0.6	0.6-0.8	0.8-1
Error (px)	5.20	5.52	4.83	2.10	0.65

Table 4.6: Average raw disparity error at different confidence ranges.

Figure 4.7 also shows some sample confidence outputs for the raw disparity map in Figure 4.1(a). The same textureless patch from Figure 4.3 is highlighted. In Figure 4.7(a) where the confidence map is computed based on ZSAD only, the confidence scores for the highlighted textureless region are high even though it contains many incorrect raw disparity estimates. By using the proposed method which incorporates both ZSAD and MND, the algorithm successfully identifies these incorrect raw disparity estimates and assigns low confidence scores to them as shown in Figure 4.7(b).



(a) (b)

Figure 4.7: Confidence maps computed by (a) ZSAD only and by (b) the proposed method. The red rectangle outlines a textureless region from the RGB image in Figure 4.3.

4.5.4 Design Analysis on Scene Flow

In order to demonstrate the benefits of different components of the pipeline, multiple configurations of the proposed approach are built and evaluated on the Scene Flow test set. The configuration with the best performance is chosen for further analysis on other datasets.

The first configuration, which is also the baseline (B) configuration, follows the same model structure as StereoNet [53]. Instead of supervised training, the baseline configuration is trained in a self-supervised manner by following

$$L_B = \frac{1}{(K+1)HW} \sum_{k=0}^{K} \left(\frac{1}{2^k} \left(\alpha_p L_{p,k}^{'} + \alpha_s L_{s,k} \right) \right), \tag{4.30}$$

where the photometric loss $L_{p,k}^{'}$ is modified based on (4.25) to exclude the confidence map and occlusion mask as

$$L_{p,k}^{'} = \sum_{\mathbf{p}^l} \left(\frac{\alpha}{2} \left(1 - SSIM \left(\mathbf{I}^l \left(\mathbf{p}^l \right), \hat{\mathbf{I}}_k^l \left(\mathbf{p}^l \right) \right) \right) + (1-\alpha) \left\| \mathbf{I}^l \left(\mathbf{p}^l \right) - \hat{\mathbf{I}}_k^l \left(\mathbf{p}^l \right) \right\| \right). \tag{4.31}$$

Additionally, to prevent $L_{s,k}$ from overpowering the training loss, α_s is changed to 0.045 for this configuration only. Built upon the baseline, the second configuration (B + DS) includes the raw disparity supervision loss without confidence guidance as

$$L_{B+DS} = \frac{1}{(K+1)HW} \sum_{k=0}^{K} \left(\frac{1}{2^k} \left(\alpha_d L_{d,k}^{'} + \alpha_p L_{p,k}^{'} + \alpha_s L_{s,k} \right) \right), \tag{4.32}$$

where the raw disparity supervision loss in this case does not include confidence as

$$L_{d,k}^{'} = \sum_{\mathbf{p}^l} \Delta_s \left(\mathbf{D}_k^0 \left(\mathbf{p}^l \right) - \tilde{D} \left(\mathbf{p}^l \right) \right). \tag{4.33}$$

The third configuration (B + DS + CL) includes the confidence guidance in addition to (4.32) with a resulting training loss of

$$L_{B+DS+CL} = \frac{1}{(K+1)HW} \sum_{k=0}^{K} \left(\frac{1}{2^k} \left(\alpha_d L_{d,k} + \alpha_p L_{p,k}^{''} + \alpha_s L_{s,k} \right) \right), \tag{4.34}$$

where the modified photometric loss only excludes the occlusion mask from (4.25) as

$$\begin{aligned} L_{p,k}^{''} = \sum_{\mathbf{p}^l} \Big(& \frac{\alpha}{2} \left(1 - SSIM \left(\mathbf{I}^l \left(\mathbf{p}^l \right), \hat{\mathbf{I}}_k^l \left(\mathbf{p}^l \right) \right) \right) \\ & + (1-\alpha) \left\| \mathbf{I}^l \left(\mathbf{p}^l \right) - \hat{\mathbf{I}}_k^l \left(\mathbf{p}^l \right) \right\| \Big) \left(1 - \mathbf{Cf} \left(\mathbf{p}^l \right) \right). \end{aligned} \tag{4.35}$$

By using the loss in (4.34), the fourth configuration (B + DS + CL + DF) further incorporates the dispariy fusion step shown in (4.14). These four configurations can verify the benefits introduced by the raw disparity supervision loss, confidence maps, and raw disparity fusion.

Another two configurations are also constructed to demonstrate the proposed occlusion-aware model design and they are both trained with (4.22). The fifth configuration (OR + DS + CL) adopts the proposed feature extraction module and the hierarchical occlusion-aware refinement module without any raw disparity fusion. In this configuration, (4.14) is omitted and the preliminary disparity map $\hat{\mathbf{D}}$ from (4.13) is treated as the initial disparity map $\mathbf{D}_i$ for refinement. The last configuration (OR + DS + CL + DF) is the proposed approach with the additional raw disparity fusion.

The accuracy of all these configurations are measured according to two metrics by following the previous work [95]. The first metric is EPE with respect to the entire predicted and ground truth disparity maps as

$$EPE = \frac{1}{HW} \sum_{\mathbf{p}^l} \left| \mathbf{D}_0 \left(\mathbf{p}^l \right) - \mathbf{D}^* \left(\mathbf{p}^l \right) \right|. \tag{4.36}$$

The second metric is the 3px-error rate (> 3 px), which is the percentage of pixels with L1 error larger than 3 px compared to the ground truth. Both metrics are calculated for

all test images and then averaged by the total number of test frames. The model is more accurate if they have lower values.

The quantitative and qualitative results for all configurations are presented in Table 4.7 and Figure 4.8, respectively. For each configuration, the same settings are used to repeat the experiment for seven times. Both the average and minimum metrics are reported in Table 4.7. Additionally, the 3px-error of the raw disparity generated by SGBM is also shown. Note that calculating EPE for raw disparity is ambiguous since the raw disparity maps contain many pixels with missing disparity predictions. Therefore, it is not included in the table.

Configuration	EPE (px)		> 3 px (%)	
	Avg	Min	Avg	Min
SGBM [71]	-	-	18.54	-
B	5.130	5.048	15.89	15.65
B + DS	3.997	3.962	13.33	13.19
B + DS + CL	3.515	3.383	13.02	12.65
B + DS + CL + DF	3.404	3.343	12.99	12.85
OR + DS + CL	3.838	3.112	16.84	13.08
OR + DS + CL + DF (Ours)	**2.647**	**2.606**	**11.02**	**10.79**

Table 4.7: Average and minimum accuracy on Scene Flow test set for different model configurations.

We first compare the results obtained from SGBM and from different configurations. According to the results in Table 4.7, the predicted disparity estimates from all six configurations are more accurate than raw disparity. The qualitative outputs show that the DNN can reason with contextual information to complete the missing estimates found in the raw disparity maps. These comparisons verify DNN as a more powerful tool for stereo matching.

Within all six configurations, the first four models based on the baseline network are then studied. The baseline configuration (B) is the least accurate one among them. Although this configuration can recover the general structure of the scene, it predicts disparity maps with incorrect estimates at object boundaries and occluded regions. After using the raw disparity supervision loss, the second configuration (B + DS) can compute substantially more accurate estimates. However, errors at the ambiguous regions are still significant. The accuracy is further improved by using confidence in the training loss in configuration B + DS + CL. With disparity fusion in configuration B + DS + CL +

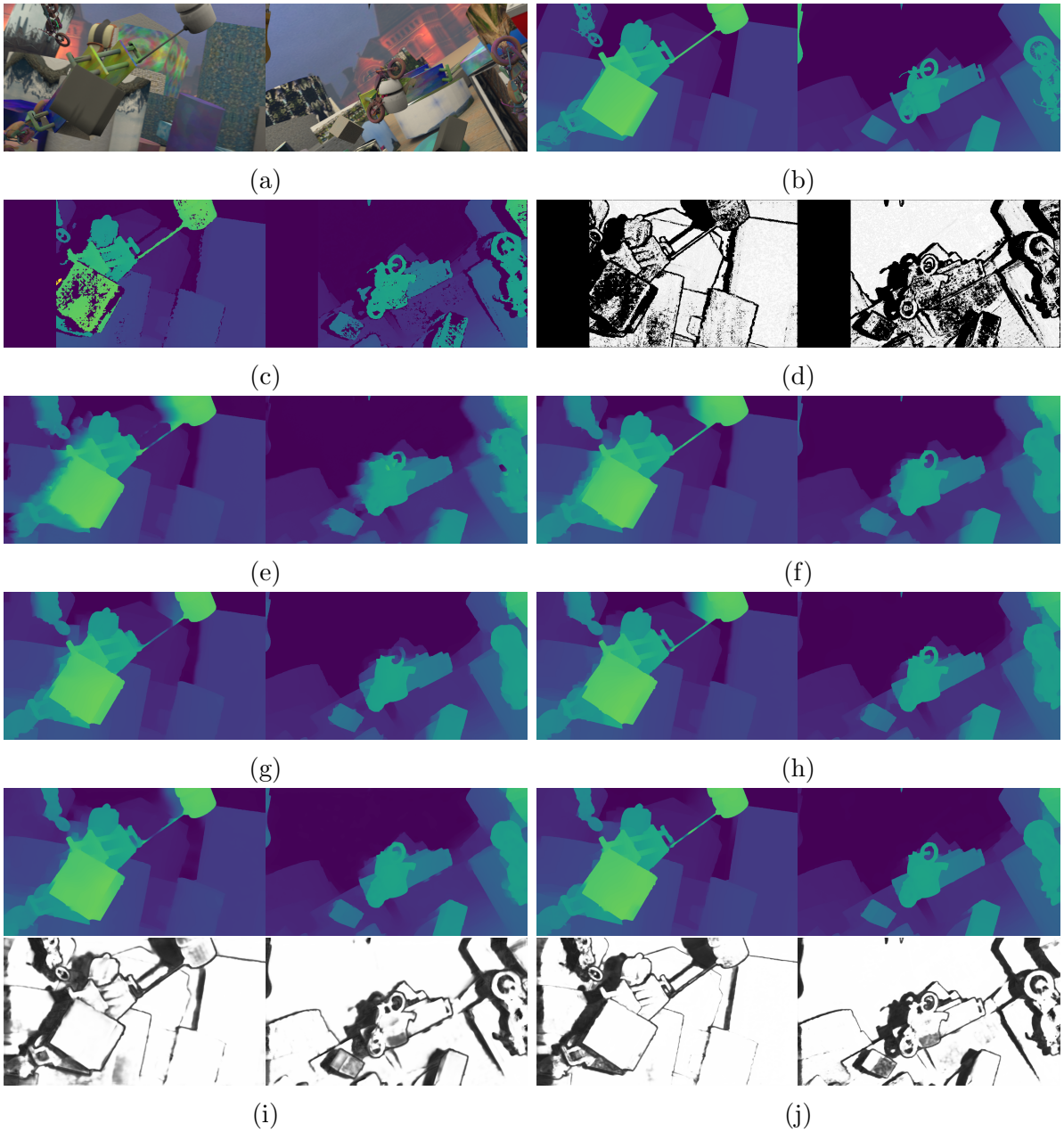


Figure 4.8: Sample images from the Scene Flow test set and their corresponding qualitative results: (a) left RGB images; (b) ground truth disparity; (c) raw disaprity from SGBM; (d) computed confidence maps; predictions from configurations (e) B, (f) B + DS, (g) B + DS + CL, (h) B + DS + CL + DF, (i) OR + DS + CL, (j) (OR + DS + CL + DF).

DF, the disparity error is reduced even more but with increased error at the occluded regions. This comparison shows that using the photometric loss (4.31) and smoothness loss (4.27) only cannot address disparity errors, especially at occluded areas. Training the network with the photometric loss (4.31) will guide the model to find disparity estimates even at occluded regions, which are not applicable to this loss. The resulting estimates for occluded pixels are typically inaccurate. The smoothness loss (4.27) further enforces constant disparity for these inaccuracies. Including raw disparity supervision, confidence guidance, and disparity fusion can remedy errors at these ambiguous regions. However, significant number of erroneous estimates are still present in the qualitative results.

To reduce error at occluded regions, we consider the configurations with our proposed occlusion-aware design. As shown in Figure 4.8(i), configuration OR + DS + CL can predict high-quality occlusion masks. The use of these masks greatly improves the disparity predictions at occluded areas compared to the configurations based on the baseline model. However, the occlusion masks cannot identify the occlusion information for certain intricate objects, which affects the accuracy at regions with many details. Additionally, the training of this configuration may become unstable. This stability issue reflects on the accuracy shown in Table 4.7, where the average accuracy of this configuration is worse than some baseline models while its minimum error is lower than that of the baseline. The training becomes more stable with the inclusion of raw disparity fusion. The strong prior knowledge from raw disparity helps configuration OR + DS + CL + DF achieve the best accuracy among all six configurations. Additionally, the predicted occlusion masks are also more defined and can capture more detailed information. The improved occlusion masks further boost the quality of the predicted disparity maps, which contain detailed predictions and accurate estimates at occluded regions.

Both the quantitative and qualitative analyses above compare different configurations of the proposed pipeline. They verify the effectiveness of different components in the approach, including confidence generation, the proposed feature extraction and occlusion-aware refinement modules, raw disparity fusion, and the self-supervised loss in (4.22). It can be concluded that the proposed occlusion-aware self-supervised stereo matching pipeline with confidence guided raw disparity fusion (configuration OR + DS + CL + DF) is an effective tool to predict both accurate disparity and occlusion masks.

4.5.5 Comparison with Existing Models on KITTI 2012/2015

A common practice in evaluating a stereo-based deep learning model is to verify its accuracy on the KITTI 2012 and KITTI 2015 datasets. This same approach is adopted in this thesis to compare the performance of the proposed pipeline with other existing methods.

	Method	D1-bg	D1-fg	D1-all	Runtime
Sup.	GC-Net [51]	2.21%	6.16%	2.87%	0.9 s
	PSMNet [9]	1.86%	4.62%	2.32%	0.41 s
	StereoNet [53]	4.30%	7.45%	4.83%	**0.015 s**
	GA-Net [103]	**1.55%**	**3.82%**	**1.93%**	0.36 s
	AANet [100]	1.99%	5.39%	2.55%	0.062 s
Self-sup.	Zhou et al. [109]	-	-	9.91%	0.39 s
	OASM-Net [59]	6.89%	19.42%	8.98%	0.73 s
	DispSegNet [104]	**4.20%**	16.97%	6.33%	0.9 s
	Flow2Stereo [62]	5.01%	14.62%	6.61%	0.05 s
	PASMnet [95]	5.41%	16.36%	7.23%	0.5 s
	Ours	4.59%	**13.68%**	**6.11%**	**0.02 s**

(a)

	Method	Out-Noc	Out-All	Avg-Noc	Avg-All	Runtime
Sup.	GC-Net [51]	1.77%	2.30%	0.6 px	0.7 px	0.9 s
	PSMNet [9]	1.49%	1.89%	**0.5 px**	0.6 px	0.41 s
	GA-Net [103]	**1.36%**	**1.80%**	**0.5 px**	**0.5 px**	0.36 s
	AANet [100]	1.91%	2.42%	**0.5 px**	0.6 px	0.06 s
Self-sup.	OASM-Net [59]	6.39%	8.60%	1.3 px	2.0 px	0.73 s
	DispSegNet [104]	4.68%	5.66%	**0.9 px**	**1.0 px**	0.9 s
	Flow2Stereo [62]	4.58%	**5.11%**	1.0 px	1.1 px	0.05 s
	Ours	**4.38%**	5.40%	**0.9 px**	1.1 px	**0.02 s**

(b)

Table 4.8: Quantitative results on the (a) KITTI 2015 and (b) KITTI 2012 datasets.

Table 4.8 summarizes the accuracy of the proposed model and other existing methods with a similar training strategy. The error metrics follow the same definition and nomenclature as the ones found on the KITTI online leaderboard [30, 31]. The accuracy in KITTI 2015 is evaluated with three quantities: D1-bg, D1-fg, D1-all. They are defined in the same way as the 3px-error rate introduced previously. However, D1-bg, D1-fg, and D1-all are applicable to background, foreground, and all pixels with valid ground truth labels, respectively [68]. In the KITTI 2012 comparison, the metrics include Out-Noc, Out-All, Avg-Noc, and Avg-All. Out-Noc and Out-All are the 3px-error rate with Out-Noc applied to the non-occluded regions only and Out-All applied to all pixels with ground truth labels. Avg-Noc and Avg-All are defined in the same way as (4.36), while the former is for

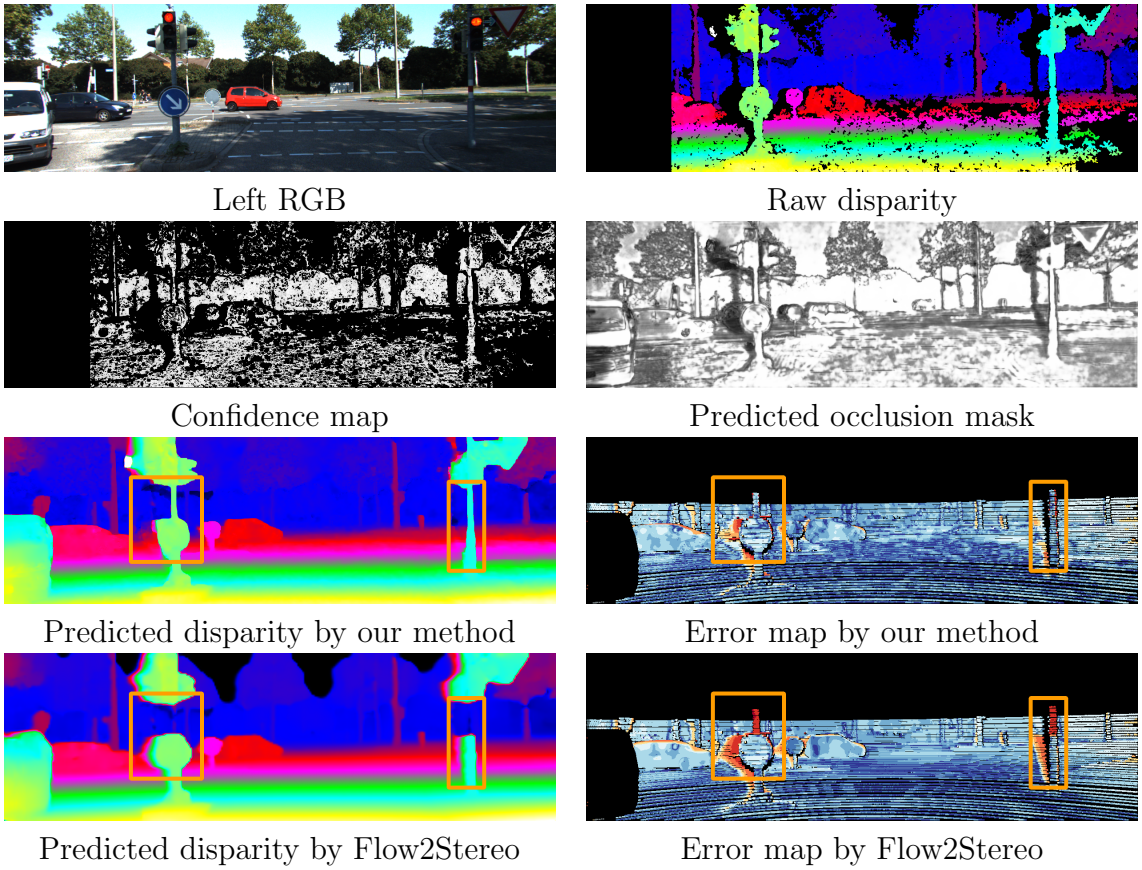


Figure 4.9: Sample qualitative results from the KITTI 2015 test set.

non-occluded pixels and the latter is for all pixels [32]. All of the results in Table 4.8 are obtained from either the online leaderboard or from the corresponding publication.

The quantitative results show that there is still a significant gap between the self-supervised approaches, including the proposed one, and the supervised methods. Nevertheless, the D1-bg metric in KITTI 2015 for the proposed pipeline is still comparable to the one for StereoNet [53], which is trained in a supervised manner. By comparing the self-supervised methods, it can be seen that the proposed one is considerably better than the approaches from [109], [59], and [95]. In KITTI 2012, the proposed pipeline, DispSetNet [104], and Flow2Stereo [62] have similar accuracy. while our pipeline is slightly more accurate at non-occluded areas. This may be due to the strong supervision from raw disparity at these regions. In KITTI 2015, the proposed approach is more accurate than other self-supervised methods according to multiple metrics. Additionally, the runtime of

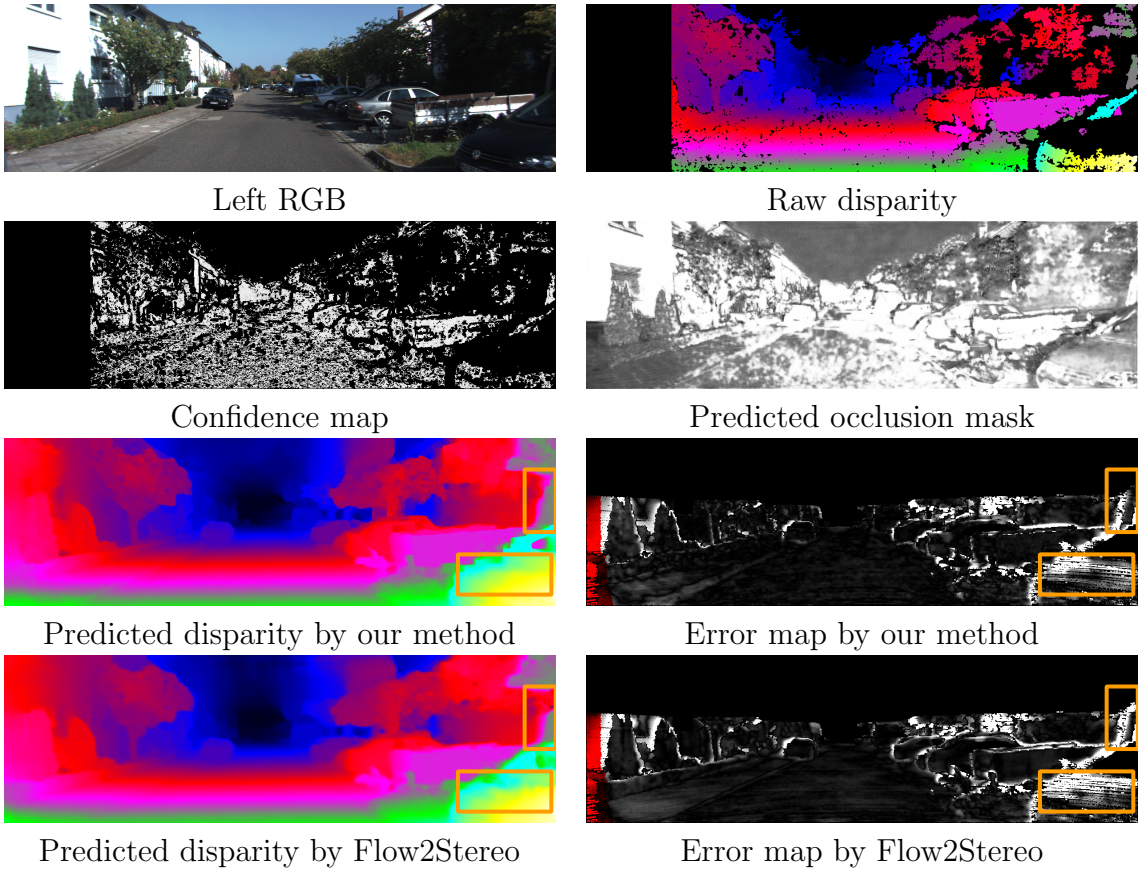


Figure 4.10: Sample qualitative results from the KITTI 2012 test set.

the pipeline is one of the lowest ones within all of the models studied here. These results show that the proposed approach is effective in predicting accurate disparity even for images recorded in real life. Although it still cannot outperform the supervised models, its accuracy is comparable or better than other self-supervised methods with a much lower runtime.

Sample qualitative results from both KITTI 2015 and KITTI 2012 can be found in Figure 4.9 and Figure 4.10, respectively. Although the ground truth disparity maps are not available in KITTI 2012/2015 test sets, the online leaderboard provides the error maps for the submitted predicted disparity. Red color in the error maps from Figure 4.9 indicates that the error is higher, while white color from the error maps in Figure 4.10 means higher error. Figure 4.9 shows that the proposed method can predict a clear occlusion mask which correctly identifies the occluded regions (e.g., occlusion at the light poles). This occlusion

mask can effectively improve disparity estimates at occluded regions, which results in better disparity predictions compared to Flow2Stereo [62]. This can be verified by the orange bounding boxes focusing on the occluded regions of the two light poles. In Figure 4.10, the improvements at occluded areas are not easily visible since the scene does not contain large occlusion. However, it shows that the proposed pipeline can predict better disparity at textureless regions, for instance, at the front hood of the black car.

The above quantitative and qualitative comparisons show that the proposed pipeline can predict accurate disparity with visually more satisfactory results as well as occlusion maps indicating the occluded regions correctly. In addition to accuracy, another important factor to consider is the model's runtime since it is designed to run in parallel with a commercial stereo camera. A commercial stereo camera is often capable of real-time processing. If the pipeline's frame rate is low, then the pipeline will not be able to exploit all of the information from the stereo camera. Hence, a detailed comparison of runtime is necessary to demonstrate if the pipeline is applicable for this use case.

Method	1660 Super	P100	V100
PSMNet [9]	1.218	2.304	2.658
StereoNet [53]	-	**61.03**	**90.77**
GA-Net [103]	0.128*	0.224	0.476
AANet [100]	6.677	9.953	17.14
PASMNet [95]	8.072	12.11	15.44
Ours	**38.92**	**31.20**	**46.74**

Table 4.9: Frame rate in fps of different approaches on KITTI 2015 test set. *Image size is fixed at 384×1248 in all tests with the exception of running GA-Net on 1660 Super. In this case, the input images are cropped to 336×1200 due to limited video memory.

The runtime shown in Table 4.8 is reported by the submitter of each method. The results are not directly comparable since they are obtained with different hardware setup. As a result, a fair runtime comparison was conducted and the results are summarized in Table 4.9. Multiple existing methods with open-source code and the proposed one are studied using different GPUs in this comparison. The runtime of processing 200 test frames from KITTI 2015 is summed up and the frame rate is computed from the total runtime. Note that the runtime for the proposed pipeline includes both the confidence generation step and the CRD-Fusion network. According to the results, StereoNet achieves the fastest frame rate, while the most accurate model shown in Table 4.8, GA-Net, is immensely slow. Although the proposed pipeline is slower than StereoNet due to the use of accurate but

complex refinement module, it is still faster than other models with more than 30 fps even on a mid-range NVIDIA 1660 Super GPU.

The above analysis on the KITTI 2012/2015 datasets indicates that the proposed pipeline can produce accurate disparity and occlusion masks in real-time. Its good balance of accuracy and runtime makes it an ideal candidate to run together with a commercial camera to achieve better depth perception.

4.5.6 Verification on Custom Datasets

Since the pipeline is designed to complement a commercial stereo camera, testing it with data from this camera is necessary. To fully verify the pipeline's effectiveness, it is evaluated on two custom datasets collected by two different stereo cameras: Intel RealSense D435 [52] and Zed Mini camera [85]. Since these two datasets do not include ground truth depth information, the evaluation is restricted to qualitative analysis only.

Figure 4.11 provides some sample results from the custom datasets. Although the raw disparity maps can provide fairly accurate depth information, they still contain lots of inaccuracies or missing predictions. For example, in the raw disparity map from the RealSense dataset, the yellow pixels at the occluded region caused by the stool are erroneous estimates. In the Zed dataset, the camera cannot predict disparity for the cardboard box located at the rightmost region of the image. Additionally, neither cameras can provide estimation of the leftmost columns since pixels there are not usually visible in the right frame.

By using the proposed pipeline, all of the problems mentioned above are resolved. The pipeline successfully identifies the incorrect estimates at the occluded regions of the RealSense image, and then it fills in the occluded areas with disparity from the background. The predicted disparity map for the Zed image also indicates that the pipeline can identify the disparity values of the cardboard box. The missing disparity information of the leftmost columns is also provided by the pipeline. Lastly, the predicted occlusion masks include clear occluded areas of the input left stereo images. This analysis demonstrates that the proposed pipeline has the capability to complement a commercial stereo camera to offer more accurate depth information.

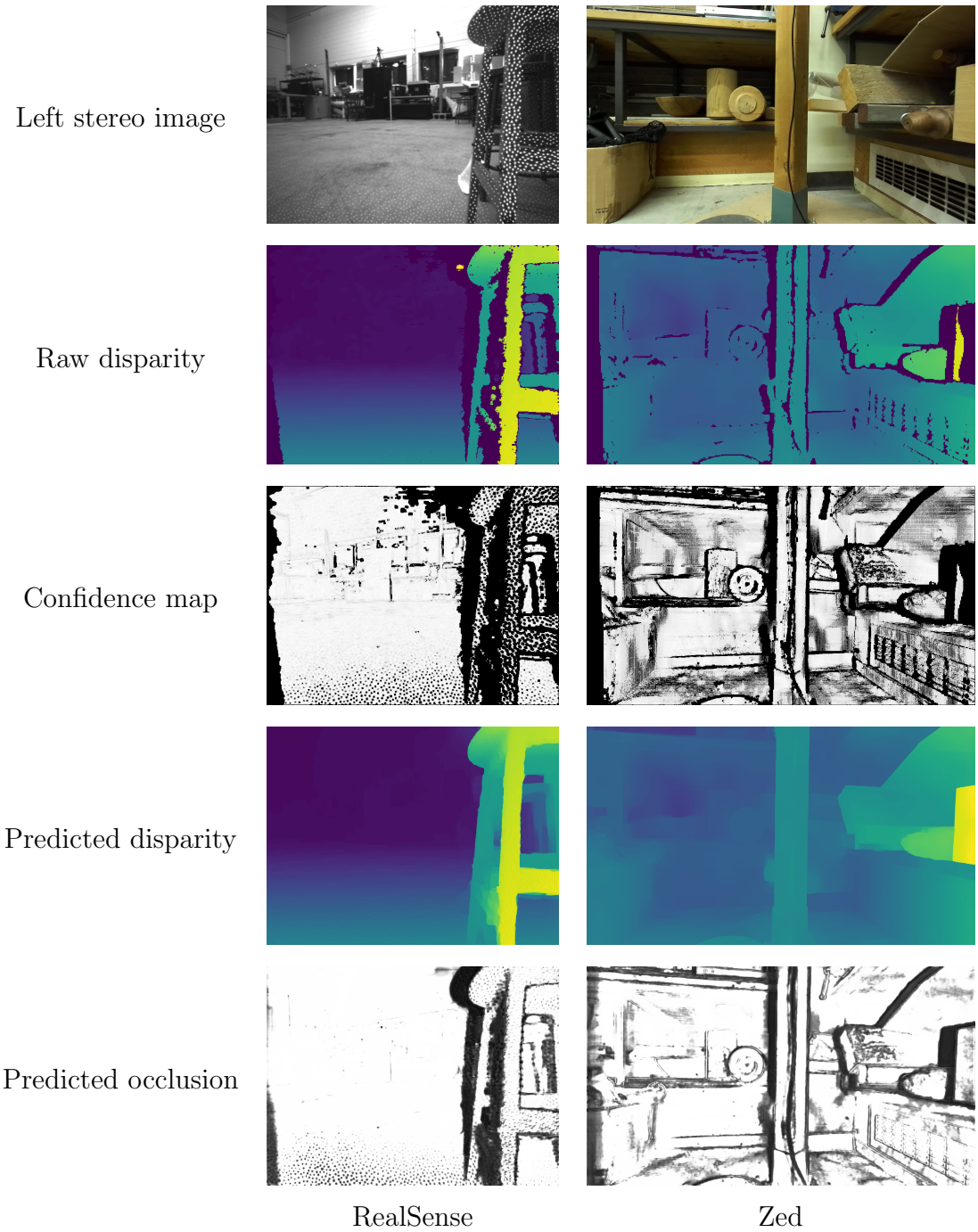


Figure 4.11: Sample qualitative results on test sets of the custom datasets

Chapter 5

Occlusion-Aware Disparity-based Direct Visual Servoing

In this chapter, a direct visual servoing scheme that utilizes the stereo matching pipeline proposed in the previous chapter is designed for a mobile robot. To eliminate the need of feature extraction and matching, which may lead to errors in visual servoing [16, 87], the proposed strategy uses the predicted disparity at each pixel location directly. The predicted occlusion masks are also incorporated to address the occlusion problem originated from the stereo setup.

This chapter first provides a derivation of the relationship between a stereo camera's velocity and the disparity map computed by images from this camera. This relationship allows us to design a disparity-based direct visual servoing controller. Next, the occlusion information is integrated into the control strategy. Lastly, the entire control framework is verified with extensive simulations and experiments.

5.1 The Proposed Visual Servoing Framework

5.1.1 Stereo Vision Modeling

A mobile robot equipped with a stereo camera is free to move in a workspace. During the robot's motion, the stereo camera is assumed to record stereo images continuously. The proposed stereo matching pipeline computes the disparity maps and occlusion masks with these stereo images. A schematic of the stereo camera capturing a pair of rectified images

is shown in Figure 5.1. In the schematic, a reference frame $\mathcal{O}$ is attached to the optical center of the stereo camera's left lens. At an arbitrary time instant t, consider an arbitrary 3D point, which is visible to both camera lenses, with coordinates $\mathbf{P} = (x_o, y_o, z_o) \in \mathbb{R}^3$ with respect to frame $\mathcal{O}$. The 3D point $\mathbf{P}$ is projected to both the left and right image planes of the camera at $\mathbf{p}_c^l$ and $\mathbf{p}_c^r$, respectively. The subscript c indicates that these image points are located in the continuous pixel space instead of discrete pixel locations used in the previous chapter. By using a pinhole camera model, the continuous coordinates of the projected point $\mathbf{p}_c^l = (v, u)$ on the left image plane are related to $\mathbf{P}$ as

$$\begin{cases} \bar{u} = \frac{u - c_u}{f} = \frac{x_o}{z_o} \\ \bar{v} = \frac{v - c_v}{f} = \frac{y_o}{z_o} \end{cases}, \tag{5.1}$$

where $\bar{u}$ and $\bar{v}$ denote the adjusted image coordinates, (c_v, c_u) is the principal point, i.e., the position where the optical axis $\mathbf{z}_\mathcal{O}$ intersects with the left image plane.

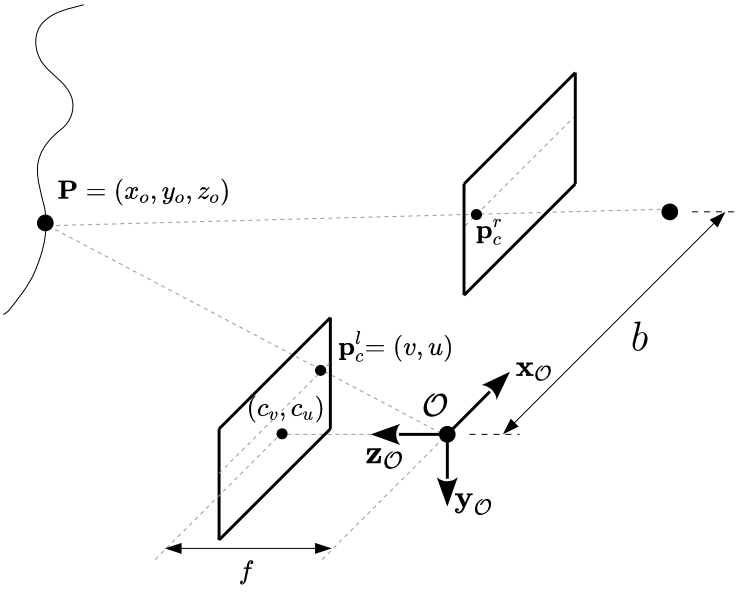


Figure 5.1: Schematic of a stereo camera capturing a pair of rectified images.

At the same left image plane, the proposed stereo matching pipeline can continuously compute the corresponding disparity maps and occlusion masks. Denote the disparity map $\mathbf{D}_0$ computed by the stereo matching pipeline at this specific time instant t as $\mathcal{D} \in \mathbb{R}_{>0}^{H \times W}$.

This disparity information can be further expressed in a vector form as

$$\mathbf{D} = \begin{bmatrix} \mathbf{D}(1) \\ \mathbf{D}(2) \\ \vdots \\ \mathbf{D}\left(W\left(\bar{v}_j - 1\right) + \bar{u}_i\right) \\ \vdots \\ \mathbf{D}(N) \end{bmatrix} = \begin{bmatrix} \mathcal{D}\left(\bar{v}_1, \bar{u}_1\right) \\ \mathcal{D}\left(\bar{v}_1, \bar{u}_2\right) \\ \vdots \\ \mathcal{D}\left(\bar{v}_j, \bar{u}_i\right) \\ \vdots \\ \mathcal{D}\left(\bar{v}_H, \bar{u}_W\right) \end{bmatrix} \in \mathbb{R}^N_{>0}, \tag{5.2}$$

where $N = WH$ is the total number of pixels in $\mathcal{D}$, $\bar{v}_j \in \{\bar{v}_1, \bar{v}_2, ..., \bar{v}_H\}$ and $\bar{u}_i \in \{\bar{u}_1, \bar{u}_2, ..., \bar{u}_W\}$ denote the discrete image coordinates (pixel indices) corresponding to the continuous adjusted image coordinates $(\bar{v}, \bar{u})$. $(\bar{v}_j, \bar{u}_i)$ is essentially the pixel coordinates (j, i) recorded at time instant t. Furthermore, $(\bar{v}_j, \bar{u}_i)$ also represents a whole-number location on the continuous pixel space formed by (v, u).

5.1.2 Disparity-based Direct Visual Servoing

The disparity-based visual servoing algorithm needs to compute the camera velocity to move the camera and robot such that the disparity map converges to the target one. By following the IBVS formulation introduced in [12, 46, 87], the instantaneous velocity $\mathbf{v} = \begin{bmatrix} v_{o,x} & v_{o,z} & \omega_{o,y} \end{bmatrix}^\top \in \mathbb{R}^3$ of the camera frame $\mathcal{O}$ and the temporal variation of $\mathbf{D}$ is defined as

$$\frac{\partial \mathbf{D}}{\partial t} = \mathcal{L}_{\mathbf{D}} \mathbf{v}, \tag{5.3}$$

where $\mathcal{L}_{\mathbf{D}} \in \mathbb{R}^{N \times 3}$ denotes the full interaction matrix based on disparity. Since the camera is attached to a mobile robot whose motion is limited on a 2D plane, the camera's velocity only contains two translational components $v_{o,x}$ and $v_{o,y}$ along the $\mathbf{x}_\mathcal{O}$ and $\mathbf{z}_\mathcal{O}$ axis, respectively, as well as the rotational component $\omega_{o,y}$ about the $\mathbf{y}_\mathcal{O}$ axis. Consider the error vector

$$\Delta\mathbf{D} = \begin{bmatrix} \mathbf{D}(1) - \mathbf{D}_{ref}(1) \\ \vdots \\ \mathbf{D}(W(\bar{v}_j - 1) + \bar{u}_i) - \mathbf{D}_{ref}(W(\bar{v}_j - 1) + \bar{u}_i) \\ \vdots \\ \mathbf{D}(N) - \mathbf{D}_{ref}(N) \end{bmatrix}, \tag{5.4}$$

where $\mathbf{D}_{ref} \in \mathbb{R}^N_{>0}$ denotes the vector form of the reference disparity map captured at the robot's target pose. The proposed velocity control law is designed with a goal to minimize

$e = \|\Delta\mathbf{D}\|^2/2$ along (5.3), by guaranteeing

$$\frac{\partial e}{\partial t} = (\Delta\mathbf{D})^\top \frac{\partial \mathbf{D}}{\partial t} = (\Delta\mathbf{D})^\top \mathcal{L}_\mathbf{D}\mathbf{v} < 0 \tag{5.5}$$

for nonzero $\mathcal{L}_\mathbf{D}^\top \Delta\mathbf{D}$. Hence, the corresponding control law is chosen as

$$\mathbf{v} = -\lambda \mathcal{L}_\mathbf{D}^+ \Delta\mathbf{D}, \tag{5.6}$$

where $\mathcal{L}_\mathbf{D}^+ = \left(\mathcal{L}_\mathbf{D}^\top \mathcal{L}_\mathbf{D}\right)^{-1} \mathcal{L}_\mathbf{D}^\top$ is the pseudoinverse of $\mathcal{L}_\mathbf{D}$. Although the global stability of controller (5.6) is not guaranteed, the local asymptotic stability is ensured within a neighborhood of $\mathbf{D} = \mathbf{D}_{ref}$ when $\mathcal{L}_\mathbf{D}$ is of column rank 3 [12]. In the actual implementation, $\mathcal{L}_\mathbf{D}^+$ is computed by using singular value decomposition, which eliminates any potential problem for matrix inversion.

In order to derive the full interaction matrix $\mathcal{L}_D$, we follow a procedure similar to that in [87]: The depth z_o captured at time instant t is modeled as a surface $z_o(\bar{v}, \bar{u}, t)$. The disparity is also considered as a time varying surface $D(\bar{v}, \bar{u}, t)$. Note that the disparity image $\mathcal{D}$ and its vectorized form $\mathbf{D}$ are essentially the discrete representations of this surface. Taking the full time derivative of $D(\bar{v}, \bar{u}, t)$ yields

$$\frac{dD}{dt} = \frac{\partial D}{\partial \bar{u}}\dot{\bar{u}} + \frac{\partial D}{\partial \bar{v}}\dot{\bar{v}} + \frac{\partial D}{\partial t}, \tag{5.7}$$

which can be rearranged as

$$\frac{\partial D}{\partial t} = \frac{dD}{dt} - \frac{\partial D}{\partial \bar{u}}\dot{\bar{u}} - \frac{\partial D}{\partial \bar{v}}\dot{\bar{v}}. \tag{5.8}$$

By triangulation shown in Figure 5.2, the depth $z_o(\bar{v}, \bar{u}, t)$ of $\mathbf{P}$ from the left optical center is geometrically related to the disparity $D(\bar{v}, \bar{u}, t)$. By using the similar triangles in Figure 5.2 and following (2.1), this relationship is

$$D(\bar{v}, \bar{u}, t) = \frac{bf}{z_o(\bar{v}, \bar{u}, t)}, \tag{5.9}$$

which implies that

$$\frac{dD}{dt} = -\frac{bf}{z_o^2}\dot{z}_o. \tag{5.10}$$

Additionally, time derivatives of the adjusted image coordinates are obtained by differentiating (5.1) as

$$\begin{cases} \dot{\bar{u}} = \frac{\dot{x}_o z_o - x_o \dot{z}_o}{z_o^2} \\ \dot{\bar{v}} = \frac{\dot{y}_o z_o - y_o \dot{z}_o}{z_o^2}. \end{cases} \tag{5.11}$$

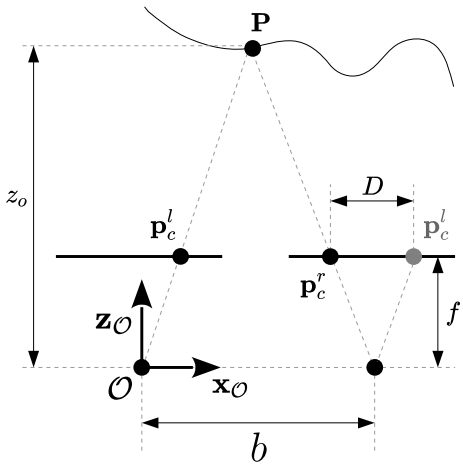


Figure 5.2: Top view of Figure 5.1 to visualize triangulation.

The velocity components $(\dot{x}_o, \dot{y}_o, \dot{z}_o)$ of the 3D point $\mathbf{P}$ in (5.11) can be related to the camera velocity. This relationship has been derived in [12] as

$$\begin{cases} \dot{x}_o = -v_{o,x} - \omega_{o,y} z_o + \omega_{o,z} y_o \\ \dot{y}_o = -v_{o,y} - \omega_{o,z} x_o + \omega_{o,x} z_o \\ \dot{z}_o = -v_{o,z} - \omega_{o,x} y_o + \omega_{o,y} x_o \end{cases}, \tag{5.12}$$

where the camera velocity $(v_{o,x}, v_{o,y}, v_{o,z}, \omega_{o,x}, \omega_{o,y}, \omega_{o,z})$ allows the camera to move in a 3D workspace with 6 DOFs. This suggests that (5.12) is applicable to camera integrated with a robot manipulator. Since the camera motion in our case is constrained in a 2D plane, this relationship can be simplified to accommodate planar motion by setting $v_{o,y} = \omega_{o,x} = \omega_{o,z} = 0$ as

$$\begin{cases} \dot{x}_o = -v_{o,x} - \omega_{o,y} z_o \\ \dot{y}_o = 0 \\ \dot{z}_o = -v_{o,z} + \omega_{o,y} x_o \end{cases}. \tag{5.13}$$

Substituting (5.1), (5.9), and (5.13) into (5.10) yields

$$\frac{dD}{dt} = -\frac{bf}{\left(\frac{bf}{D}\right)^2} \left(-v_{o,z} + \omega_{o,y} x_o\right),$$

$$\frac{dD}{dt} = -\frac{D^2}{bf} \left(-v_{o,z} + \omega_{o,y} z_o \bar{u}\right),$$

$$\frac{dD}{dt} = -\frac{D^2}{bf}\left(-v_{o,z} + \omega_{o,y}\frac{bf}{D}\bar{u}\right),$$
$$\frac{dD}{dt} = \frac{D^2}{bf}v_{o,z} - D\bar{u}\omega_{o,y}. \tag{5.14}$$

Similarly, (5.11) can also be expanded as

$$\dot{\bar{u}} = \frac{\left(-v_{o,x} - \omega_{o,y}z_o\right)z_o - x_o\left(-v_{o,z} + \omega_{o,y}x_o\right)}{z_o^2},$$
$$\dot{\bar{u}} = -\frac{v_{o,x}}{z_o} - \omega_{o,y} + \frac{x_o}{z_o^2}v_{o,z} - \frac{x_o^2}{z_o^2}\omega_{o,y},$$
$$\dot{\bar{u}} = -\frac{v_{o,x}}{\frac{bf}{D}} - \omega_{o,y} + \frac{D}{bf}\bar{u}v_{o,z} - \bar{u}^2\omega_{o,y},$$
$$\dot{\bar{u}} = -\frac{D}{bf}v_{o,x} + \frac{D\bar{u}}{bf}v_{o,z} - \left(\bar{u}^2 + 1\right)\omega_{o,y}. \tag{5.15}$$
$$\dot{\bar{v}} = \frac{-y_o\left(-v_{o,z} + \omega_{o,y}x_o\right)}{z_o^2},$$
$$\dot{\bar{v}} = \frac{y_o}{z_o^2}v_{o,z} - \frac{x_o y_o}{z_o^2}\omega_{o,y},$$
$$\dot{\bar{v}} = \frac{D\bar{v}}{bf}v_{o,z} - \bar{u}\bar{v}\omega_{o,y}. \tag{5.16}$$

Substituting (5.14), (5.15), and (5.16) into (5.8), we obtain

$$\frac{\partial D}{\partial t} = \begin{bmatrix} \frac{D}{bf}\frac{\partial D}{\partial \bar{u}} \\ \frac{D}{bf}\left(D - \bar{u}\frac{\partial D}{\partial \bar{u}} - \bar{v}\frac{\partial D}{\partial \bar{v}}\right) \\ -D\bar{u} + \left(\bar{u}^2 + 1\right)\frac{\partial D}{\partial \bar{u}} + \bar{u}\bar{v}\frac{\partial D}{\partial \bar{v}} \end{bmatrix}^\top \begin{bmatrix} v_{o,x} \\ v_{o,z} \\ \omega_{o,y} \end{bmatrix}. \tag{5.17}$$

From (5.17), we define an interaction manifold, which is also referred to as interaction matrix in previous works, based on an arbitrary point on the disparity surface $D\left(\bar{u}, \bar{v}, t\right)$ as

$$\mathbf{L}_D = \begin{bmatrix} \frac{D}{bf}\frac{\partial D}{\partial \bar{u}} \\ \frac{D}{bf}\left(D - \bar{u}\frac{\partial D}{\partial \bar{u}} - \bar{v}\frac{\partial D}{\partial \bar{v}}\right) \\ -D\bar{u} + \left(\bar{u}^2 + 1\right)\frac{\partial D}{\partial \bar{u}} + \bar{u}\bar{v}\frac{\partial D}{\partial \bar{v}} \end{bmatrix}^\top. \tag{5.18}$$

The full interaction matrix $\mathcal{L}_{\mathbf{D}}$ is constructed as a stack of interaction manifolds $\mathbf{L}_D$ as

$$\mathcal{L}_{\mathbf{D}} = \begin{bmatrix} \mathbf{L}_D\left(\bar{v}_1, \bar{u}_1, t\right) \\ \vdots \\ \mathbf{L}_D\left(\bar{v}_j, \bar{u}_i, t\right) \\ \vdots \\ \mathbf{L}_D\left(\bar{v}_H, \bar{u}_W, t\right) \end{bmatrix}, \tag{5.19}$$

where $\mathbf{L}_D\left(\bar{v}_j, \bar{u}_i, t\right)$ is the interaction manifold evaluated with the discrete pixel indices $(\bar{v}_j, \bar{u}_i)$ and $\mathbf{D}(W(\bar{v}_j - 1) + \bar{u}_i)$ recorded at time instant t. In the actual implementation, the adjusted image coordinates $(\bar{v}, \bar{u})$ given pixel $(\bar{v}_j, \bar{u}_i)$ are evaluated via (5.1) by setting $u = \bar{u}_i$ and $v = \bar{v}_j$. Additionally, the spatial gradients $\frac{\partial D}{\partial \bar{u}}$ and $\frac{\partial D}{\partial \bar{u}}$ can also be computed based on (5.1) as

$$\begin{cases} \frac{\partial D}{\partial \bar{u}} = \frac{\partial D}{\partial u}\frac{\partial u}{\partial \bar{u}} = f\frac{\partial D}{\partial u} \\ \frac{\partial D}{\partial \bar{v}} = \frac{\partial D}{\partial v}\frac{\partial v}{\partial \bar{v}} = f\frac{\partial D}{\partial v} \end{cases}, \tag{5.20}$$

where $\frac{\partial D}{\partial u}$ and $\frac{\partial D}{\partial v}$ are approximated by applying the Sobel filters to the disparity $\mathbf{D}(W(\bar{v}_j - 1) + \bar{u}_i)$ located at $(\bar{v}_j, \bar{u}_i)$.

In the above derivation of $\mathcal{L}_{\mathbf{D}} \in \mathbb{R}^{N\times 3}$, the camera motion is restricted to a 2D plane since it is attached to a mobile robot. Similar procedures can be followed to derive a more generic $\mathcal{L}_{\mathbf{D}} \in \mathbb{R}^{N\times 6}$ for applications with robot manipulators by using (5.12) instead of (5.13).

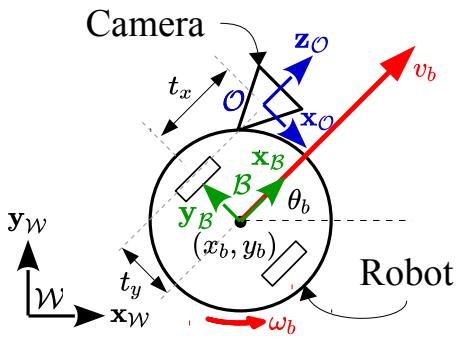


Figure 5.3: Model of a mobile robot with a camera.

After obtaining the full interaction matrix $\mathcal{L}_{\mathbf{D}}$, the camera velocity $\mathbf{v}$ can be computed by the control law (5.6). However, $\mathbf{v}$ is based on frame $\mathcal{O}$ attached to the camera. Since

this camera frame does not always align with the robot's base, moving the robot directly with this velocity will cause undesired pose. Hence, $\mathbf{v}$ needs to be transformed to the robot's base frame in order to provide the appropriate velocity command to the robot. To obtain this transformation, consider Figure 5.3 that shows the model of a nonholonomic differential drive mobile robot with a camera and a reference base frame $\mathcal{B}$ attached to its base. The robot's motion is governed by

$$\begin{bmatrix} \dot{x}_b \\ \dot{y}_b \\ \dot{\theta}_b \end{bmatrix} = \begin{bmatrix} v_b \cos\theta_b \\ v_b \sin\theta_b \\ \omega_b \end{bmatrix}, \tag{5.21}$$

where (x_b, y_b) is the position of the robot's base in the world reference frame $\mathcal{W}$, and θ_b is its orientation with respect to axis $\mathbf{x}_{\mathcal{W}}$. The linear velocity of the robot's base v_b is inline with axis $\mathbf{x}_{\mathcal{B}}$, while the rotational velocity ω_b is about axis $\mathbf{z}_{\mathcal{B}}$. The camera's velocity $\mathbf{v}$ is related to the robot's velocity $\mathbf{v}_b = \begin{bmatrix} v_b & \omega_b \end{bmatrix}^\top$ in frame $\mathcal{B}$ [27] by

$$\mathbf{v} = \mathbf{T}\mathbf{v}_b = \begin{bmatrix} 0 & -t_x \\ 1 & -t_y \\ 0 & -1 \end{bmatrix} \mathbf{v}_b, \tag{5.22}$$

where t_x and t_y are the offsets of the camera from $\mathcal{B}$ as shown in Figure 5.3. Hence, given the camera's velocity from (5.6), the control law in the form of the robot's velocity is

$$\mathbf{v}_b = \mathbf{T}^+\mathbf{v}, \tag{5.23}$$

where $\mathbf{T}^+$ is the pseudoinverse of the transformation $\mathbf{T}$.

5.2 Occlusion-Aware Visual Servoing and Final Design

Due to the setup of a stereo camera, occluded regions are only visible in one of the stereo views but not in the other. Some stereo algorithms remove the disparity estimates in these regions from their predictions. In the case of the proposed stereo matching pipeline, contextual information is utilized to provide these estimates. Since they are not obtained according to the general principle of stereo matching, they may not reflect accurate depth information. Including them in the control law may affect the overall performance. Therefore, they should be removed from the visual servoing algorithm.

The removal of occluded regions is facilitated by the predicted occlusion mask $\mathbf{O}_o$ computed at the same time as the predicted disparity $\mathbf{D}_0$ by the stereo matching pipeline. For the disparity image vector $\mathbf{D}$ at time instant t, denote the corresponding occlusion mask $\mathbf{O} \in \mathbb{R}^N$, which has been vectorized in a similar way as (5.2). Given the pixel $(\bar{v}_j, \bar{u}_i)$, $\mathbf{O}\left(W(\bar{v}_j - 1) + \bar{u}_i\right) \in (0, 1)$ indicates how likely this pixel belongs to an non-occluded region according to the model's knowledge. When $\mathbf{O}\left(W(\bar{v}_j - 1) + \bar{u}_i\right)$ is close to 1, the model is certain that $(\bar{v}_j, \bar{u}_i)$ is from a non-occluded region, while $\mathbf{O}\left(W(\bar{v}_j - 1) + \bar{u}_i\right)$ close to 0 indicates the opposite. This soft occlusion mask can be used to effectively select non-occluded pixels to be included in the controller.

To incorporate this occlusion information at time instant t and at the robot's target pose, both the occlusion mask $\mathbf{O}$ and the vectorized occlusion mask $\mathbf{O}_{ref} \in \mathbb{R}^N$ obtained at the target pose are used to compute a joint occlusion mask by

$$\bar{\mathbf{O}} = \mathbf{O} \otimes \mathbf{O}_{ref} \in \mathbb{R}^N. \tag{5.24}$$

The soft joint occlusion mask $\bar{\mathbf{O}}$ can be viewed as a weighting vector based on the occlusion predictions at both time instant t and at the target pose for all pixels. In [87], a different weighting scheme is used in the control law to address the problem where structure of the scene changes during the visual servoing process. By extending this technique, the controller in (5.6) is modified to include the joint occlusion mask as

$$\mathbf{v} = -\lambda \left(\bar{\mathbf{O}}^D \mathcal{L}_{\mathbf{D}}\right)^+ \left(\bar{\mathbf{O}}^D \Delta \mathbf{D}\right), \tag{5.25}$$

where $\bar{\mathbf{O}}^D$ is a diagonalized matrix form of $\bar{\mathbf{O}}$ as

$$\bar{\mathbf{O}}^D = \text{diag}\left(\bar{\mathbf{O}}\left(1\right), ..., \bar{\mathbf{O}}\left(N\right)\right) \in \mathbb{R}^{N \times N} \tag{5.26}$$

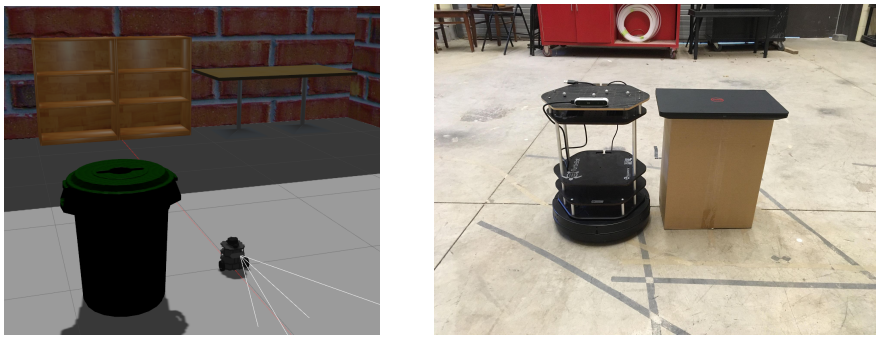
Equation (5.25) explicitly lowers the contribution of pixels that are identified as part of the occluded regions to the calculated camera velocity.

To summarize the above derivation and design, a stereo camera installed on a mobile robot captures a pair of stereo images and a raw disparity map at each arbitrary time step. These images are processed by the proposed stereo matching pipeline, which has been trained offline. The pipeline estimates a disparity map and an occlusion mask online. The disparity map and occlusion mask estimates are exploited on the fly to obtain the interaction matrix (5.19), the error signal (5.4), and the joint occlusion mask (5.24). Then (5.25) utilizes all this information to compute the camera's commanded velocity. Lastly, the camera velocity is transformed to the robot's velocity command according to (5.23).

5.3 Simulation and Experimental Results

5.3.1 System Setup

The proposed visual servoing algorithm is verified in both a simulation environment and on a physical robot, as shown in Figure 5.4, with the use of Robot Operating System [78]. In the simulation environment, a Turtlebot 3 [70] with a simulated Intel RealSense D435 camera [52] is used, while a Turtlebot 2 [69] with a RealSense D415 camera [52] is utilized for the physical experiments. Both of the robots are nonholonomic differential drive mobile robots. The simulated camera captures stereo images at a resolution of 600×800 with focal length $f = 435$ px and baseline $b = 0.05$ m. The local block matching method from the OpenCV library [72] then computes the raw disparity maps from the stereo images. In the physical experiments, the camera outputs stereo images and raw disparity maps directly at a resolution of 480×640 with $f = 608.3$ px and $b = 0.055$ m. Note that the raw disparity maps are converted from the raw depth maps computed by the camera's processor.



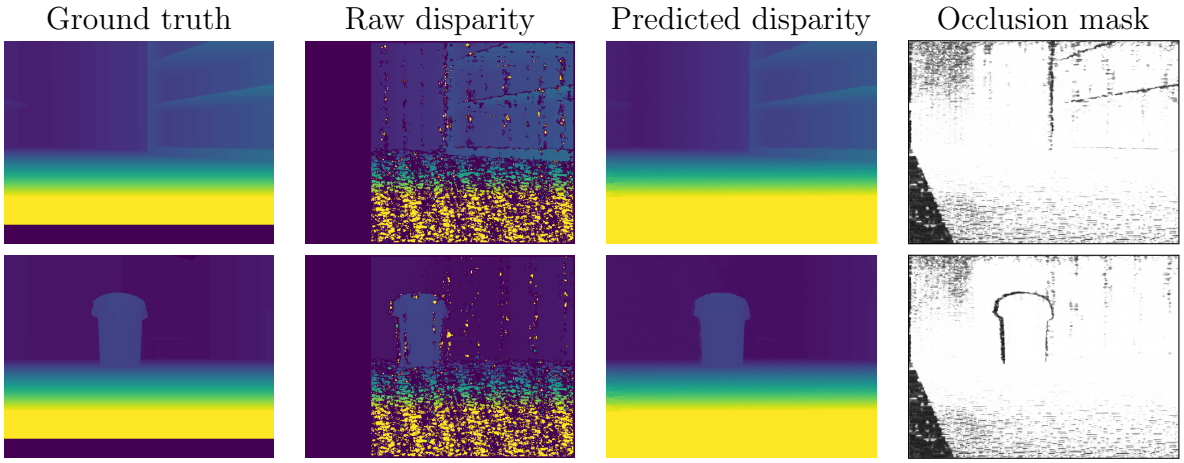
(a) (b)

Figure 5.4: Setup in the (a) simulation and (b) physical environments.

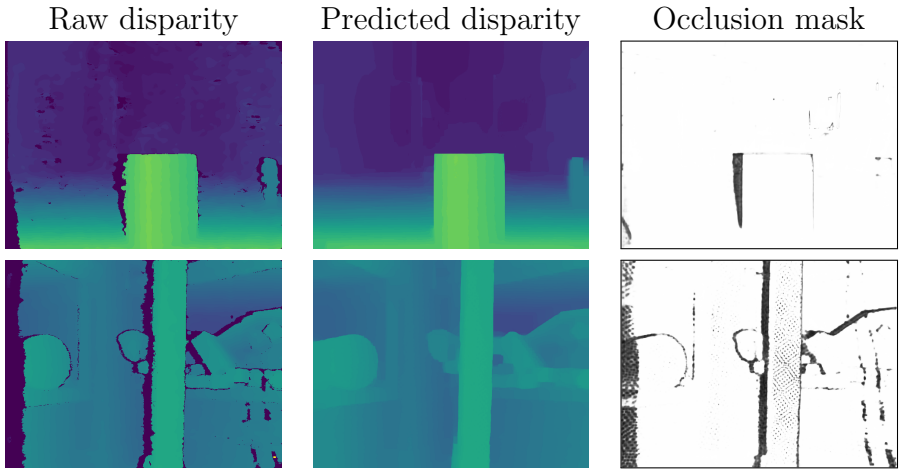
After obtaining the stereo images and raw disparity maps, the proposed stereo matching pipeline computes the final predicted disparity maps and occlusion masks. In order to lower the runtime, the images are cropped to $W = 320$ and $H = 240$ at the center prior to feeding them into the stereo pipeline. The predicted disparity and occlusion are utilzied by the controller with its gain chosen as $\lambda = 30$ for the simulated controller and $\lambda = 5$ in the experiments. The robot's ground truth poses in experiments are captured by an Optitrack camera tracking system.

With the above implementation details, the simulation can run smoothly on a desktop with an Intel i7-10700K CPU and an NVIDIA RTX 3060 GPU at approximately 0.075

sec per iteration. In the experiments, a laptop with an Intel i5-7500HQ CPU and an NVIDIA GTX 1050 GPU is connected to the robot to perform all computations. Due to the hardware limitation, the runtime in the experiments is slightly lower at around 0.095 sec per iteration.



(a)



(b)

Figure 5.5: Sample qualitative results from the (a) simulated and (b) real datasets.

The stereo pipeline discussed previously is trained and evaluated with images captured in scenes different from the simulation or physical environment. Using the trained model directly in the current use cases may not yield accurate disparity estimates. To improve

the performance of the stereo pipeline, the model pretrained on the Scene Flow dataset is fine-tuned with datasets recorded in the simulation and experimental settings separately, leading to two trained models for both settings. The dataset collected in the simulation environment consists of 5,056 frames, while there are 12,814 frames in the dataset from the physical environment. For both datasets, 80% of the images are considered as the training images, while the remaining ones are reserved to validate the trained model. When training with the simulation dataset, the model is fine-tuned with an initial learning rate of 0.0001 for 40 epochs. The learning rate is reduced to 0.00005 at the 20th epoch. Fine-tuning with the real images is performed for 30 epochs with the same initial learning rate, which is lowered to 0.00001 at the 15th epoch. Other hyperparameters are the same as the ones used for training with the Scene Flow dataset according to Table 4.5.

After fine-tuning, the stereo pipeline can predict accurate disparity and occlusion in these environments. When evaluating the model on the simulation dataset's validation set, the EPE is only 0.3301 px and only 0.3852% of pixels have disparity error larger than 3 px. Since no ground truth data is available in the real dataset, only the qualitative results are given in Figure 5.5. The qualitative results in Figure 5.5 show that the proposed stereo pipeline can provide high-quality disparity and occlusion predictions in the simulation and physical environments.

5.3.2 Simulation Results

In the simulation environment, three sets of tests have been performed. The first test is designed to verify the proposed controller in (5.25). Then the performance between the proposed scheme and visual servoing methods based on feature extraction is compared. The last test justifies the importance of occlusion masks in the controller.

Verification of the Proposed Design

The proposed disparity-based visual servoing scheme is verified by a positioning task in the simulation environment. The robot is first placed at a target pose. Then the predicted disparity map and occlusion mask at this pose are recorded as the reference images. By using these reference images, the robot moves from its initial pose according to the velocity commands computed by the controller. During the robot's motion, its position and heading angle are recorded.

Figure 5.6 shows the results from this test. The task error shown in Figure 5.6(a) is

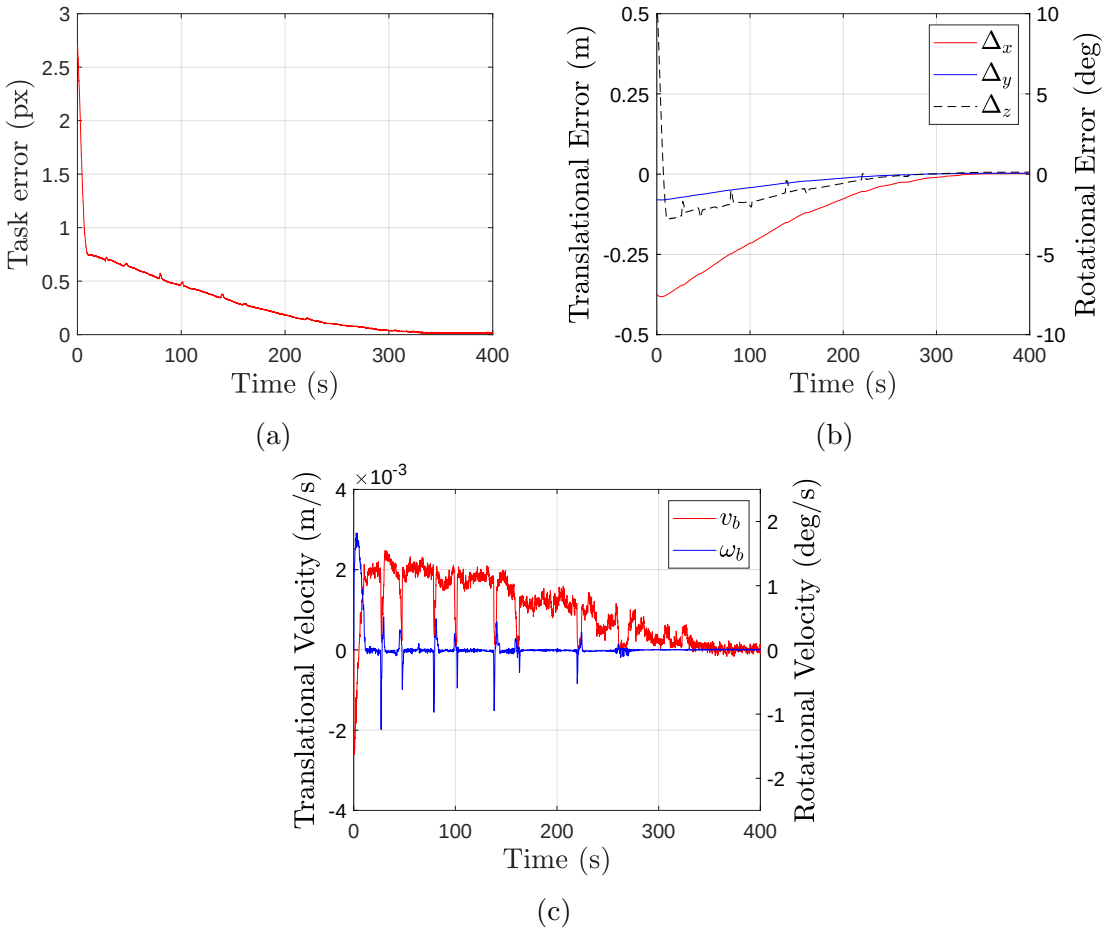


Figure 5.6: Results from the positioning task in the simulation environment: (a) task error; (b) positioning error; (c) robot's velocity.

defined as

$$e_d = \frac{1}{N} \sum_{n=1}^{N} \bar{\mathbf{O}}\,(n)\,|\Delta \mathbf{D}(n)|\,. \tag{5.27}$$

This task error computes the difference between the robot's current disparity map and the target one with occlusion considerations. The positioning error shown in Figure 5.6(b) includes three elements: the translational difference Δ_x between the robot's position and the target position along the $\mathbf{x}_{\mathcal{W}}$ axis in Figure 5.3, the translational error Δ_y along axis $\mathbf{y}_{\mathcal{W}}$, and rotational error Δ_z about the $\mathbf{z}_{\mathcal{W}}$ axis that can be derived by applying the right-

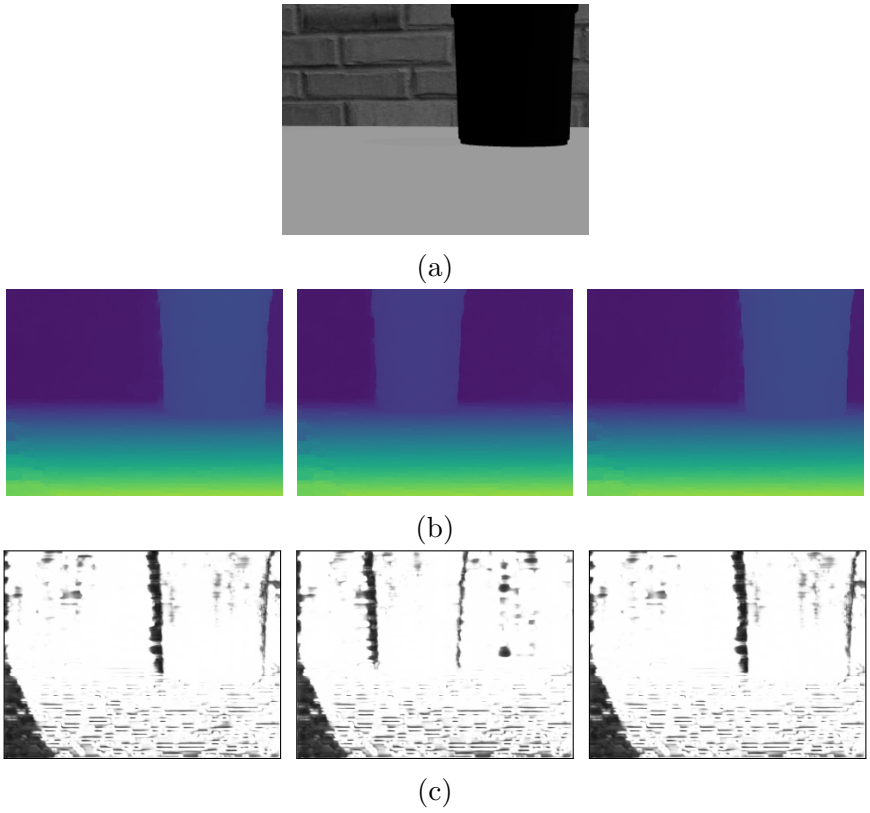


Figure 5.7: (b) Disparity maps and (b) occlusion masks captured in the (a) simulated scene for the positioning task. For (b) and (c), images from left to right are captured at the robot's target, initial, and final pose.

hand rule to $\mathbf{x}_{\mathcal{W}}$ and $\mathbf{y}_{\mathcal{W}}$. The linear and rotational velocities of the robot's base are also shown in Figure 5.6(c).

Figure 5.6(b) shows that the proposed controller in (5.25) can provide the appropriate control actions to move the robot towards its target pose. After the robot reaches its steady state pose at approximately $t = 300$ sec, the translational and rotational errors are in close vicinity of zero, i.e., the robot reaches its target pose. The task error shown in Figure 5.6(a) also decreases as the robot moves towards its target. At steady state, the task error is close to zero, which indicates that the robot's disparity map at steady state is almost identical to the target disparity map at the non-occluded regions. This result is also supported by the images shown in Figure 5.7, where the robot's final disparity map and final occlusion mask are visually similar to the reference images. These results verify that

the proposed controller can utilize the predicted disparity and occlusion information to compute the appropriate control commands. These commands are effective in regulating the robot's positioning error between its current pose and the target pose as well as the difference between the current and target disparity maps.

Comparison with Feature-based Approaches

In the second test, the performance of the proposed direct disparity-based controller is compared with two other approaches relying on feature extraction and matching. The controller design of these two baseline methods is based on the formulation given in (2.2). In these methods, a set of features are first extracted from the left stereo image recorded at the robot's target pose. The same set of features are extracted and matched from the current left stereo image. The image coordinates of these features are used as the image feature properties $\mathbf{s}$ and $\mathbf{s}^*$ in (2.2). The interaction matrix $\mathcal{L}_\mathbf{s}$ based on these properties has been derived in [46]. In the first baseline method, the SIFT features [63] are extracted from the images, while the oriented FAST and rotated BRIEF (ORB) descriptors [80] are used in the second baseline. For simplicity, the extraction and matching algorithms provided by open-source libraries [5, 92] are used directly in the implementation.

	Δ_x (m)	Δ_y (m)	Δ_z (°)
Initial error	-0.375	-0.080	10.873
SIFT	-0.232	-0.040	1.337
ORB	0.013	-0.004	0.555
Proposed	**0.002**	**0.003**	**0.124**

Table 5.1: Initial and final positioning error for different visual servoing approaches.

Table 5.1 shows the final positioning error obtained by different controllers used in this test. Compared to the initial positioning error, all three controllers can compute commanded velocity to move the robot closer to the target. The controller relying on SIFT feature yields the least accurate result. After the robot reaches its steady state with the SIFT-based controller, the robot's pose is still significantly different from the target one. After replacing SIFT features with ORB descriptors in the algorithm, the positioning error is reduced substantially. The proposed direct disparity-based controller achieves the best result with final positioning error close to zero. These results indicate that our designed controller has more superior performance than the ones relying on feature extraction and matching. When using feature-based controllers, the performance is dependent on the accuracy of the feature extraction and matching algorithms, which may match non-identical

features together. Using the direct visual servoing approach can avoid these potential inaccuracies.

Significance of Occlusion Masks

The controller in (5.25) incorporates occlusion information to address the limitation caused by the stereo setup. In this test, we verify the importance of this occlusion information in the control strategy. We compare the robot's positioning error from the proposed controller and from the one without occlusion information as shown in (5.6).

We observe that the proposed stereo pipeline may not compute correct disparity estimates at the occluded regions, especially when the objects in the scene are close to the camera. As shown in the example in Figure 5.8, the predicted disparity map includes notable error next to the left object boundary in the scene. Meanwhile, the occlusion mask can effectively identify this erroneous region as an occluded region.

(a) (b) (c)

Figure 5.8: Images recorded at the robot's target pose for occlusion study: (a) left stereo view; (b) predicted disparity; (c) occlusion mask.

By using the images shown in Figure 5.8 as the reference images, a positioning experiment is performed with results shown in Figure 5.9. When using the controller without occlusion information in (5.6), the robot does not converge to the desired pose according to Figure 5.9(b). The incorrect disparity at the occluded region contaminates the calculated velocity, leading to unsatisfactory positioning performance. As the occlusion mask removes the contribution of incorrect disparity in the control algorithm according to (5.25), the robot can move to the target pose with the final positioning error close to zero as shown in Figure 5.9(a). This comparison verifies the effectiveness of the proposed controller to address the occlusion problem inherited from a stereo setup.

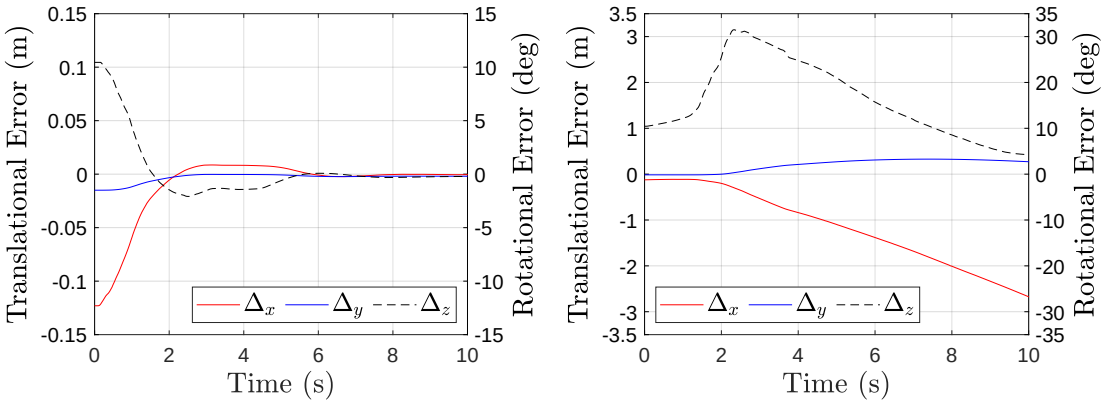


Figure 5.9: Positioning error for the robot (a) with and (b) without the proposed occlusion-aware control law.

5.3.3 Experimental Results

The proposed controller has also been tested on a physical robot to verify its effectiveness in a real-life environment. A positioning experiment similar to the one performed in the simulation environment is carried out. Additionally, a navigation experiment is also included to show the controller's performance in a longer run.

Positioning Experiment

In the positioning experiment, the experimental procedure is similar to the one used in the simulation test. The results of this experiment are shown in Figure 5.10. Using the reference images, the robot moves towards the target pose in the experiment and reaches close proximity to zero in terms of the positioning error in three dimensions as found in Figure 5.10(b). The task error in Figure 5.10(a) also reduces to approximately zero during the experiment. This can be further verified by Figure 5.11. The final disparity map and occlusion mask in Figure 5.11 are visually similar to the reference ones. These experimental results demonstrate that the proposed direct disparity-based controller is also applicable to a robot in a real-life environment. Furthermore, the robot used here is a differential-drive robot with dimensions and sensor setup different from those of the simulated robot. Therefore, the results further indicate that the proposed control algorithm is generic, and it can be applied to different mobile robotic systems with differential drive.

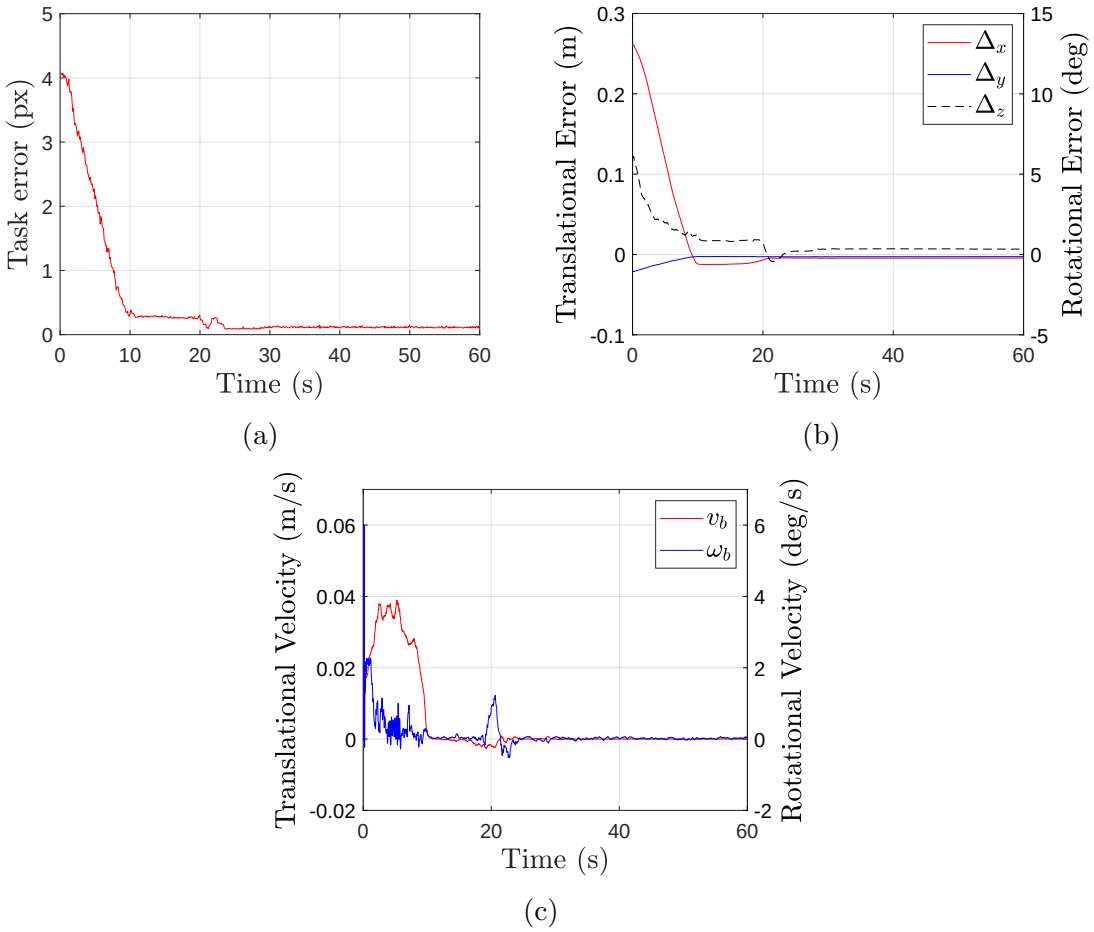


Figure 5.10: Results from the positioning task in the physical environment: (a) task error; (b) positioning error; (c) robot's velocity.

Navigation Experiment

In the navigation experiment, the robot is required to move to a sequence of poses in the environment shown in Figure 5.12(a). The robot is first placed at a sequence of six different poses. At each pose, a disparity map and an occlusion mask are recorded, resulting in a sequence of reference disparity maps in Figure 5.12(b) and a sequence of reference occlusion masks in Figure 5.12(c). At the beginning of the experiment, the robot starts from an initial pose. The controller uses the first set of disparity map and occlusion mask from the reference image sequences as the current target images. When the robot moves, the task

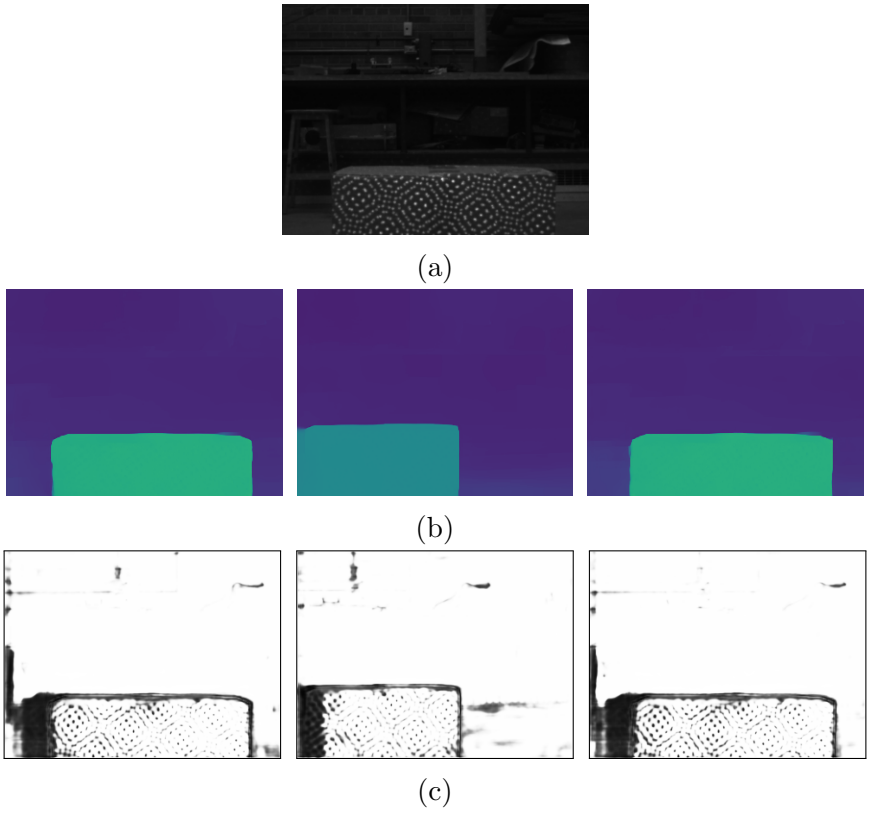
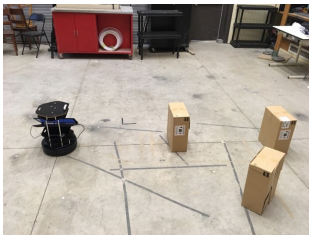


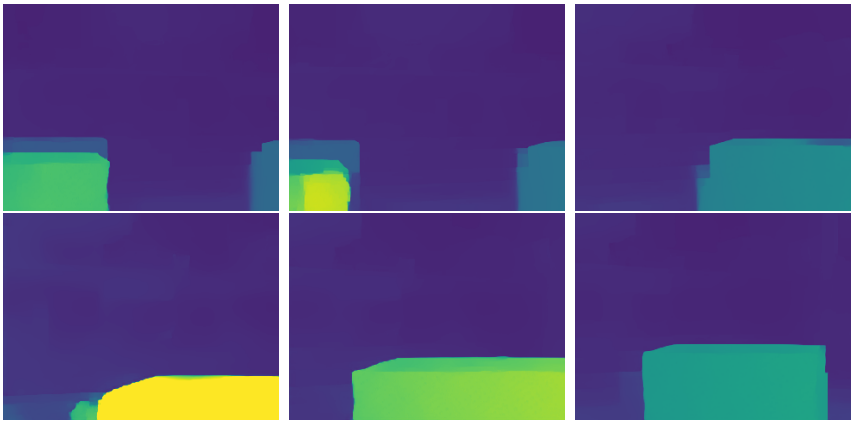
Figure 5.11: (b) Disparity maps and (c) occlusion masks captured in the (a) physical scene for the positioning task. For (b) and (c), images from left to right are captured at the robot's target, initial, and final pose.

error in (5.27) is computed online with the current and target images. When the task error is less than 0.3 px, the second set of disparity map and occlusion mask from the recorded sequences is used as the new reference images. The same procedure continues until the robot reaches its final pose.

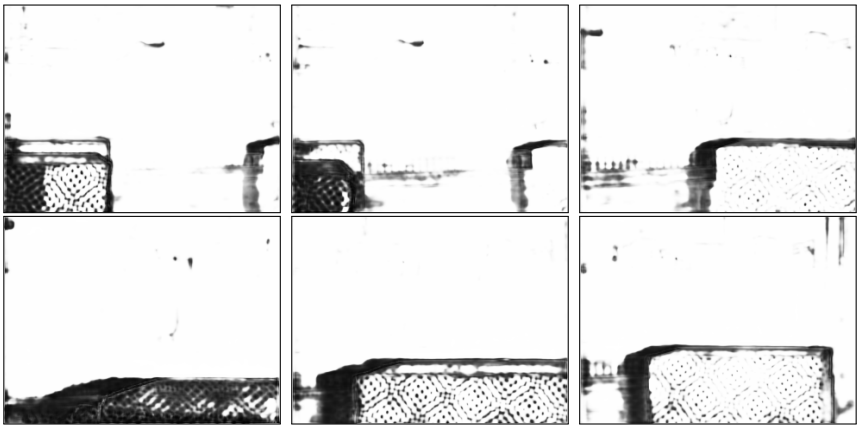
Figure 5.13 shows the robot's target poses on the $\mathbf{x}_{\mathcal{W}}$-$\mathbf{y}_{\mathcal{W}}$ plane in red, selected actual robot poses in blue, and the approximate obstacle locations in black. From this result, it can be observed that the robot can move towards the target poses in a sequential manner by using the proposed controller. When the robot reaches close proximity to a target pose, the reference images are switched automatically to guide the robot towards the next pose. This experiment demonstrates that the proposed direct disparity-based controller can be used to move the robot according to a sequence of reference images, which indicates the



(a)



(b)



(c)

Figure 5.12: Reference (b) disparity and (c) occlusion sequences recorded in the (a) navigation experiment setup. The order of the images in (b) and (c) is in the clockwise direction from the upper left image.

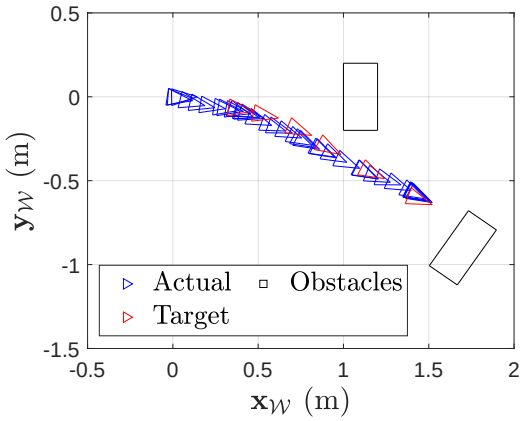


Figure 5.13: Actual robot poses and target poses from the navigation experiment. Obstacles' poses are approximate.

potential use of this control strategy in longer runs.

Figure 5.13 further shows that it is more difficult for the robot to correct its lateral error than its longitudinal error. This behavior may be attributed to the robot's nonholonomic motion constraints. To correct its lateral error, the robot is required to move along its longitudinal direction and about its rotational axis. However, these constraints are not considered in the control law shown in (5.25). Incorporating them in the controller design [15, 44] may lead to better performance.

Chapter 6

Conclusion

6.1 Summary

In this thesis, we have presented a self-supervised stereo matching pipeline with confidence guided raw disparity fusion and a direct disparity-based visual servoing framework. The stereo matching pipeline first calculates the confidence of a raw disparity map from a traditional stereo matching algorithm/commercial stereo camera, and then predicts an accurate disparity image and occlusion mask by fusing the raw disparity and confidence into a self-supervised deep neural network. Extensive experiments on public datasets demonstrate the effectiveness of each design component in the pipeline. Comparison with results from existing methods highlights the competitive accuracy and fast inference rate of the proposed approach. Further experiments with custom datasets collected by commercial stereo cameras also show that our method can effectively improve disparity computed by these cameras.

The proposed direct disparity-based visual servoing framework utilizes the high-quality predicted dispairty maps obtained from the stereo pipeline. Unlike many other visual servoing approaches, the proposed control framework does not rely on complex image feature extraction and matching, which are not always accurate and may lead to unsatisfactory performance. Moreover, the occlusion masks are also included to address the occlusion problem inherited from a stereo camera setup. The proposed control algorithm has been applied to mobile robots in both a simulation and physical environment. The simulation and experimental results show high positioning accuracy of the controller.

6.2 Future Work

There are multiple potential future research directions which may improve the stereo matching pipeline and the visual servoing framework. To further increase the accuracy of the stereo matching pipeline, different DNN designs and confidence guided raw disparity fusion schemes can be studied. Although CNNs are widely used in deep learning models designed for computer vision tasks, using alternative DNN architectures like recurrent neural networks (RNNs) and transformers may also be explored. Designing the stereo matching pipeline according to these architectures is another potential solution to achieve high-quality depth perception. Furthermore, the proposed confidence guided raw disparity fusion step may be included at different stages of the pipeline to further exploit the supervisory signals from raw disparity.

In addition to different network designs, tuning the hyperparameters in the pipeline may be improved by developing some systematic procedures as alternatives to the current ad-hoc approaches. For this improvement, an automatic hyperparameter tuning algorithm according to properties of the input information may be designed. Enhancing the tuning process can maintain the high-quality predictions when fine tuning the pipeline on new datasets. Moreover, this may also allow online adaptation of the pipeline, leading to improvement of its accuracy continuously on the fly.

In robotic applications, using light and fast algorithms is essential. Hence, the runtime and memory footprint of the stereo matching pipeline may be further reduced by applying various model compression techniques to the DNN. Improving the runtime of the pipeline is beneficial for other downstream processes relying on the pipeline. An algorithm with low memory footprint can be deployed on robots more easily since the capacity of their hardware may be limited.

For the visual servoing framework, better performance may be achieved if more factors are included in the controller design. For instance, the robot's nonholonomic motion constraints may be included in the control design to improve the tracking performance in the lateral direction. Considering the robot's dynamical model when designing the controller can better address disturbances that occur when the robot moves. Lastly, using more advanced control strategies, such as linear-quadratic (LQ) optimal control, model predictive control (MPC), and neuro-adaptive control, may result in higher tracking accuracy.

References

[1] M.D. Adams. Coaxial range measurement - current trends for mobile robotic applications. *IEEE Sensors Journal*, 2(1):2–13, 2002.

[2] M. Bakthavatchalam, O. Tahri, and F. Chaumette. A direct dense visual servoing approach using photometric moments. *IEEE Transactions on Robotics*, 34(5):1226–1239, 2018.

[3] Q. Bateux and E. Marchand. Histograms-based visual servoing. *IEEE Robotics and Automation Letters*, 2(1):80–87, 2017.

[4] H. Bay, T. Tuytelaars, and L. Van Gool. SURF: Speeded up robust features. In *Proc. European Conference on Computer Vision*, pages 404–417, 2006.

[5] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 25(11):120–123, 2000.

[6] J. Bridle. Training stochastic model recognition algorithms as networks can lead to maximum mutual information estimation of parameters. In *Proc. Advances in Neural Information Processing Systems*, pages 1–7, 1989.

[7] C. Cai, E. Dean-León, D. Mendoza, N. Somani, and A. Knoll. Uncalibrated 3D stereo image-based dynamic visual servoing for robot manipulators. In *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 63–70, 2013.

[8] V. Casser, S. Pirk, R. Mahjourian, and A. Angelova. Depth prediction without the sensors: Leveraging structure for unsupervised learning from monocular videos. *Proc. AAAI Conference on Artificial Intelligence*, 33:8001–8008, 2019.

[9] J. Chang and Y. Chen. Pyramid stereo matching network. In *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5410–5418, 2018.

[10] F. Chaumette. Potential problems of unstability and divergence in image-based and position-based visual servoing. In *Proc. European Control Conference*, pages 4549–4554, 1999.

[11] F. Chaumette. Image moments: a general and useful set of features for visual servoing. *IEEE Transactions on Robotics*, 20(4):713–723, 2004.

[12] F. Chaumette and S. Hutchinson. Visual servo control. I. Basic approaches. *IEEE Robotics & Automation Magazine*, 13(4):82–90, 2006.

[13] Y. Chen, C. Schmid, and C. Sminchisescu. Self-supervised learning with geometric constraints in monocular video: Connecting flow, depth, and camera. In *Proc. IEEE/CVF International Conference on Computer Vision*, pages 7062–7071, 2019.

[14] X. Cheng, Y. Zhong, M. Harandi, Y. Dai, X. Chang, H. Li, T. Drummond, and Z. Ge. Hierarchical neural architecture search for deep stereo matching. In *Proc. Advances in Neural Information Processing Systems*, volume 33, pages 22158–22169, 2020.

[15] A. Cherubini, F. Chaumette, and G. Oriolo. Visual servoing for path reaching with nonholonomic robots. *Robotica*, 29(7):1037–1048, 2011.

[16] C. Collewet and E. Marchand. Photometric visual servoing. *IEEE Transactions on Robotics*, 27(4):828–834, 2011.

[17] J. Dai, H. Qi, Y. Xiong, Y. Li, G. Zhang, H. Hu, and Y. Wei. Deformable convolutional networks. In *Proc. IEEE International Conference on Computer Vision*, pages 764–773, 2017.

[18] N. Djelal, N. Saadia, and A. Ramdane-Cherif. Target tracking based on SURF and image based visual servoing. In *Proc. International Conference on Communications, Computing and Control Applications*, pages 1–5, 2012.

[19] A. Dosovitskiy, P. Fischer, E. Ilg, P. Häusser, C. Hazirbas, V. Golkov, P. van der Smagt, D. Cremers, and T. Brox. FlowNet: Learning optical flow with convolutional networks. In *Proc. IEEE International Conference on Computer Vision*, pages 2758–2766, 2015.

[20] P. Dovesi, M. Poggi, L. Andraghetti, M. Martí, H. Kjellström, A. Pieropan, and S. Mattoccia. Real-time semantic stereo matching. In *Proc. IEEE International Conference on Robotics and Automation*, pages 10780–10787, 2020.

[21] G. Egnal and R. P. Wildes. Detecting binocular half-occlusions: empirical comparisons of five approaches. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(8):1127–1133, 2002.

[22] D. Eigen, C. Puhrsch, and R. Fergus. Depth map prediction from a single image using a multi-scale deep network. In *Proc. Advances in Neural Information Processing Systems*, volume 27, page 2366–2374, 2014.

[23] B. Espiau, F. Chaumette, and P. Rives. A new approach to visual servoing in robotics. *IEEE Transactions on Robotics and Automation*, 8(3):313–326, 1992.

[24] P. Fankhauser, M. Bloesch, D. Rodriguez, R. Kaestner, M. Hutter, and R. Siegwart. Kinect v2 for mobile robot navigation: Evaluation and modeling. In *Proc. International Conference on Advanced Robotics*, pages 388–394, 2015.

[25] M. Ferrera, A. Boulch, and J. Moras. Fast stereo disparity maps refinement by fusion of data-based and model-based estimations. In *Proc. International Conference on 3D Vision*, pages 9–17, 2019.

[26] S. Foix, G. Alenya, and C. Torras. Lock-in time-of-flight (ToF) cameras: A survey. *IEEE Sensors Journal*, 11(9):1917–1926, 2011.

[27] W. Fu, H. Hadj-Abdelkader, and E. Colle. Visual servoing based mobile robot navigation able to deal with complete target loss. In *Proc. International Conference on Methods & Models in Automation & Robotics*, pages 502–507, 2013.

[28] Z. Fu and M. A. Fard. Learning confidence measures by multi-modal convolutional neural networks. In *Proc. IEEE Winter Conference on Applications of Computer Vision*, pages 1321–1330, 2018.

[29] R. Garg, V. Kumar B.G, G. Carneiro, and I. Reid. Unsupervised cnn for single view depth estimation: Geometry to the rescue. In *Proc. European Conference on Computer Vision*, pages 740–756, 2016.

[30] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun. Stereo evaluation 2012. http://www.cvlibs.net/datasets/kitti/eval_stereo_flow.php?benchmark=stereo. accessed: Jun. 15, 2022.

[31] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun. Stereo evaluation 2015. http://www.cvlibs.net/datasets/kitti/eval_scene_flow.php?benchmark=stereo. accessed: Jun. 15, 2022.

[32] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? The KITTI vision benchmark suite. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages 3354–3361, 2012.

[33] R. Girshick. Fast R-CNN. In *IEEE International Conference on Computer Vision*, pages 1440–1448, 2015.

[34] C. Godard, O. M. Aodha, and G. J. Brostow. Unsupervised monocular depth estimation with left-right consistency. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages 6602–6611, 2017.

[35] C. Godard, O. M. Aodha, M. Firman, and G. Brostow. Digging into self-supervised monocular depth estimation. In *Proc. IEEE/CVF International Conference on Computer Vision*, pages 3827–3837, 2019.

[36] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, Cambridge, MA, 2016.

[37] R. Haeusler, R. Nair, and D. Kondermann. Ensemble learning for confidence measures in stereo vision. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages 305–312, 2013.

[38] G.D. Hager, W.-C. Chang, and A.S. Morse. Robot feedback control based on stereo vision: towards calibration-free hand-eye coordination. In *Proc. IEEE International Conference on Robotics and Automation*, volume 4, pages 2850–2856, 1994.

[39] K. He, X. Zhang, S. Ren, and J. Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(9):1904–1916, 2015.

[40] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[41] H. Hirschmuller. Stereo processing by semiglobal matching and mutual information. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(2):328–341, 2008.

[42] F. Hoffmann, T. Nierobisch, T. Seyffarth, and G. Rudolph. Visual servoing with moments of SIFT features. In *Proc. IEEE International Conference on Systems, Man and Cybernetics*, volume 5, pages 4262–4267, 2006.

[43] X. Hu and P. Mordohai. A quantitative evaluation of confidence measures for stereo vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(11):2121–2133, 2012.

[44] Y. Huang and J. Su. Visual servoing of nonholonomic mobile robots: A review and a novel perspective. *IEEE Access*, 7:134968–134977, 2019.

[45] Z. Huang, T. B. Norris, and P. Wang. ES-Net: An efficient stereo matching network. *arXiv:2103.03922*, 2021.

[46] S. Hutchinson, G.D. Hager, and P.I. Corke. A tutorial on visual servo control. *IEEE Transactions on Robotics and Automation*, 12(5):651–670, 1996.

[47] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proc. International Conference on Machine Learning*, volume 37, pages 448–456, 2015.

[48] M. Jaderberg, K. Simonyan, A. Zisserman, and Kavukcuoglu K. Spatial transformer networks. In *Proc. Advances in Neural Information Processing Systems*, volume 28, page 2017–2025, 2015.

[49] S. Joung, S. Kim, K. Park, and K. Sohn. Unsupervised stereo matching using confidential correspondence consistency. *IEEE Transactions on Intelligent Transportation Systems*, 21(5):2190–2203, 2020.

[50] H. Jung, Y. Kim, D. Min, C. Oh, and K. Sohn. Depth prediction from a single image with conditional adversarial networks. In *Proc. IEEE International Conference on Image Processing*, pages 1717–1721, 2017.

[51] A. Kendall, H. Martirosyan, S. Dasgupta, P. Henry, R. Kennedy, A. Bachrach, and A. Bry. End-to-end learning of geometry and context for deep stereo regression. In *Proc. IEEE International Conference on Computer Vision*, pages 66–75, 2017.

[52] L. Keselman, J.I. Woodfill, A. Grunnet-Jepsen, and A. Bhowmik. Intel(R) realsense(TM) stereoscopic depth cameras. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 1267–1276, 2017.

[53] S. Khamis, S. Fanello C. Rhemann, A. Kowdle, J. Valentin, and S. Izadi. StereoNet: Guided hierarchical refinement for real-time edge-aware depth prediction. In *Proc. European Conference on Computer Vision*, pages 573–590. Springer International Publishing, 2018.

[54] S. Kim, D. Yoo, and Y. H. Kim. Stereo confidence metrics using the costs of surrounding pixels. In *Proc. International Conference on Digital Signal Processing*, pages 98–103, 2014.

[55] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *Proc. International Conference on Learning Representations*, pages 1–15, 2015.

[56] A. Klaus, M. Sormann, and K. Karner. Segment-based stereo matching using belief propagation and a self-adapting dissimilarity measure. In *Proc. International Conference on Pattern Recognition*, volume 3, pages 15–18, 2006.

[57] V. Kolmogorov and R. Zabih. Computing visual correspondence with occlusions using graph cuts. In *Proc. IEEE International Conference on Computer Vision*, volume 2, pages 508–515, 2001.

[58] Z. Lee, J. Juang, and T. Q. Nguyen. Local disparity estimation with three-moded cross census and advanced support weight. *IEEE Transactions on Multimedia*, 15(8):1855–1864, 2013.

[59] Ang Li and Zejian Yuan. Occlusion aware stereo matching via cooperative unsupervised learning. In *Proc. Asian Conference on Computer Vision*, pages 197–213, 2018.

[60] B. Li, C. Shen, Y. Dai, A. van den Hengel, and M. He. Depth and surface normal estimation from monocular images using regression on deep features and hierarchical crfs. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages 1119–1127, 2015.

[61] C.-Y. Liu, Z.-W. Chang, and C.-L. Chen. Visual servo tracking of a wheeled mobile robot utilizing both feature and depth information. In *Proc. IEEE Conference on Industrial Electronics and Applications*, pages 1045–1050, 2019.

[62] P. Liu, I. King, M. R. Lyu, and J. Xu. Flow2Stereo: Effective self-supervised learning of optical flow and stereo matching. In *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6647–6656, 2020.

[63] D.G. Lowe. Object recognition from local scale-invariant features. In *Proc. IEEE International Conference on Computer Vision*, volume 2, pages 1150–1157, 1999.

[64] A. L. Maas, A. Y. Hannun, and A. Y. Ng. Rectifier nonlinearities improve neural network acoustic models. In *Proc. International Conference on Machine Learning*

Workshop on Deep Learning for Audio, Speech and Language Processing, pages 1–6, 2013.

[65] E. Marchand. Direct visual servoing in the frequency domain. *IEEE Robotics and Automation Letters*, 5(2):620–627, 2020.

[66] N. Mayer, E. Ilg, P. Hausser, P. Fischer, D. Cremers, A. Dosovitskiy, and T. Brox. A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In *Proc. IEEE International Conference on Computer Vision and Pattern Recognition*, 2016.

[67] X. Mei, X. Sun, M. Zhou, S. Jiao, H. Wang, and X. Zhang. On building an accurate stereo matching system on graphics hardware. In *Proc. IEEE International Conference on Computer Vision Workshops*, pages 467–474, 2011.

[68] M. Menze and A. Geiger. Object scene flow for autonomous vehicles. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages 3061–3070, 2015.

[69] Open Source Robotics Foundation. TurtleBot2. https://www.turtlebot.com/turtlebot2/. accessed: Oct. 15, 2021.

[70] Open Source Robotics Foundation. TurtleBot3. https://www.turtlebot.com/turtlebot3/. accessed: Jun. 7, 2022.

[71] OpenCV. cv::StereoSGBM class reference. https://docs.opencv.org/3.4.4/d2/d85/classcv_1_1StereoSGBM.html, 2018. accessed: Oct. 3, 2021.

[72] OpenCV. cv::StereoBM class reference. https://docs.opencv.org/3.4.4/d9/dba/classcv_1_1StereoBM.html, Nov. 17, 2018. accessed: Mar. 29, 2022.

[73] M. Ourak, B. Tamadazte, O. Lehmann, and N. Andreff. Direct visual servoing using wavelet coefficients. *IEEE/ASME Transactions on Mechatronics*, 24(3):1129–1140, 2019.

[74] M. Park and K. Yoon. Learning and selecting confidence measures for robust stereo matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 41(6):1397–1411, 2019.

[75] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, Edward Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. Py-

Torch: An imperative style, high-performance deep learning library. In *Proc. Advances in Neural Information Processing Systems*, pages 8024–8035, 2019.

[76] M. Poggi, S. Kim, F. Tosi, S. Kim, F. Aleotti, D. Min, K. Sohn, and S. Mattoccia. On the confidence of stereo matching in a deep-learning era: a quantitative evaluation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–20, 2021. doi: 10.1109/TPAMI.2021.3069706.

[77] M. Poggi and S. Mattoccia. Learning from scratch a confidence measure. In *Proc. British Machine Vision Conference*, pages 46.1–46.13, 2016.

[78] M. Quigley, B. Gerkey, K. Conley, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler, and A. Ng. ROS: an open-source robot operating system. In *Proc. IEEE International Conference on Robotics and Automation Workshop on Open Source Robotics*, 2009.

[79] A. Roy and S. Todorovic. Monocular depth estimation using neural regression forest. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages 5506–5514, 2016.

[80] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. ORB: An efficient alternative to sift or surf. In *Proc. International Conference on Computer Vision*, pages 2564–2571, 2011.

[81] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

[82] D. Scharstein, R. Szeliski, and R. Zabih. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. In *Proc. IEEE Workshop on Stereo and Multi-Baseline Vision*, pages 131–140, 2001.

[83] A. Shademan and F. Janabi-Sharifi. Using scale-invariant feature points in visual servoing. In *Machine Vision and its Optomechatronic Applications*, volume 5603, pages 63 – 70, 2004.

[84] A. Spyropoulos, N. Komodakis, and P. Mordohai. Learning to detect ground control points for improving the accuracy of stereo matching. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages 1621–1628, 2014.

[85] StereoLabs. ZED stereo camera. https://www.stereolabs.com/zed/, 2021. accessed: Oct. 3, 2021.

[86] R. Szeliski. *Computer Vision: Algorithms and Applications*. Sringer, London, 2010.

[87] C. Teulière and E. Marchand. A dense and direct approach to visual servoing using depth maps. *IEEE Transactions on Robotics*, 30(5):1242–1249, 2014.

[88] B. Thuilot, P. Martinet, L. Cordesses, and J. Gallice. Position based visual servoing: keeping the object in the field of vision. In *Proc. IEEE International Conference on Robotics and Automation*, volume 2, pages 1624–1629, 2002.

[89] A. Tonioni, M. Poggi, S. Mattoccia, and L. Di Stefano. Unsupervised domain adaptation for depth prediction from images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(10):2396–2409, 2020.

[90] A. Tonioni, F. Tosi, M. Poggi, S. Mattoccia, and L. Di Stefano. Real-time self-adaptive deep stereo. In *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 195–204, 2019.

[91] F. Tosi, M. Poggi, A. Benincasa, and S. Mattoccia. Beyond local reasoning for stereo confidence estimation with deep learning. In *Proc. European Conference on Computer Vision*, pages 319–334, 2018.

[92] S. van der Walt, J. L. Schönberger, J. Nunez-Iglesias, F. Boulogne, J. D. Warner, N. Yager, E. Gouillart, T. Yu, and the scikit-image contributors. scikit-image: image processing in Python. *PeerJ*, 2:e453, 2014.

[93] Velodyne. Alpha Prime. https://velodynelidar.com/products/alpha-prime/. accessed: May 20, 2021.

[94] H. Wang, R. Fan, P. Cai, and M. Liu. PVStereo: Pyramid voting module for end-to-end self-supervised stereo matching. *IEEE Robotics and Automation Letters*, 6(3):4353–4360, 2021.

[95] L. Wang, Y. Guo, Y. Wang, Z. Liang, Z. Lin, J. Yang, and W. An. Parallax attention for unsupervised stereo correspondence learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(4):2108–2125, 2022.

[96] Z. Wang, A.C. Bovik, H.R. Sheikh, and E.P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004.

[97] W.J. Wilson, C.C. Williams Hulls, and G.S. Bell. Relative end-effector control using cartesian position based visual servoing. *IEEE Transactions on Robotics and Automation*, 12(5):684–696, 1996.

[98] D. Wofk, F. Ma, T. Yang, S. Karaman, and V. Sze. FastDepth: Fast monocular depth estimation on embedded systems. In *Proc. IEEE International Conference on Robotics and Automation*, pages 6101–6108, 2019.

[99] Y. Xiong, X. Zhang, J. Peng, and W. Yu. 3D depth map based optimal motion control for wheeled mobile robot. In *Proc. IEEE International Conference on Systems, Man and Cybernetics*, pages 2045–2050, 2017.

[100] H. Xu and J. Zhang. AANet: Adaptive aggregation network for efficient stereo matching. In *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1956–1965, 2020.

[101] Z. Yu, L. Jin, and S. Gao. P^2Net: Patch-match and plane-regularization for unsupervised indoor depth estimation. In *Proc. European Conference on Computer Vision*, pages 206–222. Springer International Publishing, 2020.

[102] J. Zbontar and Y. LeCun. Computing the stereo matching cost with a convolutional neural network. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages 1592–1599, 2015.

[103] F. Zhang, V. Prisacariu, R. Yang, and P.H.S. Torr. GA-Net: Guided aggregation net for end-to-end stereo matching. In *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 185–194, 2019.

[104] J. Zhang, K. A. Skinner, R. Vasudevan, and M. Johnson-Roberson. DispSegNet: Leveraging semantics for end-to-end learning of disparity estimation from stereo imagery. *IEEE Robotics and Automation Letters*, 4(2):162–1169, 2019.

[105] K. Zhang, J. Lu, Q. Yang, G. Lafruit, R. Lauwereins, and L. Van Gool. Real-time and accurate stereo: A scalable approach with bitwise fast voting on CUDA. *IEEE Transactions on Circuits and Systems for Video Technology*, 21(7):867–878, 2011.

[106] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia. Pyramid scene parsing network. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages 6230–6239, 2017.

[107] S. Zhao, Y. Sheng, Y. Dong, E.I. Chang, and Y. Xu. MaskFlownet: Asymmetric feature matching with learnable occlusion mask. In *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6277–6286, 2020.

[108] H. Zhong, H. Wang, Z. Wu, C. Zhang, Y. Zheng, and T. Tang. A survey of LiDAR and camera fusion enhancement. *Procedia Computer Science*, 183:579–588, 2021.

[109] C. Zhou, H. Zhang, X. Shen, and J. Jia. Unsupervised learning of stereo matching. In *Proc. IEEE International Conference on Computer Vision*, pages 1576–1584, 2017.

[110] J. Zhou, Y. Wang, K. Qin, and W. Zeng. Moving indoor: Unsupervised video depth learning in challenging environments. In *Proc. IEEE/CVF International Conference on Computer Vision*, pages 8617–8626, 2019.

[111] T. Zhou, M. Brown, N. Snavely, and D. G. Lowe. Unsupervised learning of depth and ego-motion from video. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages 6612–6619, 2017.